# SEMANTIC ROLE LABELING OF IMPLICIT ARGUMENTS FOR NOMINAL PREDICATES

By

Matthew Steven Gerber

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2011

ABSTRACT

SEMANTIC ROLE LABELING OF IMPLICIT ARGUMENTS
FOR NOMINAL PREDICATES

By

Matthew Steven Gerber

Natural language is routinely used to express the occurrence of an event and existence of entities that participate in the event. The entities involved are not haphazardly related to the event; rather, they play specific roles in the event and relate to each other in systematic ways with respect to the event. This basic semantic scaffolding permits construction of the rich event descriptions encountered in spoken and written language. Semantic role labeling (SRL) is a method of automatically identifying events, their participants, and the existing relations within textual expressions of language. Traditionally, SRL research has focused on the analysis of verbs due to their strong connection with event descriptions. In contrast, this dissertation focuses on emerging topics in noun-based (or nominal) SRL.

One key difference between verbal and nominal SRL is that nominal event descriptions often lack participating entities in the words that immediately surround the predicate (i.e., the word denoting an event). Participants (or arguments) found at longer distances in the text are referred to as implicit. Implicit arguments are relatively uncommon for verbal predicates, which typically require their arguments to appear in the immediate vicinity. In contrast, implicit arguments are quite common for nominal predicates. Previous research has not systematically investigated implicit argumentation, whether for verbal or nominal predicates. This dissertation shows that implicit argumentation presents a significant challenge to nominal SRL systems: after introducing implicit argumentation into the evaluation, the state-of-the-art nominal SRL system presented in this dissertation suffers a performance degradation of more than 8%.

Motivated by these observations, this dissertation focuses specifically on implicit argumentation in nominal SRL. Experiments in this dissertation show that the aforementioned

performance degradation can be reduced by a discriminative classifier capable of filtering out nominals whose arguments are implicit. The approach improves performance substantially for many frequent predicates - an encouraging result, but one that leaves much to be desired. In particular, the filter-based nominal SRL system makes no attempt to identify implicit arguments, despite the fact that they exist in nearly all textual discourses.

As a first step toward the goal of identifying implicit arguments, this dissertation presents a manually annotated corpus in which nominal predicates have been linked to implicit arguments within the containing documents. This corpus has a number of unique properties that distinguish it from preexisting resources, of which few address implicit arguments directly. Analysis of this corpus shows that implicit arguments are frequent and often occur within a few sentences of the nominal predicate.

Using the implicit argument corpus, this dissertation develops and evaluates a novel model capable of recovering implicit arguments. The model relies on a variety of information sources that have not been used in prior SRL research. The relative importance of these information sources is assessed and particularly troubling error types are discussed. This model is an important step forward because it unifies work on traditional verbal and nominal SRL systems. The model extracts semantic structures that cannot be recovered by applying the systems independently.

Building on the implicit argument model, this dissertation then develops a preliminary joint model of implicit arguments. The joint model is motivated by the fact that semantic arguments do not exist independently of each other. The presence of a particular argument can promote or inhibit the presence of another. Argument dependency is modeled by using the TextRunner information extraction system to gather general purpose knowledge from millions of Internet webpages. Results for the joint model are mixed; however, a number of interesting insights are drawn from the study.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Joyce Chai, who has provided many years of unwavering intellectual and moral support. Joyce has worked tirelessly to keep me on track and has done a wonderful job of separating the promising research directions from the far fetched ones that I occasionally put forward. I am fortunate to have had her as my primary collaborator. I am also grateful for the many friendships I formed in East Lansing over the years. To my lab mates, thanks for making EB3315 a fun and (usually) productive place. Our many intense but ultimately frivolous discussions were always a welcome diversion. To everyone else, including those who have long since graduated and moved away, know that I will remember all of the poker nights and other gatherings with particular fondness. Of course, none of this work would have happened without the lifelong love and incredible support of my parents Randy and Dee Ann, who always remained curious to learn about my research. And finally, to my wife Amanda: thank you for putting up with me for the last nine months as I have worked on this dissertation. You deserve an award. Thank you for reminding me to eat. Thank you for showing me that there is more to life than research. **Thank you.**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER 1
# Introduction

Automatic textual analysis has successfully dealt with many aspects of the information explosion that has taken place over the last few decades. One of the most prominent types of textual analysis is what I will be referring to as non-semantic analysis. This type of analysis often manifests itself in the ubiquitous bag-of-words (and related) models of natural language. Such models are not typically concerned with the underlying meaning of text, as evidenced by their use of stemming, stop word removal, and a host of other techniques that trade semantic information for improved statistical information. While immensely successful when dealing with large objects of interest (e.g., web pages, blog entries, PDFs, etc.), these approaches do not perform well when the information need is described by a question or imperative, or when the information need is a small object such as a concise answer or the identification of an entity-entity relationship. More generally, non-semantic analyses perform relatively poorly in situations that require semantic understanding and inference.

In an effort to fill this semantic gap, the research community has proposed a wide range of resources and methods. Some of these address the semantic properties of individual words, whereas others target the semantic properties of entire sentences or discourses. Below, I give examples at a few points along this spectrum. As one moves down the list, larger units of text are analyzed in greater semantic detail.

**Words and phrases**

- Word sense disambiguation (Joshi et al., 2006): determining whether the word "bank" in a particular context refers to a mound of earth or a financial institution.
- Named entity identification (Bikel et al., 1999): identifying particular classes of

entities, for example, people or countries.

- Lexical semantics (Fellbaum, 1998): identifying semantic relationships between concepts, for example, the fact that all humans are mammals.

**Short-distance relationships**

- Relations between nominals (Girju et al., 2007): identifying a product-producer relation between the words in a phrase such as "honey bee".
- Temporal relations (Verhagen et al., 2007): identifying the relationship between a temporal expression (e.g., "yesterday") and the events mentioned in a sentence.

**Shallow sentential meaning**

- Event extraction (ACE, 2007): identification of a few specific event types.
- Semantic role labeling (Surdeanu et al., 2008): identifying a large number of event types and their participants within a sentence.

**Deep sentential meaning**

- First-order semantics (Bos, 2005; Mooney, 2007): transforming natural language text into a first-order logic representation.

Tasks near the bottom of this list are generally more difficult because they require detailed analyses of large text fragments; however, these analyses provide a more complete semantic picture of natural language expressions.

This dissertation focuses on a specific type of shallow sentential meaning called semantic role labeling (SRL). In the SRL paradigm, a predicate word (typically denoting an event) is bound to various entities in the surrounding text by means of relationships that describe the entities' roles in the event. Consider the following example:

(1.1) [*Sender* John] [*Predicate* shipped] [*Thing_shipped* a package] [*Source* from Michigan] [*Destination* to California].

In this example, *John* is purposely acting to ship *a package* from its source location (*Michigan*) to its destination location (*California*). The goal of automatic SRL is to identify the predicates and role-filling constituents that together provide a basic understanding of a sentence's event structure.

Traditionally, automatic SRL research has focused on verb predicates due to their strong connection with event descriptions; however, recent years have witnessed an increased emphasis on SRL for nouns, which frequently denote events and are also amenable to role analysis. Although they are related, nominal and verbal SRL exhibit important differences that must be taken into account. One key difference is that nominal SRL structures often lack argument fillers that would normally be required for the corresponding verbal SRL structure. Consider the following variant of Example 1.1:

(1.2) [*Sender* John] made a [*Predicate* shipment] [*Source* from Michigan] [*Destination* to California].

The nominal predicate in Example 1.2 does not require the *Thing_shipped* to be overtly expressed in the sentence, whereas the verbal predicate in Example 1.1 does (it is ungrammatical otherwise). When the *Thing_shipped* (or any other argument) is expressed elsewhere in the discourse, we have an instance of implicit argumentation. In general, implicit argumentation is extremely common for nominal predicates, but the research community has paid very little attention to it. This dissertation focuses specifically on the issue of implicit argumentation in nominal SRL.

I begin by developing a nominal SRL system capable of producing analyses similar to Example 1.2. When given a predicate known to take arguments in the current sentence, this system is able to recover the arguments with an F-measure ($\beta = 1$) score of 75.7%. This is a state-of-the-art result for the task; however, the practice of supplying a system with an argument-bearing nominal has serious implications due to implicit argumentation. When evaluated over all predicates (including those whose arguments are entirely implicit), the same system achieves an argument $F_1$ score of only 69.3%.

In an attempt to address the issue of implicit argumentation, I develop a model that is able to accurately ($F_1 = 87.6\%$) identify nominal predicates with explicit arguments, effectively filtering out predicates whose arguments are implicit. This model pushes the argument $F_1$ score to 71.1% for all nominal predicates. The model more than doubles the argument identification performance for particular groups of frequent nominal predicates. These are encouraging results, but they leave much to be desired. In particular, the nominal SRL system does not attempt to recover implicit arguments, which are often present in the surrounding discourse.

Motivated by the results described above, I investigate the automatic identification of implicit arguments using information from a predicate's sentence and surrounding discourse. This represents a dramatic departure from traditional SRL approaches, which, for a given predicate, do not look past sentence boundaries for argument fillers. I base my investigation on a corpus of manually annotated implicit arguments. This corpus is one of the first of its kind and has been made freely available for research purposes. Using this corpus, I show that implicit arguments constitute a significant portion of the semantic structure of a document; they are frequent, often located within a few sentences of their respective predicates, and they provide information that cannot be recovered using standard verbal and nominal SRL techniques.

Given their importance, it is interesting to note that very little attention has been paid to the automatic recovery of implicit arguments. I address this issue by developing a model that is capable of identifying implicit arguments across sentence boundaries. Whereas traditional SRL models have relied primarily on syntactic information, the implicit argument model relies primarily on semantic information. This information comes from a variety of sources, many of which have not previously been explored. Overall, the implicit argument model achieves an $F_1$ score of approximately 50%. This result represents the current state-of-the-art, since the task of implicit argument identification is a new one within the field.

The implicit argument model described above simplifies the modeling task by assuming

that implicit arguments are independent of each other. Each candidate is classified independently of the other candidates, and a heuristic post-processing procedure is applied to arrive at the final configuration. I present a preliminary investigation of this assumption in which implicit arguments are identified in a joint fashion. The model relies, in part, on knowledge extracted from millions of Internet webpages. This knowledge serves to identify likely joint occurrences of implicit arguments. Evaluation results for this model are mixed; however, they suggest a variety of interesting future directions.

This dissertation is organized as follows. In Chapter 2, I review the theoretical status of semantic roles as well as previous SRL research. In the same chapter, I present the basic nominal SRL system mentioned above. Chapter 3 begins by introducing implicit argumentation in more detail and assessing its implications for the basic nominal SRL system. The chapter then provides a detailed description and evaluation of the nominal filtering model. Chapter 4 begins with an empirical analysis of nominal event structure, which is found to be largely implicit. As part of this analysis, I describe in detail the implicit argument annotation effort I conducted. The chapter then presents and evaluates the model for implicit argument identification. Chapter 5 presents the exploration of joint implicit argument modeling. I conclude, in Chapter 6, with a summary of contributions and future work.

# CHAPTER 2
# Nominal semantic role labeling

## 2.1 Introduction

The notion of semantic role (variously referred to as thematic relation, thematic role, and theta role) has enjoyed a long and occasionally contentious history within linguistics. Gruber (1965), in an analysis of motion verbs, observed that certain semantic properties apply to the entity undergoing motion, regardless of that entity's surface syntactic position. For example, consider the following alternations of the verb *throw*:

(2.1) John threw a ball to Mary.
(2.2) John threw Mary a ball.
(2.3) A ball was thrown to Mary by John.
(2.4) A ball was thrown by John to Mary.
(2.5) To Mary was thrown a ball by John.
...

In all cases, *a ball* is the entity undergoing motion; however, this entity fills the syntactic object position in Example 2.1 and the syntactic subject position in Example 2.3. Gruber introduced the term *Theme* to denote objects that passively undergo such actions, and made similar generalizations for other event participants. For example, *John* fills the role of *Agent* in the examples because he is the intentional causer of the event. Mary, to whom John is throwing the ball, fills the role of *Recipient* in the examples. These roles reflect underlying semantic properties of the entities within the context of the *throw* event. Semantic roles, with their power to generalize over numerous syntactic constructions, have been of great interest to a variety of researchers in fields from linguistics to philosophy to natural language

processing (NLP).

However, as alluded to above, semantic roles are not uncontroversial. A wide-ranging debate has raised questions about the composition and requisite number of semantic roles. The case grammar of Fillmore (1968) and the frame-based semantics of Fillmore (1976) each posit a large number of specific semantic roles. Following these theories, entities in text are assigned a specific role based on their relation to the event under consideration. On the other end of the spectrum, Dowty (1991) posits only two roles: proto-agent and proto-patient. These two proto-roles are composed of many different "contributing properties" that entities assigned to them should possess. For example, an entity assigned to the proto-agent role should be volitional and sentient, whereas an entity assigned to the proto-patient role should involuntarily undergo a change of state. Constituents are assigned to these roles in a graded fashion depending on how many of the relevant properties they possess. These properties, though, are no more agreed upon than the various semantic roles mentioned above, so the controversy would seem far from being resolved.

Despite a lack of consensus on finer points, semantic roles have much to offer automatic natural language understanding systems. As demonstrated by Examples 2.1-2.5, semantic roles generalize over the myriad ways in which an event may be described. Thus, because events play a central role in everyday language use, the automatic identification of semantic roles should prove helpful in many NLP tasks. In general, the task of automatic semantic role labeling (SRL) is defined as follows:

> **Automatic SRL task:** Given some unstructured text, identify the events and the fillers of the events' semantic roles.[1]

Figure 2.1 shows the position of an automatic SRL system with respect to unstructured text (the input) and target applications that make use of structured information (the SRL output). The figure includes three target applications to which SRL has been successfully

[1]In this dissertation, I will use the terms *event* and *predicate* interchangeably. I will do the same for *semantic role* and *argument*. Thus, I will also use predicate-argument identification to refer to the SRL task.

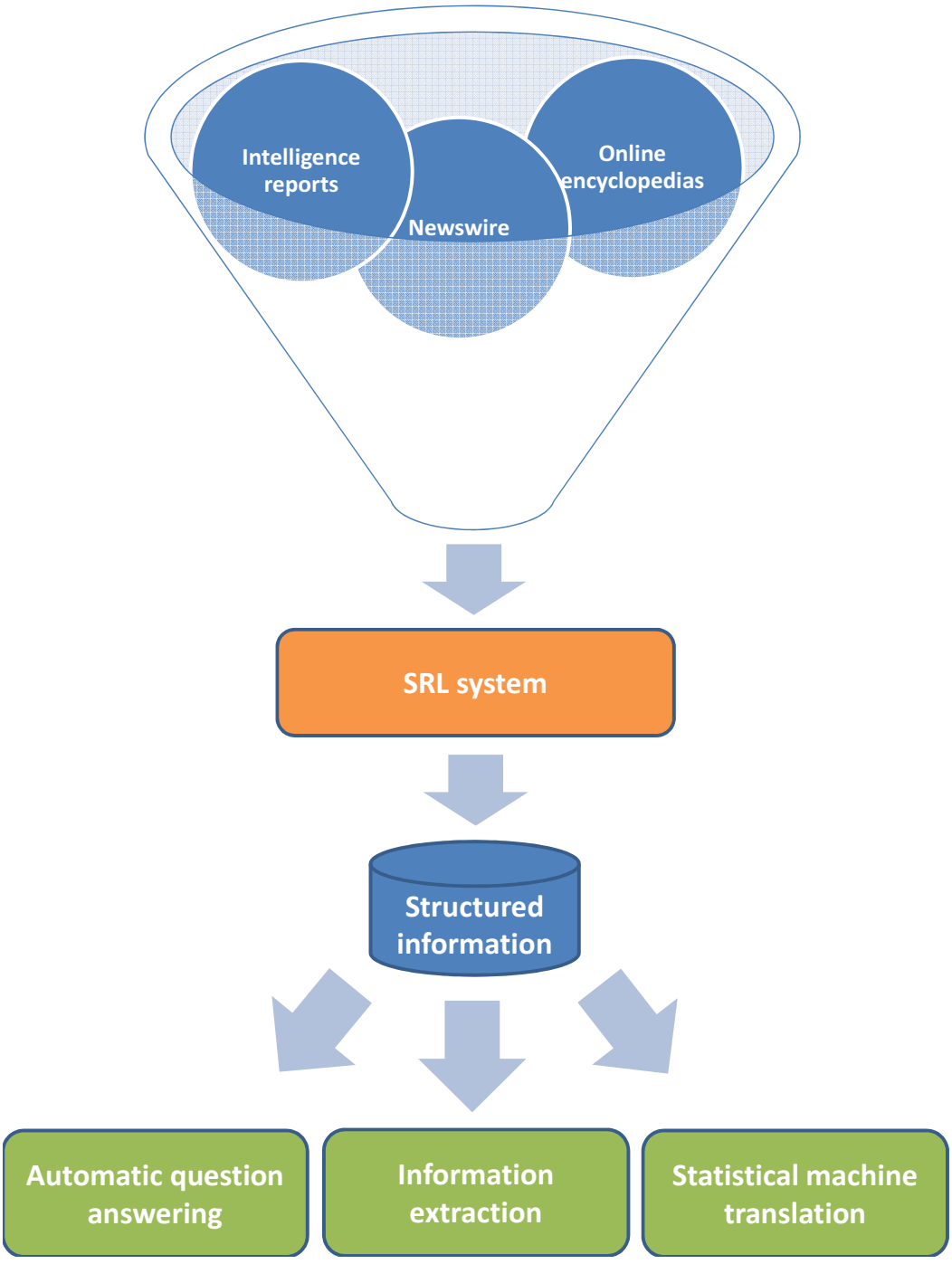



Figure 2.1: Position of an SRL system with respect to target applications. Unstructured information flows in from the top. The SRL system identifies structure within this information, which is consumed by target applications. *For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.*

applied. Below, I describe these applications and the corresponding role of SRL.

**Automatic question answering (QA)** is the task of providing a precise answer to a user's question. Assume the following question has been issued:

(2.6) Who invented the polio vaccine?

Traditional Internet search engines are not suitable for directly answering 2.6 because they often return ranked lists of documents instead of precise answers. An SRL-based approach, on the other hand, might proceed as follows:

1. Query a large corpus of documents for exact matches to "invented the polio vaccine".
2. Perform SRL on the returned sentences.
3. Identify and filter the *Agent* roles for the *invent* events, returning the single best as the answer to the question.

Configured this way, the system stands a chance of returning the correct answer: "Jonas Salk". Kaisser and Webber (2007) and Pizzato and Mollá (2008) have shown that automatic QA can benefit from SRL information.

**Information extraction (IE)** is the task of identifying facts, relations, events, and other types of information within unstructured documents. Recently, there has been a surge of interest in Web-scale IE, where information is extracted from millions of documents. Banko et al. (2007) developed the Open IE (OIE) methodology to extract an open set of semantic relations from text in an unsupervised fashion. The standard OIE approach tends to be a low precision, high recall process. Supervised SRL, on the other hand, tends to be a high precision, low recall process, particularly for out-of-domain data in which previously unseen events are encountered. Banko and Etzioni (2008) showed how methods similar to SRL can be combined with the standard OIE approach, yielding a hybrid system with the advantages

of both sub-systems.

**Statistical machine translation (SMT)** is a classic NLP task in which a system must translate a text from the original source language $S$ to a target language $T$. A simple approach to this task is to translate each word, possibly reordering it in the target language sentence according to a distortion probability. Recently, researchers have shown that integrating syntactic information into the model can have a positive impact on translation performance (May and Knight, 2007). Following this work, Liu et al. (2010) demonstrated that SRL information can also help improve translation performance. In both cases, the translation system used the additional information to filter out less plausible translation results that contain either uncommon syntactic constructions or uncommon semantic argument arrangements.

For the three tasks described above, researchers found that system performance increased when taking SRL-based information into account. More specifically, these systems used information derived from *verbal* SRL analyses. Verbal SRL (demonstrated in Examples 2.1-2.5) is based on predicates that take the form of verbs within a sentence. Historically, semantic roles have been associated with verbs for the simple reason that nearly all verbs have semantic roles. However, other parts of speech are associated with semantic roles in precisely the same manner as verbs. This dissertation focuses on semantic roles associated with predicates in noun form (i.e., nominal predicates). To see the parallel between verbal and nominal SRL, consider the following examples:

(2.7) Freeport-McMoRan Energy Partners will be liquidated and [*Theme* shares of the new company] [*Predicate* distributed$_{(verb)}$] [*Destination* to the partnership's unitholders].[2]

(2.8) Searle will give [*Destination* pharmacists] [*Theme* brochures on the use of prescription drugs] for [*Predicate* distribution$_{(noun)}$] in their stores.[3]

[2]Borrowed from Kingsbury and Palmer (2003)

[3]Borrowed from Meyers (2007a)

Example 2.7 uses a verbal form of *distribute*, and Example 2.8 uses a nominal form. As expected, the semantic properties of interest (i.e., those related to the *distribute* event) hold for fillers of the semantic roles regardless of the fillers' syntactic positions or the parts of speech of their respective predicates. This is a key observation because it suggests that tasks like QA, IE, and SMT might benefit from nominal SRL just as they do from verbal SRL. At least, this might be the case if nominal predicates are also as frequent as verbal predicates. As shown in Section 2.2.2, nominal predicates are on average more frequent per document than verbal predicates.

The remainder of this chapter is structured as follows. In the next section, I review work related to the tasks of verbal and nominal SRL, paying special attention to the latter as it is the focus of this dissertation. Then, in Section 2.3, I present a nominal SRL system inspired by previous work that significantly improves the state-of-the-art, as shown in Section 2.4. This work sets the stage for a more in-depth investigation into nominal SRL, which is taken up in subsequent chapters.

## 2.2 Related work

As in many other NLP tasks, research in semantic role labeling has progressed from hand-crafted rule systems based on human engineering to statistical systems based on supervised and unsupervised machine learning. In this section, I give a brief history of this progression, starting with rule-based systems in Section 2.2.1. I then give an overview of the relevant supervised training corpora in Section 2.2.2, followed by recent statistical SRL work in Section 2.2.3. The nominal SRL model developed in Section 2.3 draws on many of the techniques presented in this section.

### 2.2.1 Rule-based SRL

As noted above, early models of language semantics typically relied on large compilations of hand-coded processing rules and world knowledge. For example, much of the work done by Hirst (1987) relied on a rule-based syntactic parser and a frame-based knowledge representation similar to the one developed by Fillmore (1976). Hirst used a mapping to link syntactic constituents to their respective frame positions, and the sentence's semantic representation was built up compositionally. A similar emphasis on hand-coded lexicons and grammars can be found in the work of Pustejovsky (1995) and Copestake and Flickinger (2000), respectively.

Early work in identifying nominal argument structure used approaches similar to those discussed above. For example, Dahl et al. (1987), Hull and Gomez (1996), and Meyers et al. (1998) each employ sets of rules that associate syntactic constituents with semantic roles for nominal predicates. Consider the following example from Dahl et al. (p. 135):

(2.9) Investigation revealed [*Instrument* metal] [*Predicate* contamination] in [*Theme* the filter].

The system created by Dahl et al. used the following rules to identify the contaminating substance (*metal*) and the contaminated entity (*the filter*):

1. The *Instrument* can be the noun preceding the predicate *contamination*.

2. The *Theme* can be the object of the prepositional phrase following *contamination*.

The rules defined above allow the system to properly identify the fillers of semantic roles in Example 2.9. This system was not formally evaluated, but it is reasonable to believe that the rules described above would often be correct when triggered.

The rules in Dahl et al.'s work have advantages and disadvantages that are common to rule-based semantics systems. On one hand, if a precise rule produces a prediction, that prediction is likely to be correct (e.g., the identification of the *Instrument* and *Theme* above).

Furthermore, the rule sets are explanatorily powerful, as any derivation can be explained in terms of the rules that produced it. However, on the other hand, the systems described above tend to be brittle, particularly when used in novel domains or on genres of text not anticipated by the rule creators. This is the result of the all-or-nothing nature of rule-based syntactic and semantic interpretation. Given the great versatility of language, it should come as no surprise that, in many cases, a limited set of rules fails to apply (i.e., interpret) a natural language utterance. Furthermore, as noted by Copestake and Flickinger (2000), the learning curve for working with and extending some rule-based grammar systems can be prohibitively steep, making it difficult to apply such systems to new domains.

In contrast to hand-coding the behavior of the analyzer as described above, this dissertation develops methods whose behaviors are determined by supervised machine learning. As described in Section 2.3, this approach allows one to identify the optimal system behavior in a flexible, automated fashion while relying on hand-coded information that is less expensive to obtain and more likely to be agreed upon across human annotators. The following section describes a few of these annotated resources, all of which are used in this dissertation.

### 2.2.2 Annotated corpora for SRL

#### FrameNet

As mentioned previously, Fillmore (1968) developed a theory of grammar in which syntactic constituents stand in various case relations with their predicates. Example cases include *Agent* and *Instrument*, which correspond to the similarly named semantic roles presented earlier. Fillmore's case theory was later refined by grouping cases into larger units termed frames (Fillmore, 1976). For example, in the *Buy* frame, one finds a *Buyer*, *Seller*, *Goods*, *Money*, etc. Each frame is also associated with a number of predicates (e.g., buy, purchase, barter) that instantiate it within a sentence.

FrameNet (Baker et al., 1998) is a machine-readable resource created by identifying and relating Fillmore's frames and documenting their presence within natural language text. In

FrameNet, case roles are called frame elements and predicates are called lexical items. These lexical items can be verbs, nouns, or adjectives. For example, consider the *Transfer* frame, shown with a few of its frame elements and lexical items:

> The *Transfer* frame
>
> Donor: the person that begins in possession of the Theme and causes it to be in the possession of the Recipient
>
> Theme: the object that changes ownership
>
> Recipient: the entity that ends up in possession of the Theme
>
> Purpose: the purpose for which the Theme is transferred
>
> Lexical items: transfer.n, transfer.v

FrameNet arranges frames into a network by defining frame-to-frame relationships such as inheritance and causation. For example, consider the *Commerce goods-transfer* frame, which inherits from the *Transfer* frame:

> The *Commerce goods-transfer* frame (inherits from the *Transfer* frame)
>
> Seller (from Transfer.Donor): entity in possession of Goods and exchanging them for Money with a Buyer
>
> Goods (from Transfer.Theme): anything that is exchanged for Money in a transaction
>
> Purpose (from Transfer.Purpose): the purpose for which a Theme is transferred
>
> Buyer (from Transfer.Recipient): entity that wants the Goods and offers Money to a Seller in exchange for them
>
> Money (new in this frame): the thing given in exchange for Goods in a transaction

As shown above, the inheritance relation allows a general frame (e.g., *Transfer*) to be specialized with a particular semantic interpretation (e.g., the transfer of commercial goods). Where applicable, the inheritance relationship also holds between the frame elements of the related frames. This is indicated above, with *Seller* inheriting from *Donor*, *Goods* inheriting from *Theme*, *Purpose* inheriting from *Purpose*, and *Buyer* inheriting from *Recipient*. Each of the inheriting frame elements contains all semantic properties of the inherited frame elements and possibly adds additional semantic properties. The two frames also show that sub-frames may provide additional frame elements (e.g., *Money*) for the frame specialization. In total, version 1.5 of FrameNet defines 1,019 frames related with 12 different relation types.

Having established a network of frames, the FrameNet annotators manually identified instances of the frames within the British National Corpus.[4] Consider the following annotation of the *Commerce goods-transfer* frame:

(2.10) Four years ago [*Buyer* I] [*Predicate* bought] [*Goods* an old Harmony Sovereign acoustic guitar] [*Money* for £20] [*Seller* from an absolute prat].

As shown in Example 2.10, not all frame elements are present in each frame annotation. Furthermore, the annotators have only identified frame elements within the sentence containing the predicate. In total, FrameNet contains approximately 150,000 annotated frame instances. As shown below in Section 2.2.3, these annotated examples can be used as supervised learning material for systems that automatically identify frames and frame elements within text.

**VerbNet**

The work of Kipper et al. (2000) (described more fully by Kipper (2005)) coincided roughly with the development of FrameNet. This work, inspired by the analysis of so-called verb classes by Levin (1993), resulted in a computer-readable lexicon of verb argument specifications called VerbNet. In VerbNet, verbs are collected into classes. The members of each class exhibit the same diathesis alternations, or meaning preserving transformations. An example alternation, the *causative-inchoative*, is shown below:

(2.11) [*Agent* John] [*Predicate* broke] [*Theme* the window]. (causative)
(2.12) [*Theme* The window] [*Predicate* broke]. (inchoative)

With respect to Examples 2.11 and 2.12, Levin and Kipper et al. made the following key observations:

1. The causative-inchoative alternation is (mostly) meaning preserving. That is, Example 2.11 has roughly the same semantic interpretation as Example 2.12. This instance

[4] http://www.natcorp.ox.ac.uk

of the alternation is not completely meaning preserving because the *Agent* remains unspecified in Example 2.12.

2. Other verbs that are capable of undergoing the causative-inchoative alternation also appear to indicate a change of state (and vice versa). For example, *close* indicates a change of state and may undergo the alternation, as shown below:

   (a) [*Agent* John] [*Predicate* closed] [*Theme* the window].
   (b) [*Theme* The window] [*Predicate* closed].

   The verb *hit*, on the other hand, does not indicate a change of state and thus cannot undergo the causative-inchoative alternation:[5]

   (a) [*Agent* John] [*Predicate* hit] [*Theme* the window].
   (b) *[*Theme* The window] [*Predicate* hit].

Each VerbNet class defines a set of semantic roles used by verbs in the class. Furthermore, verb classes are arranged into an inheritance tree, such that sub-classes inherit the roles of super-classes (similarly to FrameNet). Currently, version 3.1 of VerbNet groups 5,725 verbs into 438 classes. This resource does not annotate instances of the verbs it contains; however, it is important because it relates semantically similar verbs to each other - a fact that will be used in Chapter 4 when predicate-predicate relations are examined.

### PropBank

To document the different ways in which verbs can express their arguments, Kingsbury and Palmer (2003) annotated semantic role information for all main verbs in the Penn TreeBank (Marcus et al., 1993). The Penn TreeBank is a corpus of English newswire text that has been annotated for syntactic structure by humans. Kingsbury and Palmer's resource, called Proposition Bank (or PropBank), contains more than 112,000 semantic role analyses for 3,256 distinct verbs. Instead of committing to one of the many competing theories of semantic

[5] A prefixed asterisk denotes an unacceptable sentence of English.

roles, the creators of PropBank chose a theory-agnostic approach in which each sense of each verb is associated with its own set of roles. Each role set for a verb is contained in a frame file for the verb. To demonstrate, consider the frame for the verbal predicate *distribute* from the PropBank lexicon:

> Frame for *distribute*, role set 1:
> $Arg_0$: the entity that is performing the distribution
> $Arg_1$: the entity that is distributed
> $Arg_2$: the entity to which the distribution is made

Next, consider an instance of *distribute* taken from the PropBank corpus:

(2.13) Freeport-McMoRan Energy Partners will be liquidated and [$Arg_1$ shares of the new company] [*Predicate* distributed] [$Arg_2$ to the partnership's unitholders].

In Example 2.13, the *Theme* and *Destination* from Gruber (1965) have been given the labels $Arg_1$ and $Arg_2$, respectively. The interpretation of these roles is defined in the role set shown above. Because the interpretation of arguments in PropBank is verb- and sense-specific, there is no guarantee that $Arg_1$ and $Arg_2$ will denote the same semantic properties for other verbs in the lexicon. However, Kingsbury and Palmer (2003) note that, across verbs, $Arg_0$ and $Arg_1$ are very often interpretable as *Agent* and *Theme*, respectively. PropBank's theory-agnosticism regarding semantic roles means that it is compatible with many different theories. For example, subsequent studies have demonstrated the feasibility of mapping PropBank argument positions into more traditional theories of semantic roles (see, for example, the PropBank-VerbNet role mapping developed by Yi et al. (2007)).

### NomBank

Unlike FrameNet, which focuses primarily on verbal argument structure, and PropBank, which focuses solely on verbal argument structure, the NomBank corpus (Meyers, 2007a) focuses solely on the argument structure of nominals. NomBank inherited the lexicon design and annotation methodology used for the PropBank project. Thus, each nominal predicate is

associated with a frame file that lists role sets and argument definitions similar to those given above for the verb *distribute*. Consider the following instance of the nominal *distribution*, taken from the NomBank corpus:

(2.14) Searle will give [$Arg_0$ pharmacists] [$Arg_1$ brochures] [$Arg_1$ on the use of prescription drugs] for [*Predicate* distribution] [*Location* in their stores].

When possible, the creators of NomBank adapted PropBank frame files for verb-based nominal predicates such as *distribution*. Thus, in Example 2.14, argument positions $Arg_0$ and $Arg_1$ have the same semantic interpretation as argument positions $Arg_0$ and $Arg_1$ for the PropBank verb *distribute*.

The compatibility between NomBank and PropBank is important because verbal and nominal predicates often interact with each other. Consider the following contrived example:

(2.15) [$Arg_0$ John] failed to make the [$Arg_1$ newspaper] [*Predicate* delivery].

To arrive at the labeling of the nominal predicate *delivery* in Example 2.15, the reader relies on his or her knowledge of how *fail* (verb), *make* (verb), and *delivery* (noun) interact in the given context. This interaction is the key to understanding many event descriptions and highlights the importance of the PropBank/NomBank compatibility - the two resources can be seamlessly integrated.

Returning to Example 2.14, notice that two spans of text are bracketed with the $Arg_1$ label. This is an instance of split argumentation, which is also present in PropBank. A split argument is a span of text that constitutes an argument but cannot be precisely subsumed by a single syntactic parse tree node within the Penn TreeBank. Split argumentation is typically caused by syntactic analyses that are not binary branching. The syntactic parse for Example 2.14 is shown in Figure 2.2. As shown, it is impossible to select a single node that subsumes only the complete $Arg_1$ in Example 2.14. Thus, the creators of NomBank and PropBank have elected to mark both the NP and the PP that together give the correct subsumption (i.e., *brochures on the use of prescription drugs*). I will return to split arguments in Section 2.3.3, where argument prediction conflicts are discussed.

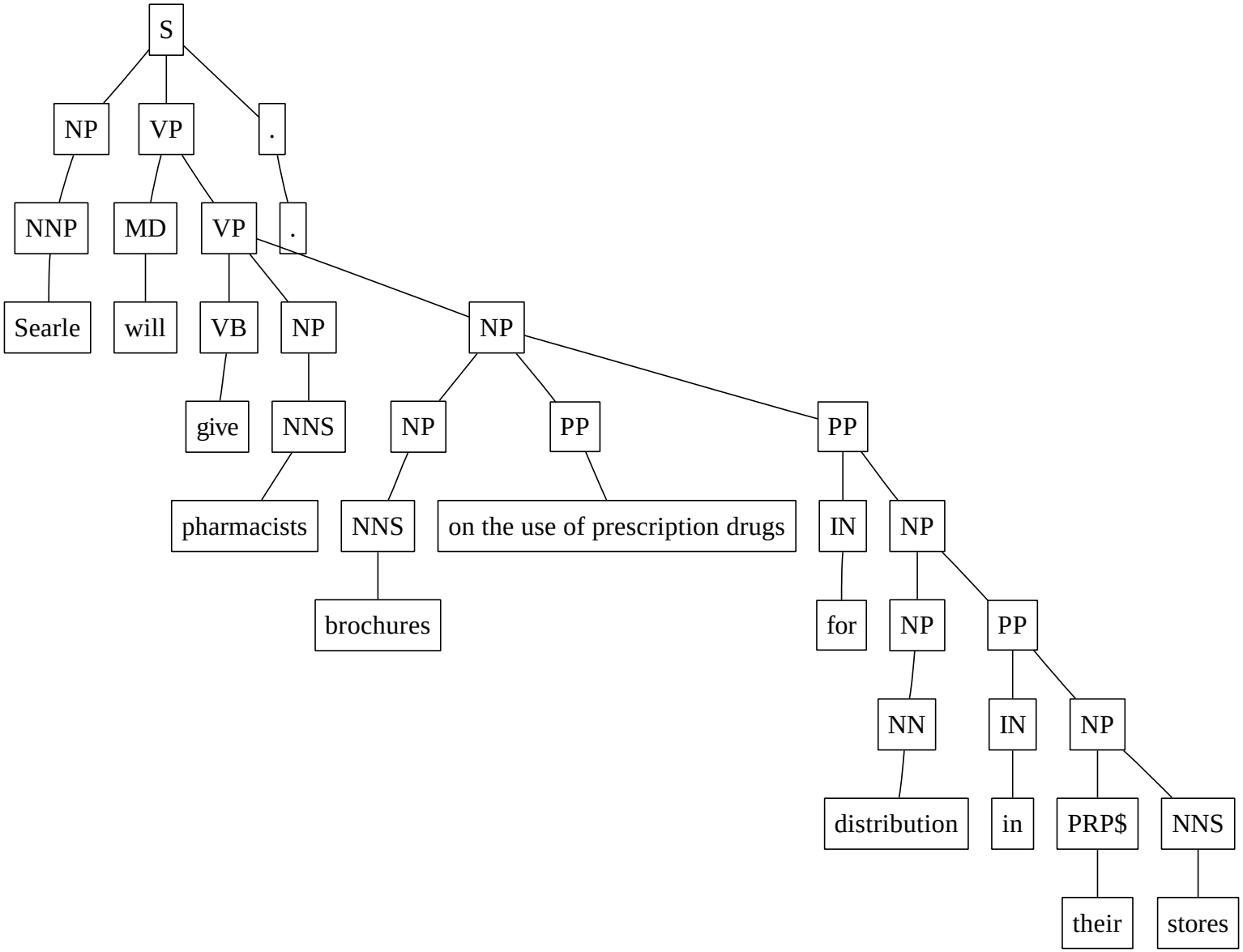



Figure 2.2: Syntax of the split argument construction in Example 2.14.

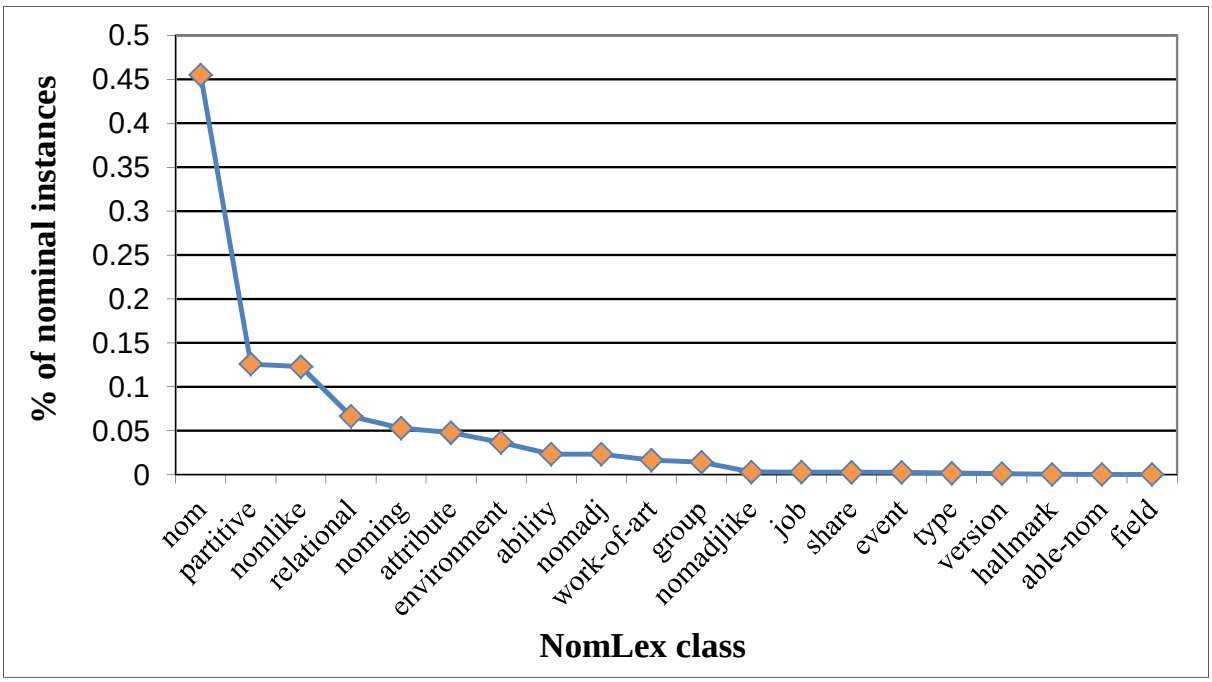



Figure 2.3: Distribution of nominal instances across the NomLex classes. The $y$-axis denotes the percentage of all nominal instances that is occupied by nominals in the class.

Example 2.14 also demonstrates the annotation of the *Location* argument, which is one of many adjunct argument types that are annotated by both PropBank and NomBank (other adjuncts include *Manner*, *Temporal*, *Purpose*, *Direction*, and *Location*, among others). The interpretation of an adjunct argument is the same across all predicates in PropBank and NomBank. For example, the *Location* argument has the same interpretation regardless of whether it is connected to the verbal predicate *send* or the nominal predicate *flight*. Because their interpretations are not predicate-specific, adjunct arguments are not included in the frame files. Instead, they are assumed to be available for all predicates.

Examples 2.14 and 2.15 involve nominal predicates that are derived from verbs. This dissertation will refer to such predicates as deverbal or event-based nominal predicates. In addition to these predicates, NomBank annotates a wide variety of nouns that are not derived from verbs and do not denote events. An example is given below of the partitive noun *percent*:

| Argument | Count | % | Argument | Count | % |
|---|---|---|---|---|---|
| $Arg_1$ | 80,102 | 40.4 | *Location* | 5,771 | 2.9 |
| $Arg_0$ | 49,823 | 25.1 | *Extent* | 865 | 0.4 |
| $Arg_2$ | 34,850 | 17.6 | *Negation* | 655 | 0.3 |
| *Temporal* | 9,495 | 4.8 | *Adverbial* | 591 | 0.3 |
| $Arg_3$ | 7,611 | 3.8 | $Arg_4$ | 494 | 0.2 |
| *Manner* | 7,210 | 3.6 | *Purpose* | 444 | 0.2 |

Table 2.1: Distribution of annotated NomBank arguments. Argument positions with fewer than 100 occurrences are omitted.

(2.16) Hallwood owns about 11 [*Predicate* %] [$Arg_1$ of Integra].

In this case, the noun phrase headed by the predicate % (i.e., *about 11% of Integra*) denotes a fractional part of the argument in position $Arg_1$. Other partitive predicates behave similarly. The NomLex resource (Macleod et al., 1998) is a hand-coded lexicon that classifies the various nominal types annotated by NomBank (e.g., deverbal, partitive, and others). Figure 2.3 shows the distribution of NomBank predicate instances across the NomLex classes. Deverbal (i.e., event-denoting) nominals reside in the *nom* class, which is significantly larger than any other class. This is the expected result because events form the foundation of many textual discourses.

In total, the NomBank corpus contains argument information for 114,574 instances of 4,704 distinct nominal predicates. Because this dissertation focuses on the automatic identification of the various argument types, it is important to understand the corresponding distribution. Table 2.1 presents this information. As shown in the table, the distribution of arguments is extremely skewed; $Arg_0$, $Arg_1$, and $Arg_2$ account for approximately 83% of the annotated argument structure. Thus, in order for a system to perform well it must target these argument types.

I conclude this section by showing that NomBank contains a significant amount of semantic information that is not present in PropBank and cannot be recovered using verbal SRL. First, note that PropBank contains approximately 49 predicates per document, whereas

NomBank contains approximately 50 predicates per document. Thus, NomBank predicates are just as frequent as PropBank predicates; however, this alone is not enough to show that NomBank contains novel information beyond that given by PropBank. Consider the situation in which an instance of the verb *distribute* is followed by an instance of the noun *distribution*. It is quite likely that these two predicate instances refer to the same event. Thus, extracting information from the latter might not enhance one's understanding of the document. Analysis shows that this behavior is more the exception than the rule: 87% of NomBank predicate instances are neither preceded nor followed by corresponding PropBank predicates in the same documents. This fact, combined with the per-document frequency of nominal predicates mentioned above, is preliminary evidence that nominal predicate-argument structure contributes a significant amount of information to the discourse. This information should complement PropBank information, which, as described earlier, has been useful in tasks such as QA, IE, and SMT.

### 2.2.3 Statistical SRL

The creation of FrameNet prompted a move from hand-coded semantic processing to statistical learning-based approaches. The seminal work of Gildea and Jurafsky (2002) treated the SRL problem as a supervised learning task and used the FrameNet corpus as a source of training data. Gildea and Jurafsky employed simple maximum likelihood statistics for various lexical and syntactic features to both identify frame element boundaries within text and assign semantic role labels (e.g., *Agent*) to the identified frame elements. Results of this study were promising: the authors reported an overall role $F_1$ score[6] of approximately 63% on the task of combined frame element identification and labeling. Gildea and Jurafsky obtained this result using (among other things) features extracted from automatically generated syntactic parse trees. Two results from this work have been particularly influential

[6]In this dissertation, $F_1$ refers to the harmonic mean of precision and recall: $\frac{2*Precision*Recall}{Precision+Recall}$, where $Precision = \frac{\text{\# true positives}}{\text{\# predicted positives}}$ and $Recall = \frac{\text{\# true positives}}{\text{\# existing positives}}$.

on subsequent work:

1. **Syntactic information is essential for high-quality SRL.** Many linguistic theories posit a strong connection between syntax and semantics. For example, Adger (2003) develops a framework in which all semantic roles for a predicate are assigned to syntactic constituents (p. 81). Furthermore, each predicate places syntactic (p. 84) and semantic (p. 87) restrictions on the semantic roles with which it can be associated. Applied work in SRL has found the syntactic restrictions to be particularly important (Gildea and Palmer, 2001; Punyakanok et al., 2008), and this chapter will place continued emphasis on syntactic information when identifying semantic arguments within a sentence.

2. **A two-stage configuration is possible.** Gildea and Jurafsky (2002) used separate classifiers to identify frame elements and apply labels to them. Numerous subsequent studies have followed this tradition; however, no compelling arguments have been offered in support of this configuration. This dissertation develops a single-stage model in which arguments are predicted directly, thus avoiding the complexities of chaining multiple classifiers together.

It is important to briefly mention the evaluation setup used by Gildea and Jurafsky (2002). The authors evaluated their system using ground-truth predicates and frames. The system's only task was to identify and label the frame elements. Thus, although the work was promising it left many open questions. One important question was how to extend the model to a more practical scenario in which a system is given raw text and must handle all processing tasks using no ground-truth information. The current chapter also assumes ground-truth predicates. Chapter 3 explores automatic predicate identification in detail.

Soon after its release, PropBank became a popular resource for statistical SRL researchers, supporting many studies and motivating a number of large-scale, competitive evaluation tasks (e.g., the CoNLL Shared Tasks described by Carreras and Màrquez (2004),

Carreras and Màrquez (2005), and Surdeanu et al. (2008)). Below, I describe three key aspects of PropBank-based SRL research.

### Syntactic representation

Syntactic information is essential for the SRL task; however, there are different ways to represent this syntactic information. Figure 2.2 (p. 19) demonstrates the constituency approach to syntax, which uses a context free grammar formalism. This formalism has a long history in linguistics and is amenable to processing by algorithms such as the popular Cocke-Younger-Kasami (CYK) algorithm, whose running time is $O(n^3)$ in the sentence length. The competition organized by Carreras and Màrquez (2005) used this syntactic representation. More recently, Surdeanu et al. (2008) organized a competition in which the dependency approach to syntax was explored. Although this formalism is not as rich as constituency representations (i.e., some syntactic properties cannot be captured), it has the advantage of processing algorithms with running times that are $O(n)$ in the sentence length (Nivre, 2003). This dissertation will use the constituency formalism in order to explore some of the deeper syntactic properties of sentences.

### Machine learning technique

As described by Carreras and Màrquez (2005) and Surdeanu et al. (2008), a majority of the most successful SRL systems have used maximum entropy models (Berger et al., 1996) or support vector machines (Burges, 1998). These techniques accommodate large-scale datasets and often learn models that generalize well from training to testing. Most of the models developed in this dissertation are produced by the logistic regression framework (LibLinear) created by Fan et al. (2008), which is capable of handling millions of training instances and features. As noted by Hsu et al. (2010), high-dimensional data does not always benefit from a mapping into a higher-dimensional space, as is often done with SVMs. For nominal SRL, I have found that the linear models produced by LibLinear perform as well as SVMs but are

significantly faster to train.

### Joint inference

Many SRL systems have incorrectly assumed that the existence of one argument is independent of the existence of other arguments. Consider the following examples, created by Toutanova et al. (2008):

(2.17) [*Temporal* The day] that [*Agent* the ogre] [*Predicate* cooked] [*Theme* the children] is still remembered.

(2.18) [*Theme* The meal] that [*Agent* the ogre] [*Predicate* cooked] [*Beneficiary* the children] is still remembered.

Only one word differs between these examples (*day* is replaced with *meal*); however, the interpretations are vastly different. In 2.17 the children are cooked, whereas in 2.18 the meal is cooked. If the initial noun phrase is changed from a *Temporal* marker to a *Theme*, the roles of other constituents are also changed. This dependence between roles prompted Toutanova et al. to study joint inference across argument assignment possibilities. Similarly, Punyakanok et al. (2008) used integer linear programming to enforce constraints on joint SRL structures for verbal SRL (e.g., one constraint is that arguments cannot overlap each other in the sentence). This dissertation explores a joint inference model for nominal SRL in Chapter 5.

The PropBank-based SRL systems mentioned above often reach argument $F_1$ scores approaching 80% when tested on PropBank data. However, these systems tend to encounter difficulties when tested on genres of text that differ from the training corpus. Carreras and Màrquez (2005) cite a performance drop of around 10 $F_1$ points for all participating systems when evaluated over PropBank annotations from the Brown Corpus of Present-day American English (Kučera and Nelson, 1967). The Brown Corpus comprises approximately one million words from a variety of sources. Pradhan et al. (2008) provide an in-depth study of the effects of text genre on verbal SRL, concluding that the second stage (argument label

assignment) contributes the most toward out-of-domain performance degradation. This is due, in large part, to a reliance on lexical and semantic features tuned specifically for the TreeBank corpus. A similar drop in performance can be expected for the model developed in the current chapter.

### Nominal SRL

Statistics-based work on nominal SRL has lagged behind its verbal counterpart by a few years. This is probably because verbs are usually the first choice when analyzing textual semantics. However, as pointed out above, nominal predicates carry a significant amount of novel information. When it comes to automatically extracting this information, one finds a few precursors to the standard nominal SRL task. For example, Lapata (2000) developed a statistical model to classify modifiers of deverbal nouns as underlying subjects or underlying objects, where subject and object denote the grammatical position of the modifier when linked to a verb. Consider two possible interpretations of the phrase "satellite observation" below:

(2.19) [*Subject* Satellite] [*Predicate* observation] techniques are used to keep track of enemy troop movements.

(2.20) The stargazers routinely engaged in [*Object* satellite] [*Predicate* observation].

In Example 2.19, it is the satellites that are being used for observation, whereas in Example 2.20 the satellites are being observed. Lapata developed a simple statistical model to identify this distinction, which corresponds roughly to the distinction between $Arg_0$ (*subject*/*Agent*) and $Arg_1$ (*object*/*Theme*) in NomBank. The study did not account for other argument positions, including adjunct arguments.

In a general sense, nominal SRL is the process of identifying relations between a noun (the predicate) and other nouns in the surrounding context. In recent years, researchers have investigated a variety of noun-noun relations. Girju et al. (2007) organized a SemEval[7] com-

[7] http://www.senseval.org

petition in which systems identified noun-noun relations such as *Cause-Effect*, an example of which is given below:

(2.21) The individual was infected with the [*Effect* flu] [*Cause* virus].

This relation is not analogous to any of the semantic role relations discussed so far. However, the *Instrument-Agency* SemEval relation is:

(2.22) The [*Instrument* phone] [*Agency* operator] answered my call.

Girju et al.'s task defined five other relations and required systems to make binary decisions about whether a segment of text contained a particular relation (the relation type was given to the system at testing time). This task was later refined by Hendrickx et al. (2010), resulting in a multi-way evaluation where each test example could exhibit any of the relations. These two tasks are certainly related to nominal SRL, but most of the relations they focus on do not have an interpretation in terms of semantic roles. Thus, the work presented in this dissertation is largely complimentary to the SemEval tasks.

Although FrameNet contains some annotations for nominal predicates, NomBank (Meyers, 2007a) has been the driving force behind true nominal SRL in recent years. Based on a pre-release version of NomBank, Jiang and Ng (2006) used standard verbal SRL techniques and achieved an overall argument $F_1$ score of 69.14% using automatically generated syntactic parse trees. Liu and Ng (2007) followed this up with a different technique (alternating structure optimization) and achieved an $F_1$ score of 72.83%; however, the latter study used an improved version of NomBank, rendering these two results incomparable. Both studies also investigated the use of features specific to the task of NomBank SRL, but observed only marginal performance gains.

Following these initial studies, NomBank supported a series of competitive evaluation tasks hosted by the Computational Natural Language Learning (CoNLL) conference. The first task, Joint Parsing of Syntactic and Semantic Dependencies (Surdeanu et al., 2008), required systems to identify the dependency syntax for a sentence as well the sentence's

predicate-argument structure for both verbal and nominal predicates. Verbal predicate-argument structure was derived from PropBank whereas nominal predicate-argument structure came from NomBank.

A majority of systems in the 2008 CoNLL competition formulated the SRL problem as a two-stage classification problem. In the first stage, spans of text were assigned a binary label indicating whether or not the span represented an argument. In the second stage, argument spans were relabeled with a final label, which was then evaluated. For nominals, the best overall $F_1$ score was 76.64%; however this score is not directly comparable to the NomBank SRL results of Jiang and Ng (2006), Liu and Ng (2007), or the results in this dissertation because the evaluation metrics are not the same (see Section 2.4 for details). A similar task was run in 2009 by Hajič et al., the only fundamental difference being the inclusion of additional languages. This dissertation only investigates nominal SRL for English text.

In the remainder of this chapter, I present a statistical NomBank SRL system that will be a starting point for the chapters that follow. In Section 2.3, I describe the SRL model in terms of its formulation, features, and general operation. I then present a formal evaluation of the model in Section 2.4. Sections 2.5 and 2.6 identify a variety of problems that will be taken up in subsequent chapters.

## 2.3 Nominal SRL model

### 2.3.1 Model formulation

The following example demonstrates the testing input to the nominal SRL model:

(2.23) Searle will give pharmacists brochures on the use of prescription drugs for [*Predicate* distribution] in their stores.

As shown, the system is given a sequence of words and the nominal predicate. Using this information, the model must assign semantic labels (e.g., $Arg_0$, $Arg_1$, ..., *Location*, etc.) to spans of text in the sentence. The correct labeling is given in Example 2.14 (p. 18).

The nominal SRL task is treated as a multi-class classification problem over parse tree nodes. Each parse tree node subsumes an unambiguous span of text. Thus, classifying a node is equivalent to labeling a span of text in the sentence (see Figure 2.2 on page 19). All nodes are classified except those that overlap the predicate. In total, there are 22 classes representing the $Arg_n$ and adjunct arguments. One additional class *null* is added to account for parse tree nodes whose text does not fill a semantic role. For a classifiable node $n$, the 23 classes are modeled in a single stage as follows:

$$\underset{l \in Labels}{\arg\max} \, Pr(Label(n) = l | f_1, \ldots, f_n) \tag{2.24}$$

Equation 2.24 constitutes a departure from the two-stage tradition in SRL; however, I have found that this single-stage approach tends to outperform the two-stage approach described previously. I used the multi-class logistic regression solver provided by LibLinear (Fan et al., 2008) to estimate Equation 2.24. In the following section, I give a precise specification for features $f_1, \ldots, f_n$, which are used as evidence for the prediction.

### 2.3.2 Model features

Starting with a wide range of features, I used a greedy selection algorithm similar to the one proposed by Pudil et al. (1994) to identify an optimal subset.[8] Table A.3 in the Appendix (p. 136) lists the selected argument features. Below, I give detailed examples for features that are not sufficiently explained in the table.

**Feature 4** identifies predicate-specific argument behavior. Consider the following examples from the Penn TreeBank:

(2.25) [$Arg_1$ Investment] [*Predicate* analysts] generally agree.

(2.26) The tender [*Predicate* offer] [$Arg_1$ for Gen-Probe's shares] is expected to begin next Monday.

[8] See Section A.8 on page 144 for a listing of the feature selection algorithm.

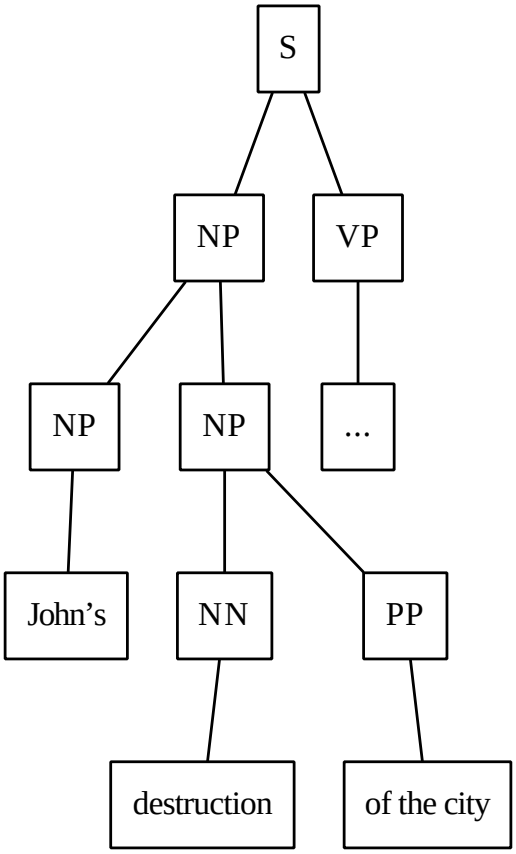



Figure 2.4: Syntactic context of the *destruction* predicate.

In Example 2.25 the $Arg_1$ (entity analyzed) precedes the predicate. 95% of all *analyst* instances behave the same way. Compare this to Example 2.26, where the $Arg_1$ (entity acquired) follows the predicate. Ninety percent of all *offer* instances behave similarly. As these examples show, an argument's location relative to the predicate can depend heavily on the predicate itself. Thus, the value of Feature 4 is obtained by concatenating the predicate stem with a binary value indicating whether the candidate argument $n$ comes before or after the predicate in the sentence. This feature would have a value of *analyst:before* in Example 2.25 and *offer:after* in Example 2.26. Many other features in Table A.3 have predicate-specific values for similar reasons.

**Feature 10** captures the basic syntactic structure that surrounds the predicate. Some approaches to SRL begin with a set of pruning heuristics to eliminate unlikely candidate arguments (Xue and Palmer, 2004). These heuristics start at the predicate node and inspect the local syntactic context for particular constituents. For example, the predicate's sibling node is included in the candidate pool if it is a prepositional phrase. This situation is shown in Figure 2.4 for the *destruction* predicate. Instead of using heuristics, Feature 10 directly encodes the syntactic context of a predicate. The value for this feature is the context-free grammar rule that expands the predicate's parent node. With respect to Figure 2.4, this grammar rule would be $NP \rightarrow NN,NP$.

**Feature 26** captures the syntactic relationship between the candidate argument node and the predicate. Its value is formed by traversing the parse tree from the candidate to the predicate node. At each step in the traversal, the current syntactic category and direction of movement (up or down) is recorded. In Figure 2.4, the parse tree path from the candidate argument *of the city* to the predicate node *destruction* would be $PP \uparrow NP \downarrow NN$. Since its introduction by Gildea and Jurafsky (2002), this feature has proved to be one of the most informative for the SRL task. In my nominal SRL model this feature ranks quite low because variations of it are yet more informative. For example, Features 1 and 2 make the standard path more specific by combining it with other information. Feature 13 makes the standard path more general by removing information. The feature selection algorithm determined that these variations were better suited to the nominal SRL task.

**Feature 17** considers the parse tree path between the candidate argument node and so-called support verbs in the sentence. Support verbs (also called light verbs) have very little semantic meaning. Their primary purpose is to link long-distance arguments to nominal predicates that are more meaningful. Consider the following contrived example:

(2.27) [$Arg_0$ John] [*Support* took] a [*Predicate* walk].

In Example 2.27, *took* does not have the usual meaning of forcibly changing possession; rather, this verb's purpose is to bring in *John* as the $Arg_0$ (walker) of *walk*. This sentence can be paraphrased with the verb *walk* as "John walked.".

Feature 17 identifies the parse tree path between the candidate argument and the nearest support verb. As shown above (see Example 2.23), the system is not given support verb information at testing time. Thus, I created a model to automatically identify support verbs so that they may be used by this feature. The model and features used to identify support verbs are described in Appendix Section A.1 (p. 134).

Before moving on, it is worth noting that there are alternatives to the extensive feature engineering and selection process described above. Moschitti et al. (2008) present a detailed analysis of so-called tree kernels and their application to various NLP problems, SRL being their primary interest. Tree kernels provide a means for feature engineering based on the "kernel trick" that is available in learning frameworks such as support vector machines. This dissertation leaves the exploration of tree kernels to future work.

**Feature binarization**

Like many other machine learning toolkits, LibLinear's instance representation format requires features with numeric values. As shown above, the value range for many features has no meaningful numeric ordering. That is, a value of $PP \uparrow NP \downarrow NN$ for Feature 26 cannot be meaningfully compared to other values for this feature (e.g., the $NP \uparrow NP \downarrow NP \downarrow NN$ path from *John* to *destruction* in Figure 2.4). Thus, it would be unwise to create a single numeric feature *Parse path* by mapping $PP \uparrow NP \downarrow NN$ to 1 and $NP \uparrow NP \downarrow NP \downarrow NN$ to 2. Instead, as suggested by Hsu et al. (2010), all non-numeric features are binarized. Assume that each candidate node $n$ is represented using only Feature 26 (the parse tree path). Also assume that this feature has two possible values (the paths mentioned above). In LibLinear, each node $n$ would be represented as one of the following:

$n$ has path $PP \uparrow NP \downarrow NN$: $\langle 1, 0 \rangle$
$n$ has path $NP \uparrow NP \downarrow NP \downarrow NN$: $\langle 0, 1 \rangle$
$n$ has neither path: $\langle 0, 0 \rangle$

Thus, for a feature with $m$ possible values, binarization creates $m$ mutually exclusive binary features. Instances are represented by activating at most one of these features.

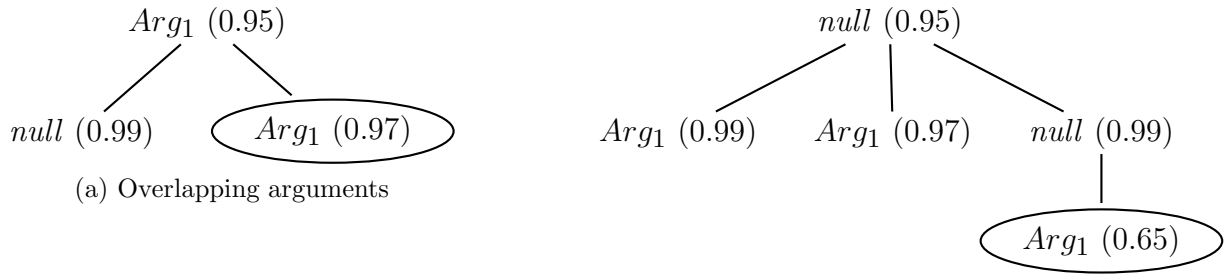
The binarized feature space can be extremely large. A single word-based feature can easily binarize to $10^5$ binary features. This poses a problem for the greedy forward search algorithm described on page 144, which inspects each individual feature. Whether this is actually a problem depends on how one defines a feature. If one defines a feature to be the unbinarized version, then the parse tree path represents a single feature that can be selected. If one defines a feature to be the binarized version, then the parse tree path represents thousands of features to be selected from. I assumed the former definition when performing feature selection. For example, by including or excluding Feature 26, the selection process implicitly includes or excludes all resulting binarizations of this feature. This compromise keeps the selection process tractable.

### 2.3.3 Post-processing

After classifying all candidate nodes in the tree using the model described above, two steps still remain. First, a special classification must be performed on the predicate node itself. Following this, the entire assignment must be made consistent. These two steps are described below.

#### Incorporated arguments

An important difference between PropBank and NomBank is that the latter often applies argument labels to predicate nodes themselves, whereas the former does not. In NomBank, predicates that are also arguments are referred to as incorporated arguments. An example is given below:




(a) Overlapping arguments

(b) Duplicate arguments

Figure 2.5: Global constraint violations. The circled node in 2.5a is reassigned to the *null* class after its score is averaged into its parent node. The circled node in 2.5b is reassigned to the *null* class because it has lower confidence than other nodes of the same type. The remaining $Arg_1$ nodes in 2.5b are kept because they are siblings. This accounts for split arguments (see Section 2.2.2 for a discussion of split argument constructions).

(2.28) Petrolane is the second-largest [$Arg_1$ propane] [*Predicate*/$Arg_0$ distributor] [*Location* in the U.S.].

In 2.28, the predicate additionally assumes the $Arg_0$ role (the entity performing the distribution). In order to account for incorporated arguments, the system uses a separate model to assign argument labels to predicate nodes. For the predicate *distributor*, the model assigns the label that maximizes the following probability:

$$P(Arg_i|distributor) = \frac{\#(Arg_i, distributor)}{\#(distributor)} \tag{2.29}$$

In Equation 2.29, $\#(Arg_i, distributor)$ is the number of times that the *distributor* predicate is observed with the incorporated argument label $Arg_i$ in the training data. $\#(distributor)$ is the total number of occurrences of the *distributor* predicate in the training data. This simple method labels incorporated arguments with an $F_1$ score of approximately 87%.

### Conflict resolution

When labeling a particular node, the feature-based logistic regression model does not take labels for other nodes into account. Neither does the model use dynamic programming to

arrive at the most likely consistent assignment of labels to constituents, as done by Jiang and Ng (2006) and Toutanova et al. (2005). As a result, argument labels sometimes violate global labeling constraints, which are illustrated in Figure 2.5. These constraints are enforced using the following heuristics:

**Overlapping argument heuristic** Overlapping arguments arise when two nodes are labeled as arguments and one node is an ancestor of the other, as shown in Figure 2.5a. If each node has the same label, the system re-scores the ancestor node with the average of the two nodes' confidence scores. The descendant node is then reassigned to the *null* class. If the two nodes have different labels, the node with higher confidence is kept and the other is reassigned to the *null* class. All reassignments to the *null* class are made with confidence equal to 1.0.

**Duplicate argument heuristic** Duplicate arguments arise when two nodes are assigned the same argument label and one is not an ancestor of the other, as shown in Figure 2.5b. If the two nodes are not siblings, the node with the higher confidence score is kept and the other is reassigned to the *null* class. If the two nodes are siblings, both are kept. Keeping both sibling nodes accounts for split argument constructions, which were discussed in Section 2.2.2 (p. 13).

**Low confidence heuristic** After the previous two heuristics are applied, all argument nodes with confidence less than a threshold $t_{arg}$ are removed. The value for $t_{arg}$ is found by maximizing the system's performance on a development dataset.

To summarize, when given a sentence and a nominal predicate within the sentence, the logistic regression model is applied to each node in the parse tree that does not overlap with the predicate node. The predicate node is then labeled, and the heuristics are applied to resolve argument conflicts and remove argument labels with low confidence scores.

| | Development $F_1$ | Testing $F_1$ |
|---|---|---|
| Jiang and Ng (2006) | 0.6677 | 0.6914 |
| Liu and Ng (2007) | (not reported) | 0.7283 |
| This dissertation | 0.7401 | **0.7574** |

Table 2.2: NomBank SRL results for argument prediction using automatically generated parse trees. The $F_1$ scores were calculated by aggregating predictions across all classes.

## 2.4 Evaluation

To test the model described in the previous section, I extracted training nodes from sections 2-21 of NomBank, keeping only those nodes that did not overlap with the predicate. LibLinear parameters were set as follows: $bias = 1$, $c = 1$, $w+ = 1$. I tuned the $t_{arg}$ threshold using section 24 as development data ($t_{arg} = 0.42$). Finally, I used section 23 for testing.[9] All syntactic parse trees were generated by the August 2006 version of Charniak's re-ranking syntactic parser (Charniak and Johnson, 2005). Each annotated predicate in the testing section was presented to the system as shown in Example 2.23 (p. 28).

Table 2.2 presents the evaluation results. I calculated the $F_1$ scores by aggregating predictions across all predicates. Precision and recall were defined in the usual way:

$$Precision = \frac{\#(\text{correct labels applied})}{\#(\text{labels applied})} \tag{2.30}$$

$$Recall = \frac{\#(\text{correct labels applied})}{\#(\text{labels in ground-truth})} \tag{2.31}$$

This evaluation methodology follows the one used by Jiang and Ng (2006) and Liu and Ng (2007); however, the results for my model are only comparable to the latter because the former used a preliminary release of NomBank.[10]

[9]This data separation is standard for PropBank/NomBank SRL evaluations. See, for example, Carreras and Màrquez (2005).

[10]The discrepancy between the development and testing results is likely due to poorer syntactic parsing performance on the development section (Carreras and Màrquez, 2005).

## 2.5 Discussion

As can be seen, the NomBank SRL system presented in this chapter comfortably outperforms the best previous result. Because the models share many properties, it is worth discussing factors that could possibly lead to the performance difference. First, I observed a significant performance increase when moving from a traditional two-stage pipeline to the single-stage classifier presented above. To my knowledge, the research community has not thoroughly investigated the need for a two-stage pipeline. It is, however, the computationally easier route. A two-stage approach requires a binary first-stage classifier trained over approximately 3.7 million nodes and a 22-class second-stage classifier trained over approximately 179,000 nodes. A single-stage nominal SRL classifier, on the other hand, requires a 23-class classifier trained over approximately 3.7 million nodes. In the one-versus-all approach to multi-class classification, the single-stage SRL classifier is much more computationally intensive. However, the single-stage approach is free of cascading errors, which are common in pipelined architectures such as the two-stage model. In the two-stage model, a false negative error in the first stage prevents the second stage from making a decision.

Another important difference between the current model and the other two is the treatment of overlapping argument nodes and incorporated arguments. In the work of Jiang and Ng (2006), incorporated arguments were not included in the training data despite the fact that they occur very frequently - approximately 15% of arguments in the training data are incorporated. The authors do attempt to label predicate nodes at evaluation time using the trained model, but the most important features for argument labeling (e.g., the parse tree path) are not informative for such nodes. In contrast, Liu and Ng (2007) included all parse tree nodes in the training data, even those that overlap with the predicate node (presumably, this includes the predicate node itself). At evaluation time, all nodes are classified; however, considering the fact that 0.02% of non-incorporated arguments overlap the predicate node, this approach is likely to create more errors than correct labels. The model presented in Sec-

tion 2.3 takes a hybrid approach. Nodes that overlap the predicate are not used as training data, nor are they labeled by the logistic regression model during testing; instead, predicate nodes are labeled by the simple model described in Section 2.3, which achieves an $F_1$ score of 0.84 on incorporated arguments of all types.

## 2.6 Conclusions

This chapter has shown that the tremendous syntactic flexibility of natural language can be semantically normalized by analysis in terms of semantic roles. This analysis does not aim to produce a deep, complete semantic interpretation; rather, the aim is to extract shallow information reliably. This chapter has also shown that, despite its shallow nature, semantic role analysis produces a significant amount of information that can be levered for language processing tasks that require inference.

The nominal SRL system described in this chapter produces state-of-the-art results using no manual intervention. It relies primarily on a rich syntactic analysis combined with traditional supervised machine learning. The single-stage architecture is computationally more expensive than the standard two-stage model; however, it does not require one to chain multiple components together. By carefully handling nominal-specific issues like argument incorporation, the system is able to recover arguments with an $F_1$ score of approximately 76%. This is an encouraging result; however, the system makes two important assumptions that must be addressed:

1. **Ground-truth predicates** are provided to the system at testing time. This does not invalidate the evaluation methodology used in this chapter, which still provides useful information about the SRL model; however, in order to assess true end-to-end performance in a practical setting, one must remove this assumption and force the system to identify predicates as well as arguments. The following chapter does exactly this.

2. **Extra-sentential arguments** are similar to the arguments described in this chapter. The only difference is that extra-sentential arguments are not present in the sentence that contains the predicate. Rather, these arguments exist in some other sentence of the document. Chapter 4 will explore the nature and extraction of extra-sentential arguments, which have received relatively little attention from the research community.

# CHAPTER 3

# Predicates that lack arguments: the problem of implicit argumentation

## 3.1 Introduction

The previous chapter presented a state-of-the-art nominal SRL system that will serve as a baseline for the current chapter. The system achieves an overall argument $F_1$ score of approximately 76% using a supervised learning approach combined with carefully constructed post-processing heuristics. Although this result is encouraging, it is produced by an evaluation methodology that has specific limitations. In particular, the evaluation (which has been used in many previous NomBank and PropBank SRL studies) provides a system with a predicate that is known to take arguments in the local context. However, nominal predicates often surface without local arguments. Consider the following instances of *distribution* from the Penn TreeBank:

(3.1) Searle will give [$Arg_0$ pharmacists] [$Arg_1$ brochures] [$Arg_1$ on the use of prescription drugs] for [*Predicate* distribution] [*Location* in their stores].

(3.2) The [*Predicate* distribution] represents [*NP* available cash flow] [*PP* from the partnership] [*PP* between Aug. 1 and Oct. 31].

In Example 3.1, *distribution* is associated with arguments annotated by NomBank. In contrast, *distribution* in 3.2 has a noun phrase and multiple prepositional phrases in its environment (similarly to 3.1), but not one of these constituents is an argument to the marked predicate. As described by Meyers (2007a), predicate instances such as 3.1 are called "markable" because they are associated with local arguments. Predicate instances such as 3.2 are called "unmarkable" because they are not associated with local arguments. In the NomBank

corpus, only markable predicate instances from the Penn TreeBank have been annotated. All other predicates have been ignored.

A number of evaluations (e.g., those described by Jiang and Ng (2006), Liu and Ng (2007), and Chapter 2 of this dissertation) have been based solely on markable predicate instances (i.e., those annotated by NomBank). This group constitutes only 57% of all nominal predicate instances found in the underlying TreeBank corpus. In order to use the output of nominal SRL systems as input for other systems (e.g., QA, IE, and SMT), it is important to develop and evaluate techniques that can handle all predicate instances instead of a select few. With respect to the evaluation procedure of the previous chapter, this amounts to eliminating the assumption that a test predicate takes arguments; instead, the SRL system must make this decision automatically for every token in the corpus.

Underlying the issues described above is a phenomenon called implicit argumentation. An implicit argument is any argument that is not annotated by NomBank. Thus, a predicate is unmarkable when all of its arguments are implicit. In this chapter, I investigate the role of implicit argumentation in nominal SRL. This is, in part, inspired by the 2008 CoNLL Shared Task (Surdeanu et al., 2008), which was the first evaluation of syntactic and semantic dependency parsing to include unmarkable nominal predicates. The current chapter extends this task to constituent parsing with techniques, evaluations, and analyses that focus specifically on implicit argumentation for nominal predicates. In the next section, I assess the prevalence of implicit argumentation and its impact on the nominal SRL system presented in Chapter 2. I find that, when applied to all nominal instances, this system achieves an argument $F_1$ score of only 69%, a loss of approximately 8%. In Section 3.3, I review the recent CoNLL Shared Task, noting similarities and differences with the current work. In Section 3.4, I present a model designed to filter out nominal predicates whose arguments are entirely implicit. This model reduces the aforementioned loss, particularly for nominals that are not often markable. In the analyses of Section 3.5, I find that SRL performance varies widely among specific classes of nominal predicates, suggesting interesting directions for future work. I conclude,

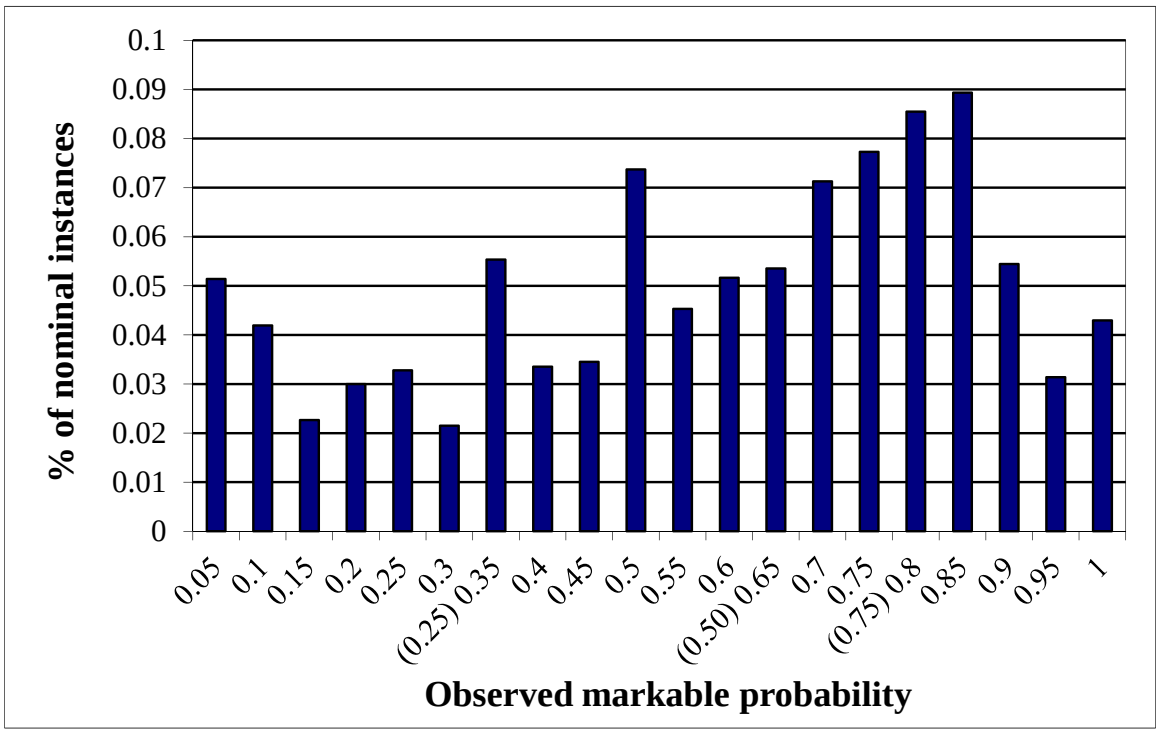



Figure 3.1: Distribution of nominal predicates. Each interval on the $x$-axis denotes a set of nominal predicates that are markable between $(x-5)\%$ and $x\%$ of the time in the Penn TreeBank corpus. The $y$-axis denotes the percentage of all nominal predicate instances in TreeBank that is occupied by nominal predicates in the interval. Quartiles are marked below the intervals. For example, the 0.25 quartile at $x = 0.35$ indicates that approximately 25% of all nominal instances are markable 35% of the time or less.

in Section 3.6, by motivating additional work on implicit argumentation, which is taken up in the following chapter.

## 3.2 Empirical analysis

As shown in Example 3.2, nominal predicates often surface without local arguments. In this section, I provide an analysis of implicit argumentation and its implications for the nominal SRL system developed in the previous chapter. On the whole, instances of predicates from the NomBank lexicon are markable only 57% of the time in the Penn TreeBank corpus. Figure 3.1 shows the distribution of nominal predicates in terms of the frequency with which

| | Markable-only evaluation | All-token evaluation | % loss |
|---|---|---|---|
| Precision | 0.8093 | 0.6832 | 15.58 |
| Recall | 0.7117 | 0.7039 | 1.10 |
| $F_1$ | 0.7574 | 0.6934 | 8.45 |

Table 3.1: Comparison of the markable-only and all-token evaluations of the SRL system from Chapter 2. In the all-token evaluation, argument identification is attempted for any nominal with at least one annotated (i.e., markable) instance in the training data.

they are markable. As shown, approximately 50% of nominal instances are markable 65% of the time or less, indicating that implicit argumentation is a very common phenomenon. This tendency toward implicit argumentation is also reflected in the percentage of roles that are filled in NomBank versus PropBank. In NomBank, 48% of possible roles are filled, whereas 61% of roles are filled in PropBank.

To assess the impact of implicit argumentation, I evaluated the nominal SRL system from Chapter 2 over each token in the testing section. The system attempted argument identification for all singular and plural nouns that have at least one annotated (i.e., markable) instance in the training portion of the NomBank corpus (morphological variations included). Table 3.1 gives a comparison of the results from the markable-only and all-token evaluations. As shown, assuming that all known nouns take local arguments results in a significant performance loss. This loss is due primarily to a drop in precision caused by false positive argument predictions made for nominal predicates with no local arguments. An example of this is shown below:[1]

(3.3) [$Arg_0$ 0.64 Canadian] [*Predicate* investment] rules require that big foreign takeovers meet that standard.

The sentence in Example 3.3 does not contain any constituents that are considered arguments to *investment* under the NomBank guidelines, but the SRL system (mistakenly) identifies

[1]In this dissertation, a number following an argument label indicates prediction probability.

*Canadian* as filling the $Arg_0$ position. Presumably, Canada is the entity imposing rules on those who invest; Canada is not the investing entity. Examples such as 3.3 demonstrate an important difference between nominal predicates and verbal predicates: the former are more flexible than the latter in terms of argument realization. Both classes of predicates may undergo syntactic alternations that change the linear order of argument expression; however, only nominal predicates routinely surface without explicit arguments. Thus, the approach to SRL used for verbal predicates is not entirely appropriate for nominal predicates.

## 3.3 Related work

Implicit argumentation was not accounted for in large-scale evaluation tasks until the 2008 Computational Natural Language Learning (CoNLL) Shared Task on dependency parsing (Surdeanu et al., 2008). In this task, systems were required to identify both syntactic and semantic dependency structure. Ground-truth syntactic dependency structure was automatically extracted from the constituent trees contained in the Penn TreeBank. Ground-truth semantic dependencies were extracted from the annotations in PropBank (for verbs) and NomBank (for nouns). In the semantic portion of the evaluation, systems were required to identify predicating verbs and nouns in addition to the corresponding arguments. Thus, systems in this evaluation were required to process instances such as Example 3.2 (p. 40).

Among all entries to the CoNLL Shared Task organized by Surdeanu et al. (2008), the system created by Johansson and Nugues (2008) fared the best overall and near the top for nominal predicates in particular. Johansson and Nugues's system used a classification-based approach to predicate-argument identification that is similar to the one presented in this chapter. However, the authors left open two important questions. First, there is the simple question of how effectively the system identifies nominal predicates. Second, the study does not evaluate the impact of the nominal predicate classifier on overall predicate-argument identification performance. As shown in the previous section, implicit argumentation has a significant negative effect on the standard approach to nominal SRL. It is important to

quantify how much of this loss can be recovered by adding a nominal predicate classifier.

In addition to answering the two questions above, this chapter develops a nominal predicate classifier that is, in many respects, simpler than the method used by Johansson and Nugues, which employed an individually trained classifier for each of the 4,704 predicates contained in NomBank. I opted for a single model capable of making predictions for all predicates in the corpus.

## 3.4 Argument-bearing predicate model

Given a sentence, the goal of predicate classification is to identify nouns that bear local arguments (i.e., those that would be annotated by NomBank). I treated this as a binary classification task over token nodes in the syntactic parse tree of a sentence. Once a token has been identified as bearing local arguments, it can be further processed by the argument identification model developed in Chapter 2. A token is ignored if it is not identified as argument-bearing.

The nominal predicate classifier was constructed using the greedy feature selection algorithm introduced in the previous chapter (see page 144 for details). I used the logistic regression solver of Fan et al. (2008) over a feature space binarized as described on page 32. Table A.2 (p. 135) presents the selected features. As shown by Table A.2, the sets of features selected for argument and nominal classification are quite different. Many of the features used for nominal classification were not used by Johansson and Nugues (2008) or Liu and Ng (2007). Below, I provide details for features that are not sufficiently explained in the table.

**Feature 1** captures the local syntactic structure that contains the candidate predicate. As shown in Table A.2, this is the most informative feature for nominal predicate classification, surpassing the predicate text itself (Feature 2). For a candidate predicate $n$, Feature 1 is actually a set of sub-features, one for each parse tree node between $n$ and the tree's root. The value of a sub-feature is the context-free grammar rule that expands the corresponding node

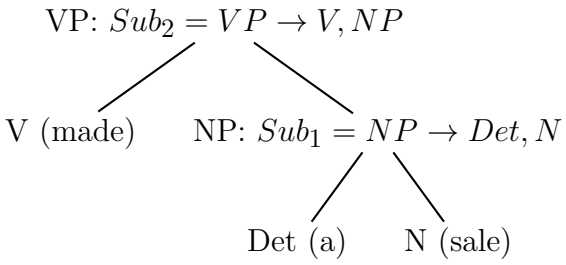



Figure 3.2: Context-free grammar rules for nominal predicate classification (Feature 1). The candidate nominal predicate *sale* is being classified. Arrows indicate grammar productions.

in the tree. Each value is additionally indexed according to its tree node distance from $n$. An example of Feature 1 with two sub-features is shown in Figure 3.2. In this example, the candidate nominal predicate *sale* is being classified. The first sub-feature ($Sub_1$) is derived from the parent of *sale*, and the second ($Sub_2$) from *sale*'s grandparent. The sub-features are indexed under the hypothesis that the same context-free grammar rule might indicate different outcomes at different levels in the parse tree. In Figure 3.2, $Sub_2$ indicates the use of a support verb structure. This in turn indicates a high likelihood that *sale* will take an $Arg_0$ that linearly precedes *made*.

**Feature 8** is a modified version of the parse tree path used for nominal argument identification. The modification is two-fold: first, the path begins at the candidate predicate and ends at the nearest support verb. As mentioned in the previous chapter, there exists a close link between nominal predicates and support verbs (see page 31). Second, the parse tree path is lexicalized, meaning it is concatenated with surface words from the beginning or end of the path. This finer-grained path captures the joint behavior of the syntactic and lexical content. For example, in the tree shown in Figure 3.2, the path from *sale* to *made* with a lexicalized destination would be $N \uparrow NP \uparrow VP \downarrow V : made$. A similar strategy is used for Features 5, 11, 13, 23, and 29. Lexicalization increases sparsity; however, it provides useful information and is often preferred over unlexicalized paths. Support verbs for this and other

features were automatically identified using the model described on page 134.

**Feature 16** leverages the existing content of PropBank to identify argument-bearing nominal predicates. The value for this feature is the probability that the context (± 5 words) of a nominal predicate is generated by a unigram language model trained over the PropBank argument words for the corresponding verb. All named entities are normalized to their entity type using BBN's IdentiFinder (Bikel et al., 1999), and adverbs are normalized to their related adjective using the *ADJADV* dictionary provided by NomBank. The normalization of adverbs to adjectives is motivated by the fact that adverbial modifiers of verbs typically have corresponding adjectival modifiers for nominal predicates. This is shown below:

(3.4) [$Arg_0$ John] [*Predicate* gossiped] [*Manner* quietly] with his coworkers.
(3.5) John's quiet [*Predicate?* gossip] was overheard.

Example 3.4 provides evidence that a predicate such as *gossip* takes arguments when surrounded by *Person* mentions and adverbs such as *quiet*. This information is useful when classifying the nominal predicate *gossip* in Example 3.5, where we find a similar named entity and adjectival modifier. Example 3.5 is indeed markable.

### LibLinear model configuration

As with other LibLinear models in this dissertation, I found it helpful to adjust the per-class costs for nominal predicate classification. I used cost $c = 2$ and $w+ = 1$, which were identified during feature selection. Two additional parameters were set: (1) the classification bias (= 1), and (2) the prediction threshold $t_{pred}$. The latter functions similarly to the $t_{arg}$ threshold used for argument classification. Any candidate predicate scoring higher than $t_{pred}$ is passed to the argument identifier. All other candidate predicates are ignored. Actual values for $t_{pred}$ are discussed in the following section.

| | Precision (%) | Recall (%) | $F_1$ (%) |
|---|---|---|---|
| Baseline | 55.5 | 97.8 | 70.9 |
| MLE | 68.0 | 90.6 | 77.7 |
| LibLinear | 86.6 | 88.5 | 87.6 |

Table 3.2: Evaluation results for identifying nominal predicates that take local arguments. The first column indicates which nominal classifier was used.

## 3.5 Evaluation

I evaluated the model described above using a practical setup in which the nominal SRL system had to process every token in a sentence. The system could not safely assume that each token took local arguments; rather, this decision had to be made automatically. In Section 3.5.1, I present results for the automatic identification of nominal predicates with local arguments. Then, in Section 3.5.2, I present results for the combined task in which nominal classification is followed by argument identification.

### 3.5.1 Predicate evaluation

Following standard practice, I trained the nominal classifier over token nodes in TreeBank sections 2-21. All syntactic parse trees were automatically generated by Charniak's reranking syntactic parser (Charniak and Johnson, 2005), and only those tokens with at least one annotated (i.e., markable) instance in NomBank were retained for training. As mentioned above, the classifier imposes a prediction threshold $t_{pred}$ on the classification decisions. The value of $t_{pred}$ was found by maximizing the nominal $F_1$ score on the development section (24) of NomBank ($t_{pred} = 0.47$). The resulting model was tested over all token nodes in section 23 of TreeBank. For comparison, I implemented the following simple classifiers:

- The **baseline** model classifies a token as locally bearing arguments if it is a singular or plural noun that is found to be markable at least once in the training sections of

NomBank. As shown in Table 3.2, this classifier achieves nearly perfect recall. Recall is less than 100% due to (1) part-of-speech errors from the syntactic parser and (2) nominal predicates that were not annotated in the training data but exist in the testing data.

- The **MLE** model operates similarly to the baseline, but also produces a score for the classification. The value of the score is equal to the probability that the nominal bears local arguments, as observed in the training data. When using this model, $t_{pred} = 0.33$. As shown by Table 3.2, this exchanges recall for precision and leads to a performance increase of approximately 8 $F_1$ points.

The last row in Table 3.2 shows the results for the feature-based nominal predicate classifier. This model outperforms the others by a wide margin, achieving balanced precision and recall scores near 88% $F_1$. In addition, the feature-based model is able to recover from part-of-speech errors because it does not filter out non-noun candidates; rather, it combines part-of-speech information with other lexical and syntactic features to classify nominal predicates.

Interesting observations can be made by grouping nominal predicates according to the probability with which they are markable in the corpus. Recall Figure 3.1 (page 42), which shows the distribution of markable nominal predicates across intervals of markability. Using this view of the data, Figure 3.3 presents the overall $F_1$ scores for the baseline and LibLinear nominal classifiers.[2] As shown, gains in nominal classification diminish as nominal predicates become more reliably associated with local arguments (i.e., as one moves right along the $x$-axis). This is because the baseline system makes fewer errors for predicates in intervals to the right. Furthermore, nominal predicates that are rarely markable (i.e., those in interval 0.05) remain problematic due to a lack of positive training instances and the unbalanced nature of the classification task.

Overall, however, the feature-based model exhibits substantial gains versus the baseline system, particularly for nominals occupying the left-most intervals of Figure 3.3. As will

[2]Baseline and MLE scores are identical above the MLE threshold.

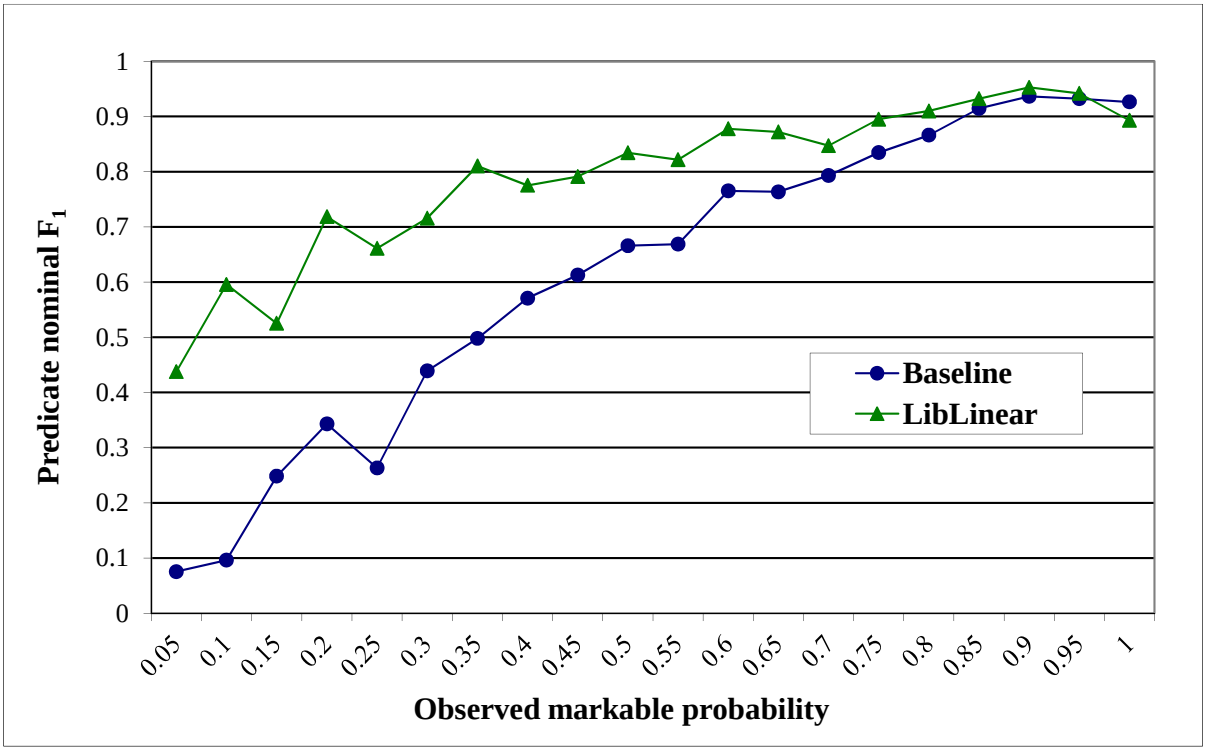



Figure 3.3: Nominal classification performance with respect to the distribution in Figure 3.1 (page 42). The $y$-axis denotes the combined $F_1$ for nominal predicates that occupy the interval given on the $x$-axis.

be shown in the next section, these gains in nominal predicate classification transfer well to gains in argument identification.

### 3.5.2 Combined predicate-argument evaluation

I now turn to the task of combined predicate-argument classification. In this task, systems must first identify nominal predicates that bear local arguments. I evaluated three configurations based on the nominal classifiers from the previous section. Each configuration uses the argument classification system described in Chapter 2. Table 3.3 presents the results of using the three configurations for combined predicate-argument classification. As shown in Table 3.3, overall argument classification $F_1$ suffers a relative loss of more than 8% under the baseline assumption that all known nouns bear local arguments. The MLE predicate clas-

| Predicate classifier used | $t_{pred}$ | $t_{arg}$ | All-token argument $F_1$ (%) | Loss (%) |
|---|---|---|---|---|
| Baseline | N/A | N/A | 69.3 | 8.5 |
| MLE | 0.23 | 0.44 | 69.9 | 7.7 |
| Logistic regression | 0.32 | 0.43 | 71.1 | 6.1 |

Table 3.3: Comparison of the combined predicate-argument classifiers in the all-token evaluation. The first column indicates which nominal predicate classifier was used. All configurations used the argument classification system described in Chapter 2. The second and third columns give the prediction thresholds used. The fourth column gives overall argument $F_1$ scores, and the last column gives the loss with respect to the standard evaluation task in which the system is given an argument-bearing predicate (this was used in the previous chapter).

sifier is able to reduce this loss slightly. The LibLinear predicate classifier reduces this loss even further, resulting in an overall argument classification $F_1$ of 71.1%. This improvement is the direct result of filtering out nominal instances that do not bear local arguments.

Similarly to the predicate classification evaluation of Section 3.5.1, one can view argument classification performance with respect to the prior probability that a nominal bears local arguments as determined by the training data. This is shown in Figure 3.4 for the three configurations. The configuration using the MLE nominal predicate classifier obtained an argument $F_1$ of zero for nominal predicates below its prediction threshold. Compared to the baseline predicate classifier, the LibLinear classifier achieved argument classification gains as large as 163% (interval 0.05), with an average gain of 58% for intervals 0.05 to 0.3. As with nominal classification, argument classification gains versus the baseline diminish for nominal predicates that occupy intervals further to the right of the graph. I observed an average gain of only 1% for intervals 0.35 through 1.00. A couple factors contribute to this result. First, the feature-based predicate model is not substantially more accurate than the baseline predicate model for the highest intervals (see Figure 3.3 on page 50). Second, the argument prediction model has substantially more training data for the nominal predicates in intervals 0.35 to 1.00; NomBank contains many more instances of these predicates than the predicates occupying lower intervals. Thus, even if the nominal classifier makes a false

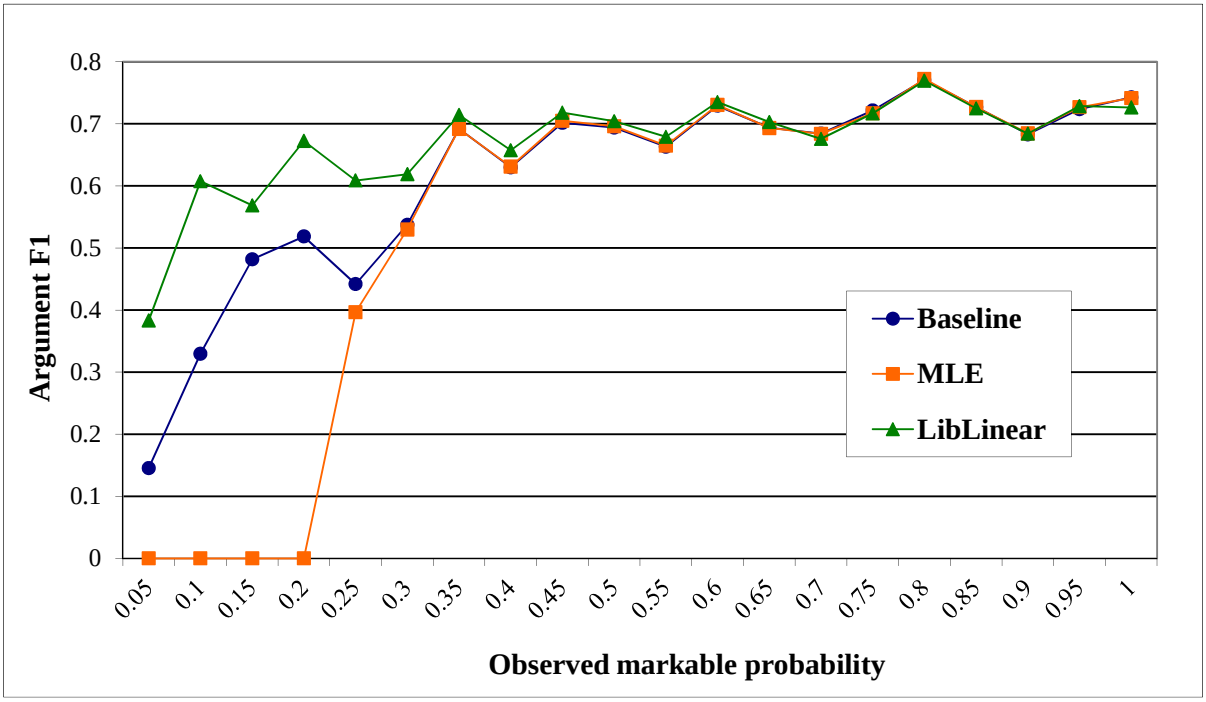



Figure 3.4: All-token argument classification performance with respect to the distribution in Figure 3.1 (p. 42). The $y$-axis denotes the combined argument $F_1$ for nominal predicates in the interval.

positive prediction in the 0.35 to 1.00 interval range, the argument model may correctly avoid labeling any arguments.

As noted in Section 3.3, the results in Table 3.3 are not directly comparable to the results of the recent CoNLL Shared Task (Surdeanu et al., 2008). This is because the semantic labeled $F_1$ score used in the Shared Task combined predicate and argument predictions into a single score. The same combined $F_1$ score for my best two-stage nominal SRL system (logistic regression predicate and argument models) is 79.1%. This compares favorably to the best score of 76.6% reported by Surdeanu et al..

### 3.5.3 NomLex-based analysis of results

As mentioned previously, NomBank annotates many classes of deverbal and non-deverbal predicates. These predicates have been semi-automatically categorized on syntactic and

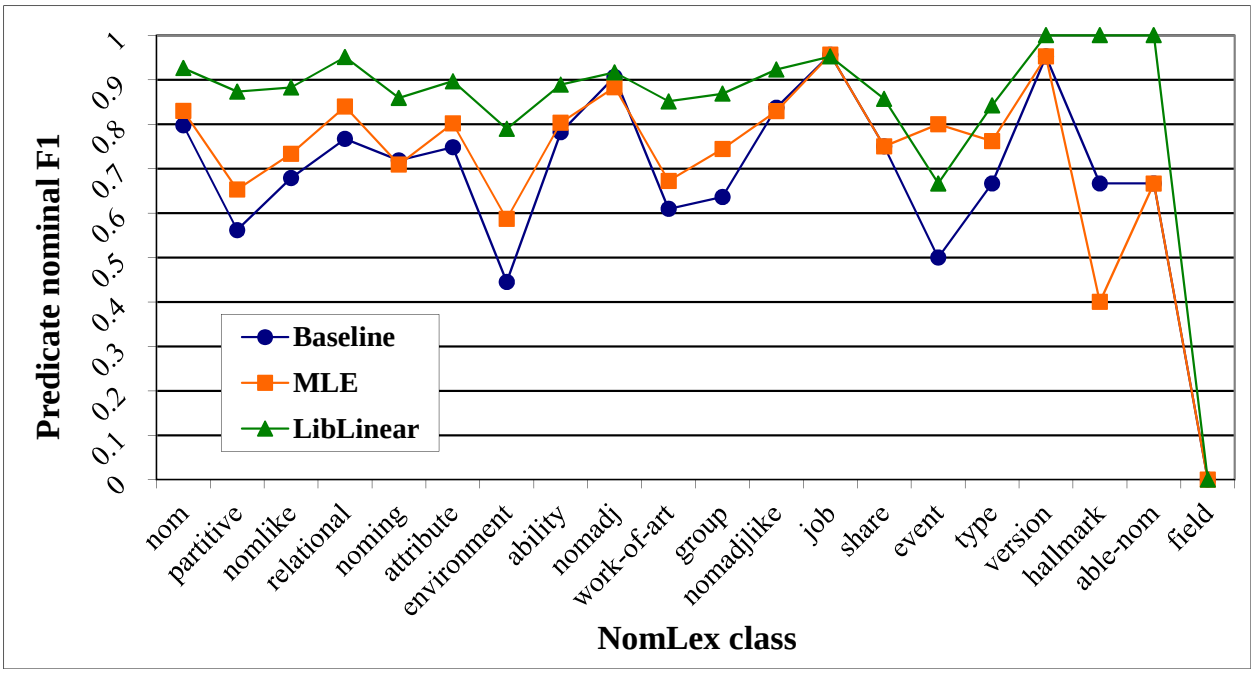



Figure 3.5: Nominal classification performance with respect to the NomLex classes in Figure 2.3. The $y$-axis denotes the combined $F_1$ for nominal predicates in the class.

semantic bases by the NomLex-PLUS resource (Meyers, 2007b). To help understand what types of nominal predicates are particularly affected by implicit argumentation, I further analyzed performance with respect to these classes.

Recall Figure 2.3 (p. 20), which shows the distribution of nominal predicates across classes defined by the NomLex resource. As shown in Figure 3.5, many of the most frequent classes exhibit significant gains. For example, the classification of partitive nominal predicates (13% of all nominal instances) with the LibLinear classifier results in gains of 55.5% and 33.7% over the baseline and MLE classifiers, respectively. For the five most common classes, which constitute 82% of all nominal predicate instances, I observed average gains of 27.5% and 19.3% over the baseline and MLE classifiers, respectively.

Table 3.4 separates predicate and argument classification results into sets of deverbal (NomLex class *nom*), deverbal-like (NomLex class *nom-like*), and all other nominal predicates. A deverbal-like predicate is closely related to some verb, although not morphologically.

| | Predicate $F_1$ (%) | | | Combined predicate-argument $F_1$ (%) | | |
|---|---|---|---|---|---|---|
| | Deverbal | Deverbal-like | Other | Deverbal | Deverbal-like | Other |
| Baseline | 79.8 | 67.9 | 67.6 | 70.6 | 67.4 | 74.5 |
| MLE | 83.0 | 73.3 | 74.9 | 72.1 | 66.4 | 76.8 |
| LibLinear | 92.6 | 88.3 | 89.1 | 72.8 | 71.8 | 78.5 |

Table 3.4: Predicate and combined predicate-argument classification $F_1$ scores for deverbal, deverbal-like, and other nominal predicates in the all-token evaluation. The first column indicates which nominal classifier was used. All configurations used the nominal SRL system described in Chapter 2.

For example, the noun *accolade* shares argument interpretation with the verb *award*, but the two are not morphologically related. As shown by Table 3.4, predicate classification tends to be easier - and argument classification harder - for deverbals when compared to other types of nominal predicates. For combined nominal-argument $F_1$, the difference between deverbal/deverbal-like predicates and the *others* is due primarily to relational nominals, which are included the *others* column. Relational nominals are accurately classified by the logistic regression model ($F_1 = 0.95$ in Figure 3.5); additionally, relational nominals exhibit a high rate of argument incorporation (i.e., predicate-as-argument behavior), which is easily handled by the maximum-likelihood model described in Section 2.3.

### 3.5.4 Analysis of end-to-end nominal SRL speed

The combined predicate-argument classification system presented in this chapter is capable of operating in an end-to-end fashion over completely unstructured text.[3] In this section, I provide asymptotic and empirical analyses for the system's performance.

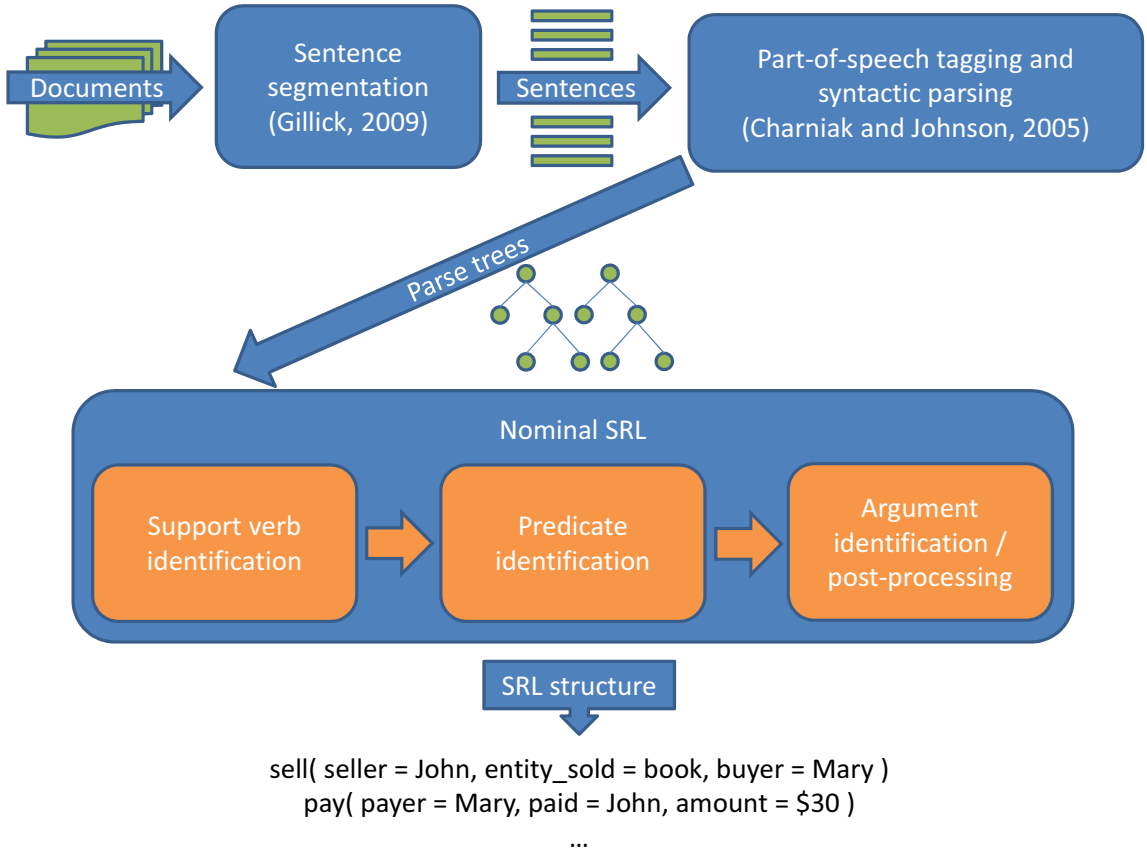



Figure 3.6: End-to-end nominal SRL architecture.

### Processing components

Figure 3.6 shows the end-to-end nominal SRL architecture. Processing begins by segmenting each document into a sequence of sentences. This step is performed using Gillick's (2009) SVM-based segmenter. Each sentence is then tagged for part-of-speech and syntactic information using the August 2006 version of Charniak and Johnson's (2005) parser. The nominal SRL classifier chain then labels support verbs (see page 134), nominal predicates (page 45), and arguments (page 28). In the final post-processing step, argument conflicts

[3]The nominal SRL system is freely available for non-commercial use. Please contact the author at gerber.matthew@gmail.com for more information.

are removed and incorporated arguments are identified (page 33). In the following sections, I provide analyses for various components in Figure 3.6.

### Asymptotic analysis

One can analyze the computational complexity of the processing components with respect to either the number of tokens in a document (denoted by $d$) or the number of tokens in a sentence (denoted by $s$).

- **Sentence segmentation** is $O(d)$. The component needs to scan the document for ambiguous punctuation marks, which might indicate the end of a sentence.
- **Part-of-speech tagging and syntactic parsing** is $O(s^3)$. The Charniak parser is a re-ranker working on top of a chart parser that is $O(s^3)$ (Charniak et al., 1998).
- **Support verb identification** is $O(s)$. Each token is classified.
- **Predicate identification** is $O(s)$. Each token is classified.
- **Argument identification** is $O(s^2)$. The length $s$ of a sentence is related to the number of nodes $n$ in the sentence's perfect binary tree (this is the worst case) by $s = \lceil \frac{n}{2} \rceil$. Thus, $\frac{n}{2} \leq s < \frac{n}{2} + 1$ and $2s - 2 < n \leq 2s$. At most, then, there are $2s$ nodes in the tree for a sentence of length $s$. During argument identification, each of these nodes is classified for each predicate in the sentence. $s$ is a theoretical upper bound for the number of predicates in a sentence of length $s$, giving $O(s^2)$ for all argument node classifications; however, in practice there are far fewer than $s$ predicates per sentence (see empirical results below).
- **Argument post-processing** is $O(s^2)$. The post-processor looks at each argument node for each predicate node and detects/resolves conflicts in a constant number of operations. The maximum number of argument nodes for a predicate node is less than the number of nodes in the tree because argument nodes are not allowed to overlap

| | |
|---|---|
| Documents | 1000 |
| Sentences per document | 8 |
| Words per sentence | 29 |
| Words per minute | 1134 |
| Predicate trees per minute | 120 |
| Sentence segmentation total (seconds) | 24 |
| Part-of-speech tagging and syntactic parsing total (minutes) | 158 |
| Support verb, predicate, and argument identification / post-processing (minutes) | 52 |

Table 3.5: Empirical speed performance of the nominal SRL system. Here, *Predicate trees* refers to a single predicate node and its associated support verbs and argument nodes.

the predicate node. Thus, the previous upper bound of $2s$ is also an upper bound on the number of argument nodes for a predicate node. There are $O(s)$ predicate nodes in a sentence, for a total of $O(s^2)$ for argument conflict resolution. The post-processor also labels incorporated arguments for each of the detected predicates, adding $O(s)$ for a total of $O(s^2)$. Note again that the practical number of predicates in a sentence of length $s$ is far less than $s$.

**Empirical analysis**

To test the speed performance of the nominal SRL system empirically, I randomly selected 1000 documents from the Gigaword corpus (Graff, 2003). Documents from this corpus contain reports from a variety of newswire agencies. Thus, the genre is quite similar to that of the training data used for the nominal SRL system. In total, the 1000 document sub-corpus contained approximately 232,000 tokens and 8,000 sentences. Each document was provided to the system without paragraph markings or any other structural information. The processing hardware consisted of standard desktop machine with a 2.8 GHz Pentium 4 CPU and 3 GB of main memory.

Table 3.5 lists the key performance statistics. As shown, sentence splitting contributed a negligible amount of time (24 seconds) to the total of more than 3.5 hours. Three quarters of the time was spent performing part-of-speech tagging and syntactic parsing with the Char-

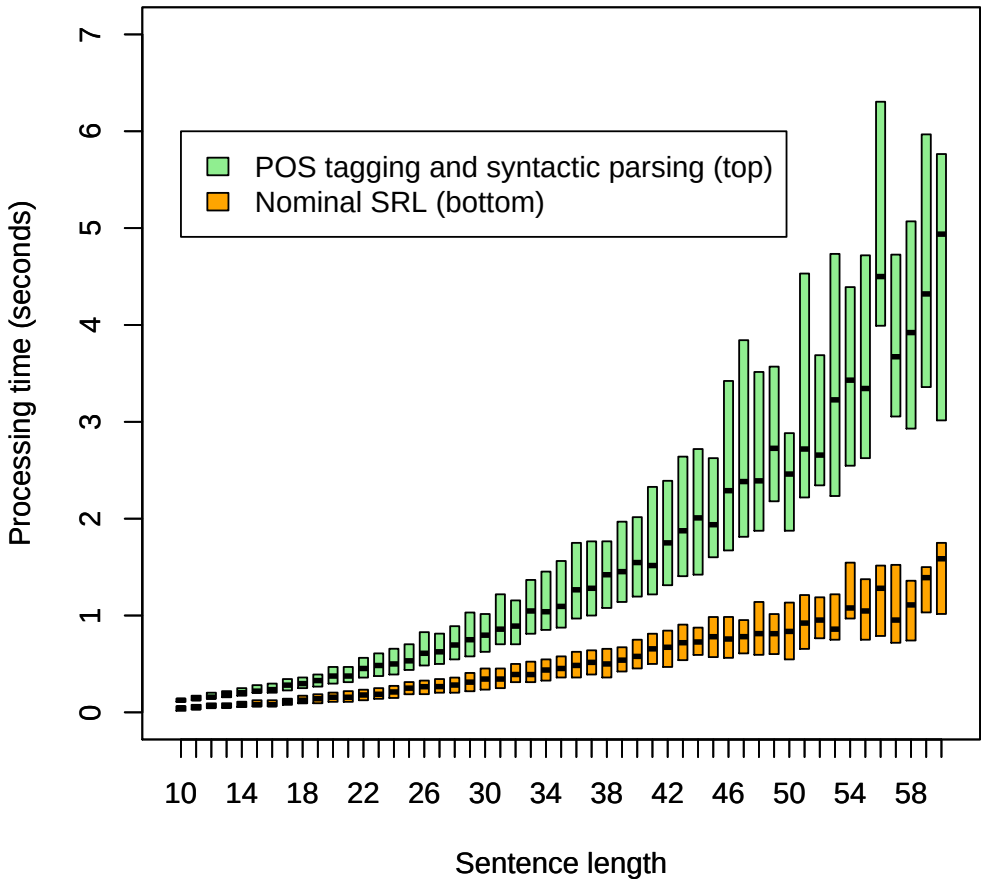



Figure 3.7: Box plot of nominal SRL speed. Each vertical box shows the range of times observed for sentences with length given along the $x$-axis. Medians are indicated by the black bar in each box, and the box spans from the first to the third quartiles.

niak parser. As mentioned above, syntactic analysis has the slowest worst-case performance of any of the components. Figure 3.7 demonstrates the cubic worst-case visually. Contrast this with the performance of the nominal SRL components, which consumed one quarter of the total time. Asymptotically, argument identification and post-processing are quadratic in the worst-case; however, the worst case (i.e., all tokens being predicates) rarely, if ever, occurs. This can be seen in Figure 3.7, where the slowing of the nominal SRL components

is roughly linear in the sentence length.

Speed performance of the end-to-end nominal SRL system could be enhanced using two approaches. First, one could instantiate multiple instances of the architecture shown in Figure 3.6 using multiple systems. Since there are no inter-document dependencies, documents can be distributed among these systems, increasing performance. Second, one could replace the cubic time syntactic parser with a linear time dependency parser (Sagae and Lavie, 2005). Dependency parsing has received a significant amount of attention in recent years. Although its parsing formalism is not as expressive as the constituent formalism used by parsers such as Charniak's, dependency parsing often produces useful results very efficiently.

## 3.6 Conclusions

The application of nominal SRL to practical NLP problems requires a system that is able to accurately process each token it encounters. Previously, it was unclear whether the models proposed by Jiang and Ng (2006) and Liu and Ng (2007) would operate effectively in such an environment. The systems described by Surdeanu et al. (2008) are designed with this environment in mind, but their evaluation did not focus on the issue of implicit argumentation. These two problems motivate the work presented in this chapter.

The contribution of this chapter is three-fold. First, it shows that the state-of-the-art nominal SRL system of the previous chapter suffers a substantial performance degradation when evaluated over nominal predicates whose arguments are implicit. Second, it identifies a set of features - many of them new - that can be used to accurately detect nominal predicates with local arguments, thus increasing the overall performance of the nominal SRL system. The nominal predicate model also allows the nominal SRL system to operate in real-world settings over completely unstructured text. Third, the evaluation results suggest interesting directions for future work. As described in Section 3.5.2, many nominal predicates do not have enough labeled training data to produce accurate argument classifiers. The generalization procedures developed by Gordon and Swanson (2007) for PropBank SRL and

Padó et al. (2008) for NomBank SRL might alleviate this problem.

Most important, however, is the following observation: the logistic regression nominal predicate classifier is able to accurately filter out predicates whose arguments are implicit; however, this model cannot actually *recover* implicit arguments, which are often expressed in the surrounding discourse. It is also the case that nominal predicates with local arguments often have additional implicit arguments somewhere in the discourse; these, too, are ignored by the models in this chapter. In the following chapter, I describe an in-depth study of implicit arguments and their recovery from the discourse. This topic has received very little attention from NLP researchers; however, as I will show, implicit argument recovery is (1) a fundamental process within discourse semantics, and (2) a process that can be modeled effectively.

# CHAPTER 4
# Identifying implicit arguments

## 4.1 Introduction

The previous chapter showed that it is possible to accurately distinguish nominals that bear local arguments from those that do not. This is an important step because it frees us from the assumption that all nominal predicates take local arguments - an assumption shown to be false. Ultimately, the goal is to use the output of the nominal SRL system as input for other NLP tasks such as QA, IE, and SMT. Being able to process all tokens in a document is essential for such tasks.

Despite these improvements, though, the system developed in the previous chapter does not address a fundamental question regarding implicit arguments: if an argument is implicit (i.e., missing) in the local context of a predicate, might the argument be located somewhere in the wider discourse? The previous chapter stopped short of answering this question, opting instead for an approach that ignores predicates whose arguments are implicit. The current chapter directly addresses this important question.

As an initial example, consider the following sentence, which is taken from the Penn TreeBank:

(4.1) A SEC proposal to ease [$Arg_1$ reporting] [$Predicate$ requirements] [$Arg_2$ for some company executives] would undermine the usefulness of information on insider trades, professional money managers contend.

The NomBank role set for *requirement* is shown below:

> Frame for *requirement*, role set 1:
> $Arg_0$: the entity that is requiring something

$Arg_1$: the entity that is required
$Arg_2$: the entity of which something is being required

In Example 4.1, the predicate has been annotated with the local argument labels provided by NomBank. As shown, NomBank does not annotate an $Arg_0$ for this instance of the *requirement* predicate; however, a reasonable interpretation of the sentence is that *SEC* is the entity that is requiring something.[1] This dissertation refers to arguments such as *SEC* in Example 4.1 as implicit. When all arguments for a predicate are implicit, one obtains the situation addressed in the previous chapter; however, this is the extreme case. In Example 4.1, some arguments are local (i.e., $Arg_1$ and $Arg_2$) and some are not (i.e., $Arg_0 = SEC$).

Building on Example 4.1, consider the following sentence, which directly follows Example 4.1 in the corresponding TreeBank document:

(4.2) Money managers make the argument in letters to the agency about [$Arg_1$ rule] [*Predicate* changes] proposed this past summer.

The NomBank role set for *change* is shown below:

Frame for *change*, role set 1:
$Arg_0$: the entity that initiates the change
$Arg_1$: the entity that is changed
$Arg_2$: the initial state of the changed entity
$Arg_3$: the final state of the changed entity

Similarly to the previous example, 4.2 shows the local argument labels provided by NomBank. These labels only indicate that rules have been changed. For a full interpretation, Example 4.2 requires an understanding of Example 4.1. Without the latter, the reader has no way of knowing that *the agency* in 4.2 actually refers to the same entity as *SEC* in 4.1. As part of the reader's comprehension process, this entity is identified as the filler for the $Arg_0$ role in Example 4.2. This identification must occur in order for these two sentences to form a coherent discourse.

[1]The Securities and Exchange Commission (SEC) is responsible for enforcing investment laws in the United States.

From these examples, it is clear that the scope of implicit arguments quite naturally spans sentence boundaries. Thus, if one wishes to recover implicit arguments as part of the SRL process, the argument search space must be expanded beyond the traditional, single-sentence window used in virtually all prior SRL research. What can we hope to gain from such a fundamental modification of the problem? Consider the following question, which targets Examples 4.1 and 4.2 above:

(4.3) Who changed the rules regarding reporting requirements?

Question 4.3 is a factoid question, meaning it has a short, unambiguous answer in the targeted text. This type of question has been studied extensively in the Text Retrieval Conference (TREC) Question Answering Track (Dang et al., 2007). Using the evaluation data from this track, Pizzato and Mollá (2008) showed that SRL can improve the accuracy of a QA system; however, a traditional SRL system alone is not enough to recover the implied answer to Question 4.3: *SEC* or *the agency*. Successful implicit argument identification provides the answer in this case.

This chapter presents an in-depth study of implicit arguments for nominal predicates.[2] The following section surveys a broad spectrum of research related to implicit argument identification. Section 4.3 describes the study's implicit argument annotation process and the data it produced. The implicit argument identification model is formulated in Section 4.4 and evaluated in Section 4.5. Discussion of results is provided in Section 4.6, and the chapter concludes in Section 4.7.

## 4.2 Related work

The research presented in this chapter is related to a wide range of topics in cognitive science, linguistics, and NLP. This is partly due to the discourse-based nature of the problem. In single-sentence SRL, one can ignore the discourse aspect of language and still obtain

[2] A condensed version of this study was published by Gerber and Chai (2010).

high marks in an evaluation (for examples, see Carreras and Màrquez (2005) and Surdeanu et al. (2008)); however, implicit argumentation forces one to consider the discourse context in which a sentence exists. Much has been said about the importance of discourse to language understanding, and this section will identify the points most relevant to implicit argumentation.

### 4.2.1 Discourse comprehension in cognitive science

In linguistics, the traditional view of sentence-level semantics has been that meaning is compositional. That is, one can derive the meaning of a sentence by carefully composing the meanings of its constituent parts (Heim and Kratzer, 1998). There are counterexamples to a compositional theory of semantics (e.g., idioms), but those are more the exception than the rule. Things change, however, when one starts to group sentences together to form coherent textual discourses. Consider the following examples, borrowed from Sanford (1981) (p. 5):

(4.4) Jill came bouncing down the stairs.
(4.5) Harry rushed off to get the doctor.

Examples 4.4 and 4.5 describe three events: *bounce*, *rush*, and *get*. These events are intricately related. One cannot simply create a conjunction of the propositions *bounce*, *rush*, and *get* and expect to arrive at the author's intended meaning, which presumably involves Jill's becoming injured by her fall and Harry's actions to help her. The mutual dependence of these sentences can be further shown by considering a variant of the situation described in Examples 4.4 and 4.5:

(4.6) Jill came bouncing down the stairs.
(4.7) Harry rushed over to kiss her.

The interpretation of Example 4.6 is is vastly different from the interpretation of Example 4.4. In 4.4, Jill becomes injured whereas in 4.6 she is quite happy.

Examples 4.4-4.7 demonstrate the fact that sentences do not have a fixed, compositional interpretation; rather, a sentence's interpretation depends on the surrounding context. The

standard compositional theory of sentential semantics largely ignores contextual information provided by other sentences. The single-sentence approach to SRL operates similarly. In both of these methods, the current sentence provides all of the semantic information. In contrast to these methods - and aligned with the preceding discussion - this chapter presents methods that rely heavily on surrounding sentences to provide additional semantic information. This information is used to interpret the current sentence in a more complete fashion.

Examples 4.4-4.7 also show that the reader's knowledge plays a key role in discourse comprehension. Researchers in cognitive science have proposed many models of reader knowledge. Schank and Abelson (1977) proposed stereotypical event sequences called *scripts* as a basis for discourse comprehension. In this approach, readers fill in a discourse's semantic gaps with knowledge of how a typical event sequence might unfold. In Examples 4.4 and 4.5, the reader knows that people typically call on a doctor only if someone is hurt. Thus, the reader automatically fills the semantic gap caused by the ambiguous predicate *bounce* with information about doctors and what they do. Similar observations have been made by van Dijk (1977) (p. 4), van Dijk and Kintsch (1983) (p. 303), Graesser and Clark (1985) (p. 14), and Carpenter et al. (1995). Inspired by these ideas, the model developed in this chapter relies partly on large text corpora, which are treated as repositories of typical event sequences. The model uses information extracted from these event sequences to identify implicit arguments.

### 4.2.2 Automatic relation discovery

Examples 4.4 and 4.5 in the previous section show that understanding the relationships between predicates is a key part of understanding a textual discourse. In this section, I review work on automatic predicate relationship discovery, which attempts to extract these relationships automatically.

Lin and Pantel (2001) proposed a system that automatically identifies relationships similar to the following:

(4.8) $X$ *eats* $Y \leftrightarrow X$ *likes* $Y$

This relationship creates a mapping between the participants of the two predicates. One can imagine using such a mapping to fill in the semantic gaps of a discourse that describes a typical set of events in a restaurant. In such a discourse, the author probably will not state directly that $X$ likes $Y$; however, the reader might need to infer this in order to make sense of the fact that $X$ left a large tip for the waiter.

Lin and Pantel created mappings such as 4.8 using a variation of the so-called "distributional hypothesis" posited by Harris (1985), which states that words occurring in similar contexts tend to have similar meanings. Lin and Pantel applied the same notion of similarity to dependency paths. For example, the inference rule in Example 4.8 is identified by examining the sets of words in the two $X$ positions and the sets of words in the two $Y$ positions. When the two pairs of sets are similar, it is implied that the two dependency paths from $X$ to $Y$ are similar as well. In Example 4.8, the two dependency paths are as follows:

$$\begin{aligned} X &\xleftarrow{subject} eats \xrightarrow{object} Y \\ X &\xleftarrow{subject} likes \xrightarrow{object} Y \end{aligned} \tag{4.9}$$

One drawback of this method is that it assumes the implication is symmetric. Although this assumption is correct in many cases, it often leads to invalid inferences. In Example 4.8, it is not always true that if $X$ likes $Y$ then $X$ will eat $Y$. The opposite - that $X$ eating $Y$ implies $X$ likes $Y$ - is more plausible but not certain.

Bhagat et al. (2007) extended the work of Lin and Pantel to handle cases of asymmetric relationships. The basic idea proposed by Bhagat et al. is that, when considering a relationship of the form $\langle x, p_1, y\rangle \leftrightarrow \langle x, p_2, y\rangle$, if $p_1$ occurs in significantly more contexts (i.e., has more options for $x$ and $y$) than $p_2$, then $p_2$ is likely to imply $p_1$ but not vice versa. Returning to Example 4.8, we see that the correct implication will be derived if *likes* occurs in significantly more contexts than *eats*. The intuition is that the more general concept (i.e.,

*like*) will be associated with more contexts and is more likely to be implied by the specific concept (i.e., *eat*). As shown by Bhagat et al., the system built around this intuition is able to effectively identify the directionality of many inference rules.

Zanzotto et al. (2006) presented another study aimed at identifying asymmetric relationships between verbs. For example, the asymmetric entailment relationship *X wins* $\rightarrow$ *X plays* holds, but the opposite (*X plays* $\rightarrow$ *X wins*) does not. This is because not all those who play a game actually win. To find evidence for this automatically, the authors examined constructions such as the following, adapted from Zanzotto et al.:

(4.10) The more experienced tennis player won the match.

The underlying idea behind the authors' approach is that asymmetric relationships such as *X wins* $\rightarrow$ *X plays* are often entailed by constructions involving agentive, nominalized verbs as the logical subjects of the main verb. In Example 4.10, the agentive nominal "player" is logical subject to "won", the combination of which entails the asymmetric relationship of interest. Thus, to validate such an asymmetric relationship, Zanzotto et al. examined the frequency of the "player win" collocation using Google hit counts as a proxy for actual corpus statistics.

A number of other studies (e.g., those by Szpektor et al. (2004) and Pantel et al. (2007)) have been conducted that are similar to the work described above. In general, such work focuses on the automatic acquisition of entailment relationships between verbs. Although this work has often been motivated by the need for lexical-semantic information in tasks such as automatic question answering, it is also relevant to the task of implicit argument identification because the derived relationships implicitly encode a participant role mapping between two predicates. For example, given a missing $Arg_0$ for a *like* predicate and an explicit $Arg_0 = John$ for an *eat* predicate in the preceding discourse, inference rule 4.8 would help identify the implicit $Arg_0 = John$ for the *like* predicate.

The missing link between previous work on verb relationship identification and the task of implicit argument identification is that previous verb relations are not defined in terms

of the $Arg_n$ positions used by NomBank. Rather, positions like *subject* and *object* are used (see Example 4.9). In order to identify implicit arguments in NomBank, one needs inference rules between specific argument positions (e.g., *eat*:$Arg_0$ and *like*:$Arg_0$). In the current chapter, I propose methods of automatically acquiring these fine-grained relationships for verbal and nominal predicates using existing corpora. I also propose a method of using these relationships to recover implicit arguments.

### 4.2.3 Coreference resolution and discourse processing

The current chapter will make heavy use of the notions of *reference* and *coreference*. The referent of a linguistic expression is the real or imagined entity to which the expression refers. Coreference, therefore, is the condition of two linguistic expressions having the same referent. In the following examples from the Penn TreeBank, the underlined spans of text are coreferential:

(4.11) "Carpet King sales are up 4% this year," said owner Richard Rippe.
(4.12) He added that the company has been manufacturing carpet since 1967.

Non-trivial instances of coreference (e.g., *Carpet King* and *the company*) allow the author to repeatedly mention the same entity without introducing redundancy into the discourse. Pronominal anaphora is a subset of coreference in which one of the referring expressions is a pronoun. For example, *he* in Example 4.12 refers to the same entity as *Richard Rippe* in Example 4.11. These examples demonstrate noun phrase coreference. Events, indicated by either verbal or nominal predicates, can also be coreferential when mentioned multiple times in a document (Wilson, 1974; Chen and Ji, 2009).

For many years, the Automatic Content Extraction (ACE) series of large-scale evaluations (ACE, 2008) has provided a test environment for systems designed to identify these and other coreference relations. Systems based on the ACE datasets typically take a supervised learning approach to coreference resolution in general (Versley et al., 2008) and pronominal anaphor in particular (Yang et al., 2008).

A phenomenon similar to the implicit argument has been studied in the context of Japanese anaphora resolution, where a missing case-marked constituent is viewed as a zero-anaphoric expression whose antecedent is treated as the implicit argument of the predicate of interest. This behavior has been annotated manually by Iida et al. (2007), and researchers have applied standard SRL techniques to this corpus, resulting in systems that are able to identify missing case-marked expressions in the surrounding discourse (Imamura et al., 2009). Sasano et al. (2004) conducted similar work with Japanese indirect anaphora. The authors used automatically derived nominal case frames to identify antecedents. However, as noted by Iida et al., grammatical cases do not stand in a one-to-one relationship with semantic roles in Japanese (the same is true for English).

Many other discourse-level phenomena interact with coreference. For example, Centering Theory (Grosz et al., 1995) focuses on the ways in which referring expressions maintain (or break) coherence in a discourse. These so-called "centering shifts" result from a lack of coreference between salient noun phrases in adjacent sentences. Discourse Representation Theory (DRT) (Kamp and Reyle, 1993) is another prominent treatment of referring expressions. DRT embeds a theory of coreference into a first-order, compositional semantics of discourse.

In Centering Theory, DRT, and the ACE coreference competitions, coreference relationships hold between relatively small constituents in one or more sentences (e.g., a pronoun and its noun phrase antecedent); however, researchers have also investigated relationships that hold between larger segments of text, including full sentences. Consider the following example, adapted from Rhetorical Structure Theory (Taboada and Mann, 2006):

(4.13) [*Objective* The visual system resolves confusion] [*Means* by applying knowledge of properties of the physical world].

In Example 4.13, the objective (resolution of confusion) is accomplished by a particular means (application of knowledge). RST analyses do not depend on "trigger" words in the way that PropBank, NomBank, and FrameNet do. Rather, segments of text are identified

and the relationships between them are then inferred.

Prasad et al. (2008) take a slightly different approach to discourse-level annotation, one that relies heavily on lexical cues to guide the annotation process. The resulting resource, called the Penn Discourse TreeBank (PDTB), identifies RST-like relationships that obtain between large fragments of text. Consider the following example, taken from the PDTB:

(4.14) [$Arg_1$ Use of dispersants was approved] when [$Arg_2$ a test on the third day showed some positive results].

In Example 4.14, I have underlined the lexical item that triggers the *Reason* discourse relationship between the bracketed spans of text. Argument position *when*:$Arg_1$ indicates the effect, and argument position *when*:$Arg_2$ indicates the cause. The latter contains an instance of the nominal predicate *test*, whose $Arg_1$ position (the entity tested) is implicitly filled by *dispersants*. The identification of this implicit argument is encouraged by the discourse connective *when*, which indicates a strong relationship between the events described in its argument positions. In Section 4.4.2, I will explore the use of PDTB relationships for implicit argument identification.

### 4.2.4 Identifying implicit arguments

Past research on the actual task of implicit argument identification tends to be sparse. Palmer et al. (1986) describe what appears to be the first computational treatment of implicit arguments. In this work, Palmer et al. manually created a repository of knowledge concerning entities in the domain of electronic device failures. This knowledge, along with hand-coded syntactic and semantic processing rules, allowed the system to identify implicit arguments across sentence boundaries. As a simple example, consider the following two sentences, borrowed from Palmer et al.:

(4.15) Disk drive was down at 11/16-2305.

(4.16) Has select lock.

Example 4.16 does not specify precisely which entity has select lock. However, the domain knowledge tells the system that only *disk drive* entities can have such a property. Using this knowledge, the system is able to search the local context and make explicit the implied fact that the disk drive from Example 4.15 has select lock.

A similar line of work was pursued by Whittemore et al. (1991), who offer the following example of implicit argumentation (p. 21):

(4.17) Pete bought a car.
(4.18) The salesman was a real jerk.

In Example 4.17, the *buy* event is not associated with an entity representing the seller. This entity is introduced in Example 4.18 as *the salesman*, whose semantic properties satisfy the requirements of the *buy* event. Whittemore et al. build up the event representation incrementally using a combination of semantic property constraints and Discourse Representation Theory.

The systems developed by Palmer et al. and Whittemore et al. are quite similar. They both make use of semantic constraints on arguments, otherwise known as selectional preferences. Selectional preferences have received a significant amount of attention over the years, with the work of Ritter et al. (2010) being some of the most recent. The model developed in the current chapter uses a variety of selectional preference measures to identify implicit arguments.

The implicit argument identification systems described above were not widely deployed due to their reliance on hand-coded, domain-specific knowledge that is difficult to create. Much of this knowledge targeted basic syntactic and semantic constructions that now have robust statistical models (e.g., those created by Charniak and Johnson (2005) for syntax and Punyakanok et al. (2005) for semantics). With this information accounted for, it is easier to approach the problem of implicit argumentation. Below, I describe a series of recent investigations that have led to a surge of interest in statistical implicit argument identification.

Fillmore and Baker (2001) provided a detailed case study of FrameNet frames as a basis for understanding written text (see page 13 for the details of FrameNet). In their case study, Fillmore and Baker manually build up a semantic discourse structure by hooking together frames from the various sentences. In doing so, the authors resolve some implicit arguments found in the discourse. This process is an interesting step forward; however, the authors did not provide concrete methods to perform the analysis automatically.

Nielsen (2004) developed a system that is able to detect the occurrence of verb phrase ellipsis. Consider the following sentences:

(4.19) John kicked the ball.
(4.20) Bill [did], too.

The bracketed text in Example 4.20 is a placeholder for the verb phrase *kicked the ball* in Example 4.20, which has been elided (i.e., left out). Thus, in 4.20, *Bill* can be thought of as an implicit argument to some kicking event that is not mentioned. If one resolved the verb phrase ellipsis, then the implicit argument would be recovered. Nielsen created a system able to detect the presence of ellipses, producing the bracketing in 4.20. Ellipsis resolution (i.e., figuring out precisely which verb phrase is missing) was described by Nielsen (2005). Implicit argument identification for nominal predicates is complementary to verb phrase ellipsis resolution: both work to make implicit information explicit.

Burchardt et al. (2005) suggested that frame elements from various frames in a text could be linked to form a coherent discourse interpretation (this is similar to the idea described by Fillmore and Baker (2001)). The linking operation causes two frame elements to be viewed as coreferent. Burchardt et al. propose to learn frame element linking patterns from observed data; however, the authors did not implement and evaluate such a method. Building on the work of Burchardt et al., this dissertation presents a model of implicit arguments that uses a quantitative analysis of naturally occurring coreference patterns.

The previous chapter, a condensed version of which was published by Gerber et al. (2009), demonstrated the importance of filtering out nominal predicates that take no local

arguments. This approach leads to appreciable gains for certain nominals. However, the approach does not attempt to actually recover implicit arguments.

Most recently, Ruppenhofer et al. (2009) conducted SemEval Task 10, "Linking Events and Their Participants in Discourse", which evaluated implicit argument identification systems over a common test set. The task organizers annotated implicit arguments across entire passages, resulting in data that cover many distinct predicates, each associated with a small number of annotated instances. As described by Ruppenhofer et al. (2010), three submissions were made to the competition, with two of the submissions attempting the implicit argument identification part of the task. Chen et al. (2010) extended a standard SRL system by widening the candidate window to include constituents from other sentences. A small number of features based on the FrameNet frame definitions were extracted for these candidates, and prediction was performed using a log-linear model. Tonelli and Delmonte (2010) also extended a standard SRL system. Both of these systems achieved an implicit argument $F_1$ score of less than 0.02. The organizers and participants appear to agree that training data sparseness was a significant problem. This is likely the result of the annotation methodology: entire documents were annotated, causing each predicate to receive a very small number of annotated examples.

In contrast to the evaluation described by Ruppenhofer et al. (2010), the study presented in this chapter focused on a select group of nominal predicates. To help prevent data sparseness, the size of the group was small, and the predicates were carefully chosen to maximize the observed frequency of implicit argumentation. I annotated a large number of implicit arguments for this group of predicates with the goal of training models that generalize well to the testing data. In the following section, I describe the implicit argument annotation process and resulting dataset.

## 4.3 Empirical analysis

As shown in the previous section, the existence of implicit arguments has been recognized for quite some time. However, this type of information was not formally annotated until Ruppenhofer et al. (2009) conducted their SemEval task on implicit argument identification. There are two reasons why I chose to create an independent dataset for implicit arguments. The first reason is the aforementioned sparsity of the SemEval dataset. The second reason is that the SemEval dataset is not built on top of the Penn TreeBank, which is the gold-standard syntactic base for all work in this dissertation. Working on top of the Penn TreeBank makes the annotations immediately compatible with PropBank, NomBank, and a host of other resources that also build on the TreeBank.

### 4.3.1 Data annotation

#### Predicate selection

Because implicit arguments were a new subject of annotation in the field, it was important to focus in on a select group of nominal predicates. Predicates in this group were required to meet the following criteria:

1. A selected predicate must have an unambiguous role set. This criterion corresponds roughly to an unambiguous semantic sense and is motivated by the need to separate the implicit argument behavior of a predicate from its semantic meaning.

2. A selected predicate must be derived from a verb. This dissertation focuses primarily on the event structure of texts. Nominal predicates derived from verbs denote events, but there are other, non-eventive predicates in NomBank (e.g., the partitive *%*). This criterion also implies that the annotated predicates have correlates in PropBank with semantically compatible role sets.

3. A selected predicate should have a high frequency in the Penn TreeBank corpus. This criterion ensures that the evaluation results say as much as possible about the event structure of the underlying corpus. I calculated frequency with basic counting over morphologically normalized predicates (i.e., *bids* and *bid* are counted as the same predicate).

4. A selected predicate should express many implicit arguments. Of course, this can only be estimated ahead of time because no data exist to compute it. To estimate this value for a predicate $p$, I first calculated $N_p$, the average number of roles expressed by $p$ in NomBank. I then calculated $V_p$, the average number of roles expressed by the verb form of $p$ in PropBank. I hypothesized that the difference $V_p - N_p$ gives an indication of the number of implicit arguments that might be present in the text for a nominal instance of $p$. The motivation for this hypothesis is as follows. Most verbs must be explicitly accompanied by specific arguments in order for the resulting sentence to be grammatical. The following sentences are ungrammatical if the parenthesized portion is left out:

   (4.21) *John loaned (the money to Mary).
   (4.22) *John invested (his money).

   Examples 4.21 and 4.22 indicate that certain arguments must explicitly accompany *loan* and *invest*. In nominal form, these predicates can exist without such arguments and still be grammatical:

   (4.23) John's loan was not repaid.
   (4.24) John's investment was huge.

   Note, however, that Examples 4.23 and 4.24 are not reasonable things to write unless the missing arguments were previously mentioned in the text. This is precisely the type of noun that should be targeted for implicit argument annotation. The value of $V_p - N_p$ thus quantifies the desired behavior.

Predicates were filtered according to criteria 1 and 2 and ranked according to the product of 3 and 4. I then selected the top ten, which are shown in the first column of Table 4.1. The role sets (i.e., argument definitions) for these predicates can be found in Appendix Section A.4 on page 137.

**Annotation procedure**

I annotated implicit arguments for instances of the ten selected nominal predicates. The annotation process proceeded document-by-document. For a document $d$, I annotated implicit arguments as follows:

1. Select from $d$ all non-proper singular and non-proper plural nouns that are morphologically related to the ten predicates in Table 4.1.

2. By design, each selected noun has an unambiguous role set. Thus, given the arguments supplied for a noun by NomBank, one can consult the noun's role set to determine which arguments are missing.[3]

3. For each missing argument position, search the current sentence and all preceding sentences for a suitable implicit argument. Annotate *all* suitable implicit arguments in this window.

4. As often as possible, match the extent of an implicit argument to the extent of an argument given by either PropBank or NomBank. This was done to maintain compatibility with these and other resources.

In the remainder of this dissertation, I will use $iarg_n$ to refer to an implicit argument position $n$. I will use $arg_n$ to refer to an argument provided by PropBank or NomBank. I will use $p$ to mark predicate instances. Below, I give an example annotation for an instance of the *investment* predicate:

[3]See page 137 for the list of role sets used in this study.

| | | | Pre-annotation | | | Post-annotation | |
|---|---|---|---|---|---|---|---|
| | | | | Role avg. (SD) | | | |
| Pred. | # Pred. | # Imp./pred. | Role coverage (%) | Noun | Verb | Role coverage (%) | Noun role avg. (SD) |
| bid | 88 | 1.4 | 26.9 | 0.8 (0.6) | 2.2 (0.6) | 73.9 | 2.2 (0.9) |
| sale | 184 | 1.0 | 24.2 | 1.2 (0.7) | 2.0 (0.7) | 44.0 | 2.2 (0.9) |
| loan | 84 | 1.0 | 22.1 | 1.1 (1.1) | 2.5 (0.5) | 41.7 | 2.1 (1.1) |
| cost | 101 | 0.9 | 26.2 | 1.0 (0.7) | 2.3 (0.5) | 47.5 | 1.9 (0.6) |
| plan | 100 | 0.8 | 30.8 | 1.2 (0.8) | 1.8 (0.4) | 50.0 | 2.0 (0.4) |
| investor | 160 | 0.7 | 35.0 | 1.1 (0.2) | 2.0 (0.7) | 57.5 | 1.7 (0.6) |
| price | 216 | 0.6 | 42.5 | 1.7 (0.5) | 1.7 (0.5) | 58.6 | 2.3 (0.6) |
| loss | 104 | 0.6 | 33.2 | 1.3 (0.9) | 2.0 (0.6) | 48.1 | 1.9 (0.7) |
| investment | 102 | 0.5 | 15.7 | 0.5 (0.7) | 2.0 (0.7) | 33.3 | 1.0 (1.0) |
| fund | 108 | 0.5 | 8.3 | 0.3 (0.7) | 2.0 (0.3) | 21.3 | 0.9 (1.2) |
| Overall | 1,247 | 0.8 | 28.0 | 1.1 (0.8) | 2.0 (0.6) | 47.8 | 1.9 (0.9) |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Table 4.1: Annotation data analysis. Columns are defined as follows: (1) the annotated predicate, (2) the number of predicate instances that were annotated, (3) the average number of implicit arguments per predicate instance, (4) of all roles for all predicate instances, the percentage filled by NomBank arguments, (5) the average number of NomBank arguments per predicate instance, (6) the average number of PropBank arguments per instance of the verb form of the predicate, (7) of all roles for all predicate instances, the percentage filled by either NomBank or implicit arguments, (8) the average number of combined NomBank/implicit arguments per predicate instance. *SD* indicates the standard deviation with respect to an average.

(4.25) [$iarg_0$ Participants] will be able to transfer [$iarg_1$ money] to [$iarg_2$ other investment funds]. The [$p$ investment] choices are limited to [$iarg_2$ a stock fund and a money-market fund].

NomBank does not associate this instance of *investment* with any arguments; however, one can easily identify the investor ($iarg_0$), the thing invested ($iarg_1$), and two mentions of the thing invested in ($iarg_2$) within the surrounding discourse.

Of course, not all implicit argument decisions are as easy as those in Example 4.25. Consider the following contrived example:

(4.26) People in other countries could potentially consume large amounts of [$iarg_0$? Coke].

(4.27) Because of this, there are [$p$ plans] to expand [$iarg_0$ the company's] international presence.

Example 4.27 contains one mention of the $iarg_0$ (the agentive planner). It might be tempting to also mark *Coke* in Example 4.26 as an additional $iarg_0$; however, the only reasonable interpretation of *Coke* in 4.26 is as a consumable fluid. Fluids cannot plan things, so this annotation should not be performed. This is a case of metonymy between *Coke* as a company and *Coke* as a drink. In all such cases, I inspected the implied meaning of the term when deciding whether to apply an implicit argument label.

Lastly, it should be noted that I placed no restrictions on embedded arguments. PropBank and NomBank do not allow argument extents to overlap. Traditional SRL systems such as the one created by Punyakanok et al. (2008) model this constraint explicitly to arrive at the final label assignment; however, as the following example shows, this constraint should not be applied to implicit arguments:

(4.28) Currently, the rules force [$iarg_0$ executives, directors and other corporate insiders] to report purchases and [$p$ sales] [$arg_1$ of [$iarg_0$ their] companies' shares] within about a month after the transaction.

Despite its embedded nature, the pronoun *their* in Example 4.28 is a perfectly reasonable implicit argument (the seller) for the marked predicate. Systems should be required to identify such arguments.

**Inter-annotator agreement**

Implicit argument annotation is a difficult task because it combines the complexities of traditional SRL annotation with those of coreference annotation. To assess the reliability of the annotation process described above, I compared my annotations to those provided by an undergraduate linguistics student who, after a brief training period, re-annotated a portion of the dataset. For each missing argument position, the student was asked to identify the textually closest acceptable implicit argument within the current and preceding sentences. The argument position was left unfilled if no acceptable constituent could be found. For a missing argument position $iarg_n$, the student's annotation agreed with my own if both identified the same implicit argument or both left $iarg_n$ unfilled. The student annotated 480 of the 1,247 predicate instances shown in Table 4.1.

I computed Cohen's chance-corrected kappa statistic for inter-annotator agreement (Cohen, 1960), which is based on two quantities:

$$p_o = \text{observed probability of agreement}$$

$$p_c = \text{probability of agreement by chance}$$

The quantity $1 - p_c$ indicates the probability of a chance disagreement. The quantity $p_o - p_c$ indicates the probability of agreement that cannot be accounted for by chance alone. Finally, Cohen defines $\kappa$ as follows:

$$\kappa = \frac{p_o - p_c}{1 - p_c}$$

Cohen's kappa thus gives the probability that a chance-expected disagreement will not occur. When agreement is perfect, $\kappa = 1$. If the observed agreement is less than the expected chance agreement, then $\kappa$ will be negative. As noted by Di Eugenio and Glass (2004), researchers have devised different scales to assess $\kappa$. Many NLP researchers use the scale created by

Krippendorff (1980):

| | |
|---|---|
| $\kappa < 0.67$ | low agreement |
| $0.67 <= \kappa < 0.8$ | moderate agreement |
| $\kappa >= 0.8$ | strong agreement |

However, Di Eugenio and Glass also note that this scale has not been rigorously defended, even by Krippendorff himself.

For the implicit argument annotation data, observed and chance agreement are defined as follows:

$$p_o = \frac{\sum_{iarg_n} \text{agree}(iarg_n)}{N}$$
$$p_c = \frac{\sum_{iarg_n} P_A(n) * P_B(n) * random\_agree(iarg_n) + (1 - P_A(n)) * (1 - P_B(n))}{N} \quad (4.29)$$

where $N$ is the total number of missing argument positions that need to be annotated, *agree* is equal to 1 if the two annotators agreed on $iarg_n$ and 0 otherwise, $P_A(n)$ and $P_B(n)$ are the observed prior probabilities that annotators $A$ and $B$ assign a label of $n$, and *random_agree* is equal to the probability that both annotators would select the same implicit argument for $iarg_n$ when choosing randomly from the discourse. In Equation 4.29, terms to the right of + denote the probability that the two annotators agreed on $iarg_n$ because they did not identify a filler for it.

Using the above values for $p_o$ and $p_c$, Cohen's kappa indicated an agreement of 64.3%. According to Krippendorff's scale, this value is borderline between low and moderate agreement. Possible causes for this low agreement include the brief training period for the linguistics student and the sheer complexity of the annotation task. If one considers only those argument positions for which both annotators actually located an implicit filler, Co-

hen's kappa indicates an agreement of 93.1%. This shows that much of the disagreement concerned the question of whether a filler was present. Having agreed that a filler was present, the annotators consistently selected the same filler. The student's annotations were only used to compute agreement. I performed all training and evaluation using randomized cross-validation over the annotations I created.

### 4.3.2 Annotation analysis

I carried out the annotation process described above on the standard training (2-21), development (24), and testing (23) sections of the Penn TreeBank. Table 4.1 on page 77 summarizes the results. Below, I highlight key pieces of information found in this table.

#### Implicit arguments are frequent

Column three of Table 4.1 shows that most predicate instances are associated with at least one implicit argument. Implicit arguments vary across predicates, with *bid* exhibiting (on average) more than one implicit argument per instance versus the 0.5 implicit arguments per instance of the *investment* and *fund* predicates. It turned out that the latter two predicates have unique senses that preclude implicit argumentation (more on this in Section 4.6).

#### Implicit arguments create fuller event descriptions

Role coverage for a predicate instance is equal to the number of filled roles divided by the number of roles in the predicate's role set. Role coverage for the marked predicate in Example 4.25 (p. 78) is 0/3 for NomBank-only arguments and 3/3 when the annotated implicit arguments are also considered. Returning to Table 4.1, the fourth column gives role coverage percentages for NomBank-only arguments. The seventh column gives role coverage percentages when both NomBank arguments and the annotated implicit arguments are considered. Overall, the addition of implicit arguments created a 71% relative (20-point absolute) gain in role coverage across the 1,247 predicate instances that I annotated.

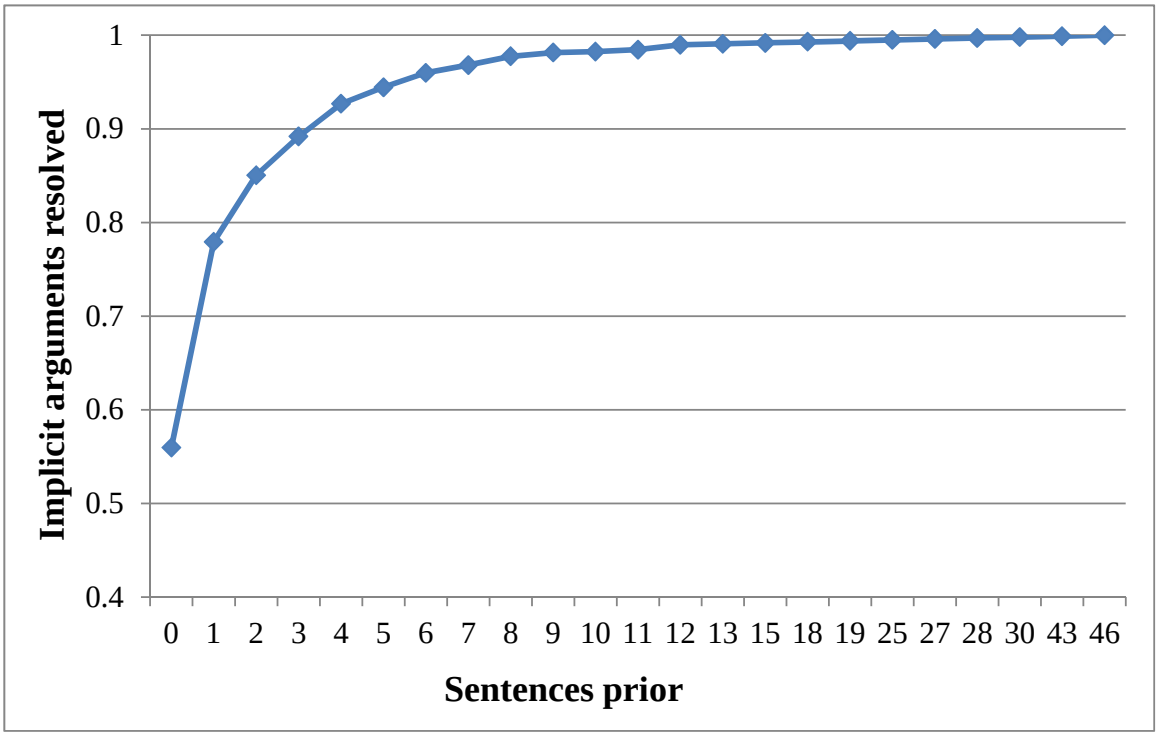



Figure 4.1: Location of implicit arguments. Of all implicitly filled argument positions, the $y$-axis indicates the percentage that are filled at least once within the number of sentences indicated by the $x$-axis (multiple fillers may exist for the same position).

When I introduced the NomBank resource, I observed that approximately 87% of nominal predicate instances do not have a corresponding verbal form present in the document. This indicates that much of the information contained in NomBank is not redundant with PropBank. Because most of the NomBank predicates are novel, the implicit arguments associated with these predicates should also contribute novel information.

### The $V_p - N_p$ predicate selection metric behaves as desired

The predicate selection method used the $V_p - N_p$ metric to identify predicates whose instances are likely to take implicit arguments. Column five in Table 4.1 shows that (on average) nominal predicates have 1.1 arguments in NomBank, this compared to the 2.0 arguments per verbal form of the predicates in PropBank (compare columns five and six). I hypothesized that this difference might indicate the presence of approximately one implicit argument per predicate instance. This hypothesis is confirmed by comparing columns six and eight: when

considering implicit arguments, many nominal predicates express approximately the same number of arguments on average as their verbal counterparts.

**Most implicit arguments are nearby**

In addition to the analyses described above, I examined the location of implicit arguments in the discourse. Figure 4.1 shows that approximately 56% of the implicit arguments in our data can be resolved within the sentence containing the predicate. Approximately 90% are found within the previous three sentences. The remaining implicit arguments require up to forty-six sentences for resolution. These observations are important; they show that searching too far back in the discourse is likely to produce many false positives without a significant increase in recall. Section 4.6 discusses additional implications of this skewed distribution.

## 4.4 Implicit argument model

### 4.4.1 Model formulation

Given a nominal predicate instance $p$ with a missing argument position $iarg_n$, the task is to search the surrounding discourse for a constituent $c$ that fills $iarg_n$. The implicit argument model conducts this search over all constituents that are marked with a core argument label ($arg_0$, $arg_1$, etc.) associated with a NomBank or PropBank predicate. Thus, the model assumes a pipeline organization in which a document is initially analyzed by traditional verbal and nominal SRL systems. The core arguments from this stage then become candidates for implicit argumentation. Adjunct arguments are excluded.

A candidate constituent $c$ will often form a coreference chain with other constituents in the discourse. Consider the following abridged sentences, which are adjacent in their Penn TreeBank document:

(4.30) [Mexico] desperately needs investment.

(4.31) Conservative Japanese investors are put off by [Mexico's] investment regulations.

(4.32) Japan is the fourth largest investor in [$c$ Mexico], with 5% of the total [$p$ investments].

NomBank does not associate the labeled instance of *investment* with any arguments, but it is clear from the surrounding discourse that constituent $c$ (referring to Mexico) is the thing being invested in (the $iarg_2$). When determining whether $c$ is the $iarg_2$ of *investment*, one can draw evidence from other mentions in $c$'s coreference chain. Example 4.30 states that Mexico needs investment. Example 4.31 states that Mexico regulates investment. These propositions, which can be derived via traditional SRL analyses, should increase our confidence that $c$ is the $iarg_2$ of *investment* in Example 4.32.

Thus, the unit of classification for a candidate constituent $c$ is the three-tuple $\langle p, iarg_n, c' \rangle$, where $c'$ is a coreference chain comprising $c$ and its coreferent constituents.[4] I defined a binary classification function $Pr(+|\langle p, iarg_n, c' \rangle)$ that predicts the probability that the entity referred to by $c$ fills the missing argument position $iarg_n$ of predicate instance $p$. In the remainder of this dissertation, I will refer to $c$ as the *primary filler*, differentiating it from other mentions in the coreference chain $c'$. In the following section, I present the feature set used to represent each three-tuple within the classification function.

### 4.4.2 Model features

Table A.4 (p. 140) lists all features used by the model described in this chapter. As shown, these features are quite different from those used in previous work to identify semantic arguments in the traditional nominal SRL setting (see Chapter 2 and Gerber et al. (2009)). This difference is due to the fact that syntactic information - a crucial part of traditional SRL - is not very informative for the implicit argument task, which is primarily a semantic phenomenon. Below, I give detailed explanations for features that are not sufficiently explained in Table A.4.

[4] I used OpenNLP for coreference identification: http://opennlp.sourceforge.net

**Feature 9** captures the semantic relationship between predicate-argument positions by examining paths between frame elements in FrameNet. SemLink[5] maps PropBank argument positions to their FrameNet frame elements. For example, the $arg_1$ position of *sell* maps to the *Goods* frame element of the *Sell* frame. NomBank argument positions (e.g., $arg_1$ of *sale*) can be mapped to FrameNet by first converting the nominal predicate to its verb form. By mapping predicate-argument structures into FrameNet, one can take advantage of the rich network of frame-frame relations provided by the resource (see Section 2.2.2 on page 13 for an example).

The value of Feature 9 has the following general form:

(4.33) $Frame_1.FE_1 \xrightarrow{Relation_1} Frame_2.FE_2 \xrightarrow{Relation_2} \ldots \xrightarrow{Relation_{n-1}} Frame_n.FE_n$

This path describes how the frame elements at either end are related. For example, consider the frame element path between the $arg_1$ of *sell* and the $arg_1$ of *buy*, both of which denote the goods being transferred:

(4.34) Sell.Goods $\xrightarrow{\text{Inherits}}$ Giving.Theme $\xrightarrow{\text{Causes}}$ Getting.Theme $\xrightarrow{\text{Inherited by}}$ Buy.Goods

This path can be paraphrased as follows: things that are sold (Sell.Goods) are part of a more general giving scenario (Giving.Theme) that can also be viewed as a getting scenario (Getting.Theme) in which the buyer receives something (Buy.Goods). This complex world knowledge is represented compactly using the relationships defined in FrameNet. In my experiments, I searched all possible frame element paths of length five or less that use the following relationships:

- Causative-of
- Inchoative-of
- Inherits
- Precedes
- Subframe-of

[5] http://verbs.colorado.edu/semlink

Feature 9 is helpful in situations such as the following (contrived):

(4.35) Consumers bought many [$c$ cars] this year at reduced prices.
(4.36) [$p$ Sales] are expected to drop when the discounts are eliminated.

In Example 4.36 we are looking for the $iarg_1$ (thing sold) of *sale*. The path shown in Example 4.34 indicates quite clearly that the candidate *cars* from Example 4.35, being the entity purchased, is a suitable filler for this position.

Lastly, note that the value for Feature 9 is the actual path instead of a numeric value. When $c$ forms a coreference chain of multiple elements, this feature can be instantiated using multiple values (i.e., paths). Ultimately, these instantiations are binarized into the LibLinear input format, so the existence of multiple feature values does not pose a problem.

**Feature 11** checks whether two predicate-argument positions have the same thematic role and reside in the same VerbNet class. As described in Section 2.2.2 (p. 15), VerbNet is a lexicon of verb classes. Each class contains verbs that are semantically related in addition to a set of thematic roles used for all verbs in the class. The classes are arranged into an inheritance hierarchy. An example class is shown below:

**54.4** appraise, approximate, price ... Thematic roles: Agent, Theme, Value

Here, the dot notation **54.4** should be viewed as a Gorn address (Gorn, 1967) to the verb class within the VerbNet hierarchy (not shown), which is tree-structured. Feature 11 uses these classes to identify implicit arguments in situations such as the following (contrived):

(4.37) John appraised [$c$ the house].
(4.38) [$arg_0$ He] determined a [$p$ price] that was fair.

In Example 4.38 we are looking for the $iarg_1$ (the valued item) of *price*. In VerbNet terms, we are looking for the 54.4.Theme argument. Candidate $c$ is the $arg_1$ of *appraise*, which maps to the VerbNet thematic role 54.4.Theme. A simple identity check allows us to fill the $iarg_1$ of *price* with *the house*. Feature 11 is a boolean feature indicating whether this identify check is satisfied.

**Feature 13** is inspired by the work of Chambers and Jurafsky (2008), who investigated unsupervised learning of narrative event sequences using pointwise mutual information (PMI) between syntactic positions. I extended this PMI score to semantic arguments instead of syntactic dependencies. Thus, the value for this feature is computed as follows:

$$pmi(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) = log\frac{P_{coref\_pmi}(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle)}{P_{coref\_pmi}(\langle p_1, arg_i\rangle, *)P_{coref\_pmi}(\langle p_2, arg_j\rangle, *)} \quad (4.39)$$

I computed Equation 4.39 using carefully selected subsets of the Gigaword corpus (Graff, 2003). I first indexed the entire Gigaword corpus (approximately $10^6$ documents) using the Lucene search engine.[6] I then queried this index using the simple boolean query "$p_1$ AND $p_2$", which retrieved documents relevant to the predicates considered in Equation 4.39. I used the verbal SRL system of Punyakanok et al. (2008) and the nominal SRL system of Gerber et al. (2009) to extract arguments from these documents, and I identified coreferent arguments with OpenNLP. Assuming the resulting data has $N$ coreferential pairs of arguments, the numerator in Equation 4.39 is defined as follows:

$$P_{coref\_pmi}(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) = \frac{\#coref(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle)}{N} \quad (4.40)$$

In Equation 4.40, $\#coref$ returns the number of times the given argument positions are found to be coreferential. In order to penalize low-frequency observations with artificially high scores, I used the simple discounting method described by Pantel and Ravichandran (2004) resulting in the following modification of Equation 4.40:

$$\begin{aligned} x &= \#coref(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) \\ P_{coref\_pmi}(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) &= \frac{x}{N} * \frac{x}{x+1} \end{aligned} \quad (4.41)$$

Thus, if two argument positions are rarely observed as coreferent, the discount factor $\frac{x}{x+1}$

[6] http://lucene.apache.org

| Argument position | #coref with loss.$arg_1$ | Raw PMI score | Discounted PMI score |
|---|---|---|---|
| win.$arg_1$ | 37 | 5.68 | 5.52 |
| gain.$arg_1$ | 10 | 5.13 | 4.64 |
| recoup.$arg_1$ | 2 | 6.99 | 4.27 |
| steal.$arg_1$ | 4 | 5.18 | 4.09 |
| possess.$arg_1$ | 3 | 5.10 | 3.77 |

Table 4.2: Targeted PMI scores between the $arg_1$ of *loss* and other argument positions. The second column gives the number of times that the argument position in the row is found to be coreferent with the $arg_1$ of the *loss* predicate. A higher value in this column results in a lower discount factor. See Equation 4.41 for the discount factor.

will be small, reducing the PMI score. The denominator in Equation 4.39 is computed with a similar discount factor:

$$x_1 = \#coref(\langle p_1, arg_i \rangle, *)$$

$$x_2 = \#coref(\langle p_2, arg_j \rangle, *)$$

$$P_{coref\_pmi}(\langle p_1, arg_i \rangle, *) P_{coref\_pmi}(\langle p_2, arg_j \rangle, *) = \frac{x_1 x_2}{(N^2)\frac{min(x_1,x_2)}{min(x_1,x_2)+1}} \tag{4.42}$$

Thus, if either of the argument positions is rarely observed as coreferent with other argument positions, the discount factor $\frac{min(x_1,x_2)}{min(x_1,x_2)+1}$ will be small, making the denominator of Equation 4.39 large, reducing the PMI score. In general, the discount factors reduce the PMI score for argument positions that are not frequent in the corpus.

I refer to Equation 4.39 as a *targeted* PMI score because it relies on data that have been chosen specifically for the calculation at hand. Table 4.2 shows a sample of targeted PMI scores between the $arg_1$ of *loss* and other argument positions. There are two things to note about this data: first, the argument positions listed are all naturally related to the $arg_1$ of *loss*. Second, the discount factor changes the final ranking by moving the less frequent *recoup* predicate from a raw rank of 1 to a discounted rank of 3, preferring instead the more common *win* predicate.

The information in Table 4.2 is useful in situations such as the following (contrived):

(4.43) Mary won [$c$ the tennis match].
(4.44) [$arg_0$ John's] [$p$ loss] was not surprising.

In Example 4.44 we are looking for the $iarg_1$ of *loss*. The information in Table 4.2 strongly suggests that the marked candidate $c$, being the $arg_1$ of *win*, would be a suitable filler for this position. Lastly, note that if $c$ were to form a coreference chain with other constituents, it would be possible to calculate multiple PMI scores. In such cases, the targeted PMI feature uses the maximum of all scores.

**Feature 23** takes on a value equal to the concatenation of $p.iarg_n$ (the missing argument position) and $p_f.arg_f$, which is the argument position of the candidate $c$. To reduce data sparsity, this feature generalizes predicates and argument positions to their VerbNet classes and thematic roles using SemLink. For example, consider the following Penn TreeBank sentences:

(4.45) [$arg_0$ The two companies] [$p$ produce] [$arg_1$ market pulp, containerboard and white paper]. The goods could be manufactured closer to customers, saving [$p$ shipping] costs.

Here we are trying to fill the $iarg_0$ of *shipping*. Let $c'$ contain a single mention, *The two companies*, which is the $arg_0$ of *produce*. Feature 23 is instantiated with a value of $26.4.Agent \rightarrow 11.1.1.Agent$, where 26.4 and 11.1.1 are the VerbNet classes that contain *produce* and *ship*, respectively. This feature captures general properties of events; in the example above, it describes the tendency of producers to also be shippers. Similarly to other non-numeric features, Feature 23 is instantiated once for each element of $c'$, allowing the model to consider information from multiple mentions of the same entity.

**Feature 27** captures the selectional preference of a predicate $p$ for the elements in $c'$ with respect to argument position $iarg_n$. In general, selectional preference scores denote the strength of attraction for a predicate-argument position to a particular word or class of words. To calculate the value for this feature, I used the information-theoretic model

proposed by Resnik (1996), which is defined as follows:

$$Pref(p, arg_n, s \in \text{WordNet}) = \frac{Pr(s|p, arg_n) log \frac{Pr(s|p, arg_n)}{Pr(s)}}{Z} \quad (4.46)$$
$$Z = \sum_{s_i \in \text{WordNet}} Pr(s_i|p, arg_n) log \frac{Pr(s_i|p, arg_n)}{Pr(s_i)}$$

In Equation 4.46, $Pref$ calculates the preference for a WordNet synset $s$ in the given predicate-argument position. Prior and posterior probabilities for $s$ were calculated by examining the arguments present in the Penn TreeBank combined with 20,000 documents randomly selected from the Gigaword corpus. PropBank and NomBank supplied arguments for the Penn TreeBank, and I used the aforementioned verbal and nominal SRL systems to extract arguments from Gigaword. The head word for each argument was mapped to its WordNet synsets, and counts for these synsets were updated as suggested by Resnik. Two things should be noted about Equation 4.46. First, *WordNet* contains only those synsets that were observed in the training data. Second, the equation is defined as zero for any synset $s$ that is in WordNet but not observed in the training data.

Equation 4.46 computes the preference of a predicate-argument position for a synset; however, a single word can map to multiple synsets if its sense is ambiguous. Given a word $w$ and its synsets $s_1, s_2, \ldots, s_n$, the preference of a predicate-argument position for $w$ is defined as follows:

$$Pref(p, arg_n, w) = \frac{\sum_{s_i} Pref(p, arg_n, s_i)}{n} \quad (4.47)$$

That is, the preference for a word is computed as the average preference across all possible synsets. The final value for Feature 23 is computed using the word-based preference score defined in Equation 4.47. Given a candidate implicit argument $c'$ comprising the primary

| Argument | Raw coreference probability | Discounted coreference probability |
|---|---|---|
| rethink.$arg_1$ | $3/6 = 0.5$ | 0.32 |
| define.$arg_1$ | $2/6 = 0.33$ | 0.19 |
| redefine.$arg_1$ | $1/6 = 0.17$ | 0.07 |

Table 4.3: Coreference probabilities between *reassess.*$arg_1$ and other argument positions. See Equation 4.49 for details on the discount factor.

filler $c$ and its coreferent mentions, the following value is obtained:

$$Pref(p, iarg_n, c') = \min_{f \in c'} Pref(p, arg_n, f) \tag{4.48}$$

In Equation 4.48, each $f$ is the syntactic head of a constituent from $c'$. The value of Equation 4.48 is in $(-\infty, +\infty)$, with larger values indicating higher preference for $c$ as the implicit filler of position $iarg_n$.

**Feature 33** implements the suggestion of Burchardt et al. (2005) that implicit arguments might be identified using observed coreference patterns in a large corpus of text. My implementation of this feature uses the same data used for the previous feature: arguments extracted from the Penn TreeBank and 20,000 documents randomly selected from Gigaword. Additionally, I identified coreferent arguments in this corpus using OpenNLP. Using this information, I calculated the probability of coreference between any two argument positions. As with Feature 13, I used discounting to penalize low-frequency observations, producing an estimate of coreference probability as follows:

$$\begin{aligned} Coref_{joint} &= \#coref(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle) \\ Coref_{marginal} &= \#coref(\langle p_1, arg_i \rangle, *) \\ P_{coref}(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle) &= \frac{Coref_{joint}}{Coref_{marginal}} * \frac{Coref_{joint}}{Coref_{joint}+1} * \frac{Coref_{marginal}}{Coref_{marginal}+1} \end{aligned} \tag{4.49}$$

For example, I observed that the $arg_1$ for predicate *reassess* (the entity reassessed) is coreferential with six other constituents in the corpus. Table 4.3 lists the argument positions with which this argument is coreferential along with the raw and discounted probabilities. The discounted probabilities can help identify the implicit argument in the following contrived examples:

(4.50) Senators must rethink [$c$ their strategy for the upcoming election].
(4.51) The [$p$ reassessment] must begin soon.

In Example 4.51 we are looking for the $iarg_1$ of *reassess*. Table 4.3 tells us that the marked candidate - an $arg_1$ to *rethink* - is likely to fill this missing argument position. When $c$ forms a coreference chain with other constituents, this feature uses the minimum coreference probability between the implicit argument position and elements in the chain.

**Feature 59** is similar to Feature 9 (the frame element path) except that it captures the distance of the relationship between predicate-argument positions. Consider the following VerbNet classes:

**13.2** lose, refer, relinquish, remit, resign, restore, gift, hand out, pass out, shell out
**13.5.1.1** earn, fetch, cash, gain, get, save, score, secure, steal

The path from *earn* to *lose* in the VerbNet hierarchy is as follows:

(4.52) 13.5.1.1 $\uparrow$ 13.5.1 $\uparrow$ 13.5 $\uparrow$ 13 $\downarrow$ 13.2

The path in Example 4.52 is four links long.

Intuitively, *earn* and *lose* are related to each other - they describe two possible outcomes of a financial transaction. The VerbNet path quantifies this intuition, with shorter paths indicating closer relationships. This information can be used to identify implicit arguments in situations such as the following from the Penn TreeBank (abridged):

(4.53) [$c$ Monsanto Co.] is expected to continue reporting higher [$p$ earnings].
(4.54) The St. Louis-based company is expected to report that [$p$ losses] are narrowing.

In Example 4.54 we are looking for the $iarg_0$ (i.e., entity losing something) for the *loss* predicate. According to SemLink, this argument position maps to the 13.2.Agent role in VerbNet. In Example 4.53, we find the candidate implicit argument *Monsanto Co.*, which is the $arg_0$ to the *earning* predicate in that sentence. This argument position maps to the 13.5.1.1.Agent role in VerbNet. These two VerbNet roles are related according to the VerbNet path in Example 4.52, producing a value for Feature 59 of four. This relatively small value supports an inference of *Monsanto Co.* as the $iarg_0$ for *loss*.

It is important to note that a VerbNet path only exists when the thematic roles are identical. For example, a VerbNet path would not exist between 13.5.1.1.Theme and 13.2.Agent because the roles are not compatible. Lastly, note that $c$ might form a coreference chain $c'$ with multiple elements. In such a situation, the minimum path length is selected as the value for this feature.

**Feature 67** identifies the discourse relation (if any) that holds between the candidate constituent $c$ and the filled predicate $p$. Consider the following example:

(4.55) [$iarg_0$ SFE Technologies] reported a net loss of $889,000 on sales of $23.4 million.

(4.56) That compared with an operating [$p$ loss] of [$arg_1$ $1.9 million] on sales of $27.4 million in the year-earlier period.

In this case, a *comparison* discourse relation (signaled by the underlined text) holds between the first and sentence sentence. The coherence provided by this relation encourages an inference that identifies the marked $iarg_0$ (the loser). The value for this feature is the name of the discourse relation (e.g., *comparison*) whose two discourse units cover the candidate ($iarg_0$ above) and filled predicate ($p$ above). Throughout my investigation, I used gold-standard discourse relations provided by the Penn Discourse TreeBank (Prasad et al., 2008).

**Filler-independent features** are those that do not depend on elements of $c'$. These features are usually specific to a particular predicate. Consider the following example:

(4.57) Statistics Canada reported that its [$arg_1$ industrial-product] [$p$ price] index dropped 2% in September.

The "[$p$ price] index" collocation is rarely associated with an $arg_0$ in NomBank or with an $iarg_0$ in the annotated data (both argument positions denote the seller). Feature 25 accounts for this type of behavior by encoding the syntactic head of $p$'s right sibling. The value of Feature 25 for Example 4.57 is *price:index*. Contrast this with the following:

(4.58) [$iarg_0$ The company] is trying to prevent further [$p$ price] drops.

The value of Feature 25 for Example 4.58 is *price:drop*. This feature captures an important distinction between the two uses of *price*: the former cannot easily take an $iarg_0$, whereas the latter can. Many other features in Table A.4 depend only on the predicate and have values that take the form *predicate:feature_value*.

### 4.4.3 Post-processing for final output selection

Without loss of generality, assume there exists a predicate instance $p$ with two missing argument positions $iarg_0$ and $iarg_1$. Also assume that there are three candidate fillers $c_1$, $c_2$, and $c_3$ within the candidate window. The discriminative model will calculate the probability that each candidate fills each missing argument position. This is depicted graphically below:

| | $iarg_0$ | $iarg_1$ |
|---|---|---|
| $c_1$ | 0.3 | 0.4 |
| $c_2$ | 0.1 | 0.05 |
| $c_3$ | 0.6 | 0.5 |

There exist two constraints on possible assignments of candidates to positions. First, a candidate may not be assigned to more than one missing argument position. To enforce this constraint, only the top-scoring cell in each row is retained, leading to the following:

| | $iarg_0$ | $iarg_1$ |
|---|---|---|
| $c_1$ | - | 0.4 |
| $c_2$ | 0.1 | - |
| $c_3$ | 0.6 | - |

Second, a missing argument position can only be filled by a single candidate. To enforce this constraint, only the top-scoring cell in each column is retained, leading to the following:

| | $iarg_0$ | $iarg_1$ |
|---|---|---|
| $c_1$ | - | 0.4 |
| $c_2$ | - | - |
| $c_3$ | 0.6 | - |

Having satisfied these constraints, a threshold $t$ is imposed on the remaining cell probabilities.[7] Cells with probabilities below $t$ are cleared. Assuming that $t = 0.42$, the final assignment would be as follows:

| | $iarg_0$ | $iarg_1$ |
|---|---|---|
| $c_1$ | - | - |
| $c_2$ | - | - |
| $c_3$ | 0.6 | - |

In this case, $c_3$ fills $iarg_0$ with probability 0.6 and $iarg_1$ remains unfilled. The latter outcome is desirable because not all argument positions have fillers that are present in the discourse.

### 4.4.4 Computational complexity

In a practical setting, the implicit argument model described above would take as input the output of the end-to-end nominal SRL system described in Chapter 2. Thus, the implicit argument model has a best-case performance of $O(s^3)$ owing to the cubic time syntactic parser. This case is achieved when the linear time predicate identifier does not identify any predicate nodes. In this case, there are no argument nodes to be used as candidate implicit arguments, and the implicit argument model will not perform any work in addition to that performed by the nominal SRL system. The best case, however, is rare. Sentences from documents in the news genre almost always have predicates and arguments. Thus, the best case $O(s^3)$ is not likely to be very informative.

In general, it is difficult to characterize the performance of the implicit argument model

[7] The threshold $t$ is learned from the training data. The learning mechanism is explained in the following section.

accurately. This is largely because many of the features used in the model required a significant amount of data pre-processing before the evaluation experiments were conducted (see, for example, Feature 13 on page 84). In a practical setting, this pre-processing would not be possible and its associated cost would need to be incorporated into the runtime analysis. This dissertation leaves an examination of this issue to future work.

## 4.5 Evaluation

#### Data

All evaluations in this chapter were performed using a randomized cross-validation configuration. The 1,247 predicate instances were annotated document by document. In order to remove any confounding factors caused by specific documents, I first randomized the annotated predicate instances. Following this, I split the predicate instances evenly into ten folds and used each fold as testing data for a model trained on the instances outside the fold.

During training, the system was provided with annotated predicate instances. The system identified missing argument positions and generated a set of candidates for each such position. A candidate three-tuple $\langle p, iarg_n, c' \rangle$ was given a positive label if the candidate implicit argument $c$ (the primary filler) was annotated as filling the missing argument position. During testing, the system was presented with each predicate instance and was required to identify all implicit arguments for the predicate.

Throughout the evaluation process I assumed the existence of gold-standard PropBank and NomBank information in all documents. This factored out errors from traditional SRL and affected the following stages of system operation:

- **Missing argument identification.** The system was required to figure out which argument positions were missing. Each of the ten predicates was associated with an unambiguous role set, so determining the missing argument positions amounted to comparing the existing local arguments with the argument positions listed in the

predicate's role set. Because gold-standard local NomBank arguments were used, this stage produced no errors.

- **Candidate generation.** As mentioned in Section 4.4.1, the set of candidates for a missing argument position contains constituents labeled with a core PropBank or NomBank argument label. Gold-standard PropBank and NomBank arguments were used; however, it is not the case that all annotated implicit arguments are given a label by PropBank or NomBank. Thus, despite the gold-standard argument labels, this stage produced errors in which the system failed to generate a true-positive candidate for an implicit argument position.

- **Feature extraction.** Many of the features described in Section 4.4.2 rely on underlying PropBank and NomBank argument labels. For example, the top-ranked Feature 1 relates the argument position of the candidate to the missing argument position. In my experiments, values for this feature contained no errors because gold-standard PropBank and NomBank labels were used. Note, however, that features such as Feature 13 were calculated using the output of an automatic SRL process that occasionally produces errors.

For simplicity, I also assumed the existence of gold-standard syntactic structure when possible. Because most of the features used for implicit argument identification are semantic, this assumption is not likely to have a significant impact on end performance.

#### Scoring metrics

I evaluated system performance using the methodology proposed by Ruppenhofer et al. (2009). For each missing argument position of a predicate instance, the system was required to either (1) identify a single constituent that fills the missing argument position or (2) make no prediction and leave the missing argument position unfilled. I scored predictions using

the Dice coefficient, which is defined as follows:

$$Dice(Predicted, True) = \frac{2 * |Predicted \bigcap True|}{|Predicted| + |True|} \tag{4.59}$$

*Predicted* is the set of tokens subsumed by the constituent predicted by the model as filling a missing argument position. $True$ is the set of tokens from a single annotated constituent that fills the missing argument position. The model's prediction receives a score equal to the maximum Dice overlap across any one of the annotated fillers ($AF$):

$$Score(Predicted) = \max_{True \in AF} Dice(Predicted, True) \tag{4.60}$$

Precision is equal to the summed prediction scores divided by the number of argument positions filled by the model. Recall is equal to the summed prediction scores divided by the number of argument positions filled in the annotated data. Predictions not covering the head of a true filler were assigned a score of zero. For example, consider the following true and predicted labelings:

(4.61) **True labeling:** [$iarg_0$ Participants] will be able to transfer [$iarg_1$ money] to [$iarg_2$ other investment funds]. The [$p$ investment] choices are limited to [$iarg_2$ a stock fund and a money-market fund].

(4.62) **Predicted labeling:** Participants will be able to transfer [$iarg_1$ money] to other [$iarg_2$ investment funds]. The [$p$ investment] choices are limited to a stock fund and a money-market fund.

In the ground-truth (4.61) there are three implicit argument positions to fill. The hypothetical system has made predictions for two of the positions. The prediction scores are shown

below:

$$\begin{aligned} Score(iarg_1 \text{ money}) &= Dice(\text{money}, \text{money}) = 1 \\ Score(iarg_2 \text{ investment funds}) &= \max\{Dice(\text{investment funds}, \text{other investment funds}), \\ &\qquad Dice(\text{investment funds}, \text{a stock} \ldots \text{money-market fund})\} \\ &= \max\{0.8, 0\} = 0.8 \end{aligned}$$

Precision, recall, and $F_1$ for the example predicate are calculated as follows:

$$\begin{aligned} Precision &= \frac{1.8}{2} = 0.9 \\ Recall &= \frac{1.8}{3} = 0.6 \\ F_1 &= \frac{2 * Precision * Recall}{Precision + Recall} = 0.72 \end{aligned}$$

I calculated the $F_1$ score for the entire testing fold by aggregating the counts used in the above precision and recall calculations. Similarly, I aggregated the counts across all folds to arrive at a single $F_1$ score for the evaluated system.

I used a bootstrap resampling technique similar to those developed by Efron and Tibshirani (1993) to test the significance of the performance difference between various systems. Given a test pool comprising $M$ missing argument positions $iarg_n$ along with the predictions by systems $A$ and $B$ for each $iarg_n$, I calculated the exact $p$-value of the performance difference as follows:

1. Create $r$ random resamples from $M$ with replacement.

2. For each resample $R_i$, compute the system performance difference $d_{R_i} = A_{R_i} - B_{R_i}$ and store $d_{r_i}$ in $D$.

3. Find the largest symmetric interval $[min, max]$ around the mean of $D$ that does not include zero.

4. The exact $p$-value equals the percentage of elements in $D$ that are not in $[min, max]$.

Experiments have shown that this simple approach provides accurate estimates of significance while making minimal assumptions about the underlying data distribution (Efron and Tibshirani, 1993). Similar randomization tests have been used to evaluate information extraction systems (Chinchor et al., 1993).

### LibLinear model configuration

Given a testing fold $F_{test}$ and a training fold $F_{train}$, I performed feature selection using only the information contained in $F_{train}$.[8] As part of the feature selection process, I conducted a grid search for the best $c$ and $w$ LibLinear parameters, which govern the per-class cost of mislabeling instances from a particular class (Fan et al., 2008). Setting per-class costs helps counter the effects of class size imbalance, which is severe even when selecting candidates from the current and previous few sentences (most candidates are negative). I ran the feature selection and grid search processes independently for each $F_{train}$. As a result, the feature set and model parameters are slightly different for each fold.[9] For all folds, I used LibLinear's logistic regression solver and a candidate selection window of two sentences prior. As shown in Figure 4.1 (p. 82), this window imposes a recall upper bound of approximately 85%. The post-processing prediction threshold $t$ was learned using a brute-force search that maximized the system's performance over the data in $F_{train}$.

### Baseline and oracle models

I compared the supervised model described above with the simple baseline heuristic defined below:

> Fill $iarg_n$ for predicate instance $p$ with the nearest constituent in the two-sentence candidate window that fills $arg_n$ for a different instance of $p$, where all nominal

[8] See Appendix Section A.8 (p. 144) for the feature selection algorithm.

[9] See Table A.5 (p. 141) for a per-fold listing of features and model parameters.

predicates are normalized to their verbal forms.

The normalization allows, for example, an existing $arg_0$ for the verb *invested* to fill an $iarg_0$ for the noun *investment*. This heuristic outperformed a more complicated heuristic that relied on the PMI score described in Section 4.4.2. I also evaluated an oracle model that made gold-standard predictions for candidates within the two-sentence prediction window.

**Results**

Table 4.4 presents the evaluation results for implicit argument identification. As column two shows, the systems were tested over 966 missing argument positions for which at least one true implicit filler existed. Overall, the discriminative model increased $F_1$ performance by 21.4 points (74.1%) compared to the baseline ($p<0.0001$). Predicates with the highest number of implicit arguments - *sale* and *price* - showed $F_1$ increases of 13.7 points and 17.5 points, respectively ($p<0.001$ for both differences). As expected, oracle precision is 100% for all predictions, and the $F_1$ difference between the discriminative and oracle systems is significant at $p<0.0001$ for all test sets. See Appendix Section A.6 (p. 141) for a per-fold breakdown of results and a listing of features and model parameters used for each fold.

I also measured human performance on this task by running the undergraduate assistant's annotations against a small portion of the evaluation data comprising 275 filled implicit arguments. The assistant achieved an overall $F_1$ score of 56.0% using the same two-sentence candidate window used by the baseline, discriminative, and oracle models. Using an infinite candidate window, the assistant increased $F_1$ performance to 64.2%. Although these results provide a general idea about the performance upper bound, they are not directly comparable to the cross-validated results shown in Table 4.4.

| | | Baseline | | | Discriminative | | | | Oracle | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | # Imp. args. | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ | $p_{exact}$(B,D) | $P$ | $R$ | $F_1$ | $p_{exact}$(D,O) |
| sale | 181 | 57.0 | 27.7 | 37.3 | 59.2 | 44.8 | 51.0 | 0.0003 | 100.0 | 72.4 | 84.0 | <0.0001 |
| price | 138 | 67.1 | 23.3 | 34.6 | 56.0 | 48.7 | 52.1 | <0.0001 | 100.0 | 78.3 | 87.8 | <0.0001 |
| bid | 124 | 66.7 | 14.5 | 23.8 | 60.0 | 36.3 | 45.2 | <0.0001 | 100.0 | 60.5 | 75.4 | <0.0001 |
| investor | 108 | 30.0 | 2.8 | 5.1 | 46.7 | 39.8 | 43.0 | <0.0001 | 100.0 | 84.3 | 91.5 | <0.0001 |
| cost | 86 | 60.0 | 10.5 | 17.8 | 62.5 | 50.9 | 56.1 | <0.0001 | 100.0 | 86.0 | 92.5 | <0.0001 |
| loan | 82 | 63.0 | 20.7 | 31.2 | 67.2 | 50.0 | 57.3 | <0.0001 | 100.0 | 89.0 | 94.2 | <0.0001 |
| plan | 77 | 72.7 | 20.8 | 32.3 | 59.6 | 44.1 | 50.7 | 0.0032 | 100.0 | 87.0 | 93.1 | <0.0001 |
| loss | 62 | 78.8 | 41.9 | 54.7 | 72.5 | 59.7 | 65.5 | 0.0331 | 100.0 | 88.7 | 94.0 | <0.0001 |
| fund | 56 | 66.7 | 10.7 | 18.5 | 80.0 | 35.7 | 49.4 | <0.0001 | 100.0 | 66.1 | 79.6 | <0.0001 |
| investment | 52 | 28.9 | 10.6 | 15.5 | 32.9 | 34.2 | 33.6 | 0.0043 | 100.0 | 80.8 | 89.4 | <0.0001 |
| Overall | 966 | 61.4 | 18.9 | 28.9 | 57.9 | 44.5 | 50.3 | <0.0001 | 100.0 | 78.0 | 87.6 | <0.0001 |

Table 4.4: Overall evaluation results for implicit argument identification. The second column gives the number of ground-truth implicitly filled argument positions for the predicate instances. $P$, $R$, and $F_1$ indicate precision, recall, and F-measure ($\beta = 1$), respectively. $p_{exact}$ is the bootstrapped exact $p$-value of the $F_1$ difference between two systems, where the systems are (B)aseline, (D)iscriminative, and (O)racle.

## 4.6 Discussion

### 4.6.1 Feature assessment

Previously, we assessed the importance of various implicit argument feature groups by conducting feature ablation tests (Gerber and Chai, 2010). In each test, the discriminative model was retrained and reevaluated without a particular group of features. I summarize the findings of this study below:

**Semantic roles are essential.** We observed statistically significant losses when excluding features that relate the semantic roles of elements in $c'$ to the semantic role of the missing argument position. For example, Feature 1 appears as the top-ranked feature in eight out of ten fold evaluations (see Table A.5 on page 141). This feature is formed by concatenating the filling predicate-argument position with the filled predicate-argument position, producing values such as $invest.arg_0$-$lose.arg_0$. This value indicates that the entity performing the investing is also the entity losing something. This type of commonsense knowledge is essential to the task of implicit argument identification.

**Other information is important.** Our 2010 study also found that semantic roles are only one part of the solution. Using semantic roles in isolation also produced statistically significant losses. This indicates that other features contribute useful information to the task.

**Discourse structure is not essential.** We also tested the effect of removing discourse relations (Feature 67) from the model. Discourse structure has received a significant amount of attention in NLP; however, it remains a very challenging problem, with state-of-the-art systems attaining $F_1$ scores in the mid-40% range (Sagae, 2009). Our 2010 work as well as the updated work presented in this dissertation used gold-standard discourse relations from the Penn Discourse TreeBank. As shown by Sagae, these

| # | Description | % |
|---|---|---|
| 1 | A true filler was classified but an incorrect filler scored higher | 30.6 |
| 2 | A true filler did not exist but a prediction was made | 22.4 |
| 3 | A true filler existed within the window but was not classified | 21.1 |
| 4 | A true filler scored highest but below threshold | 15.9 |
| 5 | A true filler existed but not within the window | 10.0 |

Table 4.5: Implicit argument identification error analysis. The second column indicates the type of error that was made and the third column gives the percentage of all errors that fall into each type.

relations are difficult to extract in a practical setting. In our 2010 work, we showed that removing discourse relations from the model did not have a statistically significant effect on performance. Thus, this information should be removed in practical applications of the model, at least until better uses for it can be identified.

To further assess the relative importance of features used in this dissertation, I aggregated the feature rank information given in Table A.5 (p. 141). For each evaluation fold, each feature received a point value equal to its reciprocal rank within the feature list. Thus, a feature appearing at rank 5 for a fold would receive $\frac{1}{5} = 0.2$ points for that fold. I totaled these points across all folds, arriving at the values shown in the final column of Table A.4 (p. 140). The scores confirm the findings described above. The highest scoring feature relates the semantic roles of the candidate argument to the missing argument position. Non-semantic information such as the sentence distance (Feature 2) also plays a key role. Discourse structure is consistently ranked near the bottom of the list (Feature 67).

### 4.6.2 Error analysis

Table 4.5 lists the errors made by the system and their frequencies. As shown, the single most common error (type 1) occurred when a true filler was classified but an incorrect filler had a higher score. This occurred in approximately 31% of the error cases. Often, though, the system did not classify a true implicit argument because such a candidate was

not generated. Without such a candidate, the system stood no chance of making a correct prediction. Errors 3 and 5 combined (also 31%) describe this behavior. Type 3 errors resulted when implicit arguments were not core (i.e., $arg_n$) arguments to other predicates. To reduce class imbalance, the system only used core arguments as candidates; however, this came at the expense of increased type 3 errors. In many cases, the true implicit argument filled a non-core (i.e., adjunct) role within PropBank or NomBank.

Type 5 errors resulted when the true implicit arguments for a predicate were outside the candidate window. Oracle recall (see Table 4.4) indicates the nominals that suffered most from windowing errors. For example, the *sale* predicate was associated with the highest number of true implicit arguments, but only 72% of those could be resolved within the two-sentence candidate window. Empirically, I found that extending the candidate window uniformly for all predicates did not increase $F_1$ performance because additional false positives were identified. The oracle results suggest that predicate-specific window settings might offer some advantage for predicates such as *fund* and *bid*, which take arguments at longer ranges.

Error types 2 and 4 are directly related to the prediction confidence threshold $t$. The former would be reduced by increasing $t$ and thus filtering out bad predictions. The latter would be reduced by lowering $t$ and allowing more true fillers into the final output. However, it is unclear whether either of these actions would increase overall performance.

### 4.6.3 The *investment* and *fund* predicates

In Section 4.4.2, I discussed the *price* predicate, which frequently occurs in the "[$p$ price] index" collocation. I observed that this collocation is rarely associated with either an overt $arg_0$ or an implicit $iarg_0$. Similar observations can be made for the *investment* and *fund* predicates. Although these two predicates are frequent, they are rarely associated with implicit arguments: *investment* takes only 52 implicit arguments and *fund* takes only 56 implicit arguments (see Table 4.4). This behavior is due in large part to collocations such as "[$p$ investment] banker", "stock [$p$ fund]", and "mutual [$p$ fund]", which use predicate senses

that are not eventive and take no arguments. Such collocations also violate the assumption that differences between the PropBank and NomBank argument structure for a predicate are indicative of implicit arguments (see Section 4.3.1 for this assumption).

Despite their lack of implicit arguments, it is important to account for predicates such as *investment* and *fund* because the incorrect prediction of implicit arguments for them can lower precision. This is precisely what happened for the *investment* predicate ($P$ = 33%). The model incorrectly identified many implicit arguments for instances such as "[$p$ investment] banker" and "[$p$ investment] professional", which take no arguments. The right context of *investment* should help the model avoid this type of error; however in many cases this was not enough evidence to prevent a false positive prediction. Additional investigation is needed to address this type of error.

### 4.6.4 Improvements versus the baseline

The baseline heuristic covers the simple case where identical predicates share arguments in the same position. Because the discriminative model also uses this information (see Feature 8), it is interesting to examine cases where the baseline heuristic failed but the discriminative model succeeded. Such cases represent more difficult inferences. Consider the following sentence:

(4.63) Mr. Rogers recommends that [$p$ investors] sell [$iarg_2$ takeover-related stock].

Neither NomBank nor the baseline heuristic associate the marked predicate in Example 4.63 with any arguments; however, the feature-based model was able to correctly identify the marked $iarg_2$ as the entity being invested in. This inference relied on a number of features that connect the *invest* event to the *sell* event (e.g., Features 1, 4, and 76). These features captured a tendency of investors to sell the things they have invested in.

I conclude my discussion with an example of a complex extra-sentential implicit argument:

(4.64) [$arg_0$ Olivetti] [$p$ exported] $25 million in "embargoed, state-of-the-art, flexible manufacturing systems to the Soviet aviation industry."

(4.65) [$arg_0$ Olivetti] reportedly began [$p$ shipping] these tools in 1984.

(4.66) [$iarg_0$ Olivetti] has denied that it violated the rules, asserting that the shipments were properly licensed.

(4.67) However, the legality of these [$p$ sales] is still an open question.

In Example 4.67, we are looking for the $iarg_0$ of *sale*. As shown, the discriminative model was able to correctly identify *Olivetti* from 4.66 as the implied filler of this argument position. The inference involved two key steps. First, the model identified coreferent mentions of *Olivetti* in 4.64 and 4.65. In these sentences, *Olivetti* participates in the marked exporting and shipping events. Second, the model identified a tendency for exporters and shippers to also be sellers (e.g., Features 1, 4, and 23 made large contributions to the prediction). Using this knowledge, the system extracted information that could not be extracted by the baseline heuristic or a traditional SRL system.

### 4.6.5 Comparison with previous results

In a previous study, we reported results similar to those in this chapter (Gerber and Chai, 2010). The key difference between the two is cross-validation, which was not used in our 2010 study. Our 2010 study used fixed partitions of training, development, and testing data. As a result, feature and model parameter selections overfit the development data; we observed a 23-point difference in $F_1$ between the development (65%) and testing (42%) partitions. The small size of the testing set also led to small sample sizes and large $p$-values during significance testing. The cross-validated approach reported in this chapter alleviated both problems. The $F_1$ difference between training and testing was approximately 10 points for all folds, and all of the data were used for testing, leading to more accurate $p$-values. It is not possible to directly compare the evaluation scores in the two studies; however, the methodology in the current chapter is preferable for the reasons mentioned.

## 4.7 Conclusions

Chapter 3 provided a partial solution to the problem of nominals with implicit arguments. The model described in that chapter is able to accurately identify nominals whose arguments are implicit using a variety of lexical and syntactic features. This increases performance by reducing the number of false positive argument predictions; however, all implicit arguments remain unidentified, leaving a large portion of the corresponding event structures unrecognized.

This chapter has presented a detailed study of implicit arguments for a select group of nominal predicates. The study was based on a manually created corpus of implicit arguments, which is freely available for research purposes. The study's primary findings include the following:

1. **Implicit arguments are frequent.** Given the predicates in a document, there exist a fixed number of possible arguments that can be filled according to NomBank's predicate role sets. Role coverage is defined as the fraction of these roles that are actually filled by constituents in the text. Using NomBank as a baseline, the study found that role coverage increases by 71% when implicit arguments are taken into consideration.

2. **Implicit arguments can be automatically identified.** Using the annotated data, I constructed a feature-based supervised model that is able to automatically identify implicit arguments. This model relies heavily on the traditional, single-sentence SRL structure of both nominal and verbal predicates. By unifying these sources of information, the implicit argument model provides a more coherent picture of discourse semantics than is typical in most recent work (e.g., the evaluation conducted by Surdeanu et al. (2008)). The model demonstrates substantial gains over an informed baseline, reaching an overall $F_1$ score of 50% and per-predicate scores in the mid-50s and mid-60s. These results are among the first for this task.

3. **Much work remains.** The study presented in the current chapter was very focused: only ten different predicates were analyzed. The goal was to carefully examine the underlying linguistic properties of implicit arguments. This examination produced many features that have not been used in other SRL studies. The results are encouraging; however, a direct application of the model to all NomBank predicates will require a substantial annotation effort. This is because many of the most important features are lexicalized on the predicate being analyzed and thus cannot be generalized to novel predicates. Additional information might be extracted from VerbNet, which groups related verbs together. Features from this resource might generalize better because they apply to entire sets of verbs.

Lastly, it should be noted that the prediction model described in this chapter is quite simple. Each candidate is independently classified as filling each missing argument position, and a heuristic post-processing step is performed to arrive at the final labeling. This approach ignores the joint behavior of semantic arguments. In the next chapter, I describe a preliminary joint model for implicit arguments that is based on a large-scale knowledge based extracted from Internet webpages.

# CHAPTER 5

# An exploration of TextRunner for joint implicit argument identification

## 5.1 Introduction

The model described in the previous chapter uses a wide variety of lexical and semantic features to make binary implicit argument predictions for constituents in the surrounding discourse. For a predicate instance $p$, each candidate constituent is classified as filling each missing argument position. A heuristic post-processing procedure is then applied to arrive at the final argument structure. With the exception of this final step, the candidates and argument positions are assumed to be independent.

It is easy to construct examples that violate the assumption of independent arguments. Consider the following sentences:

(5.1) [$c_1$ The president] is currently struggling to manage [$c_2$ the country's economy].
(5.2) If he cannot get it under control, [$p$ loss] of [$arg_1$ the next election] might result.

In Example 5.2, we are searching for the $iarg_0$ of *loss* (the entity that is losing). The sentence in 5.1 supplies two reasonable candidates for this position: $c_1$ and $c_2$. If one only considers the predicate *loss*, then $c_1$ and $c_2$ would appear to be equally likely: presidents often lose things (e.g., votes and allegiance) and economies often lose things (e.g., jobs and value). However, the sentence in 5.2 supplies additional information. It tells the reader that *the next election* is the entity being lost. Given this information, one would likely prefer $c_1$ over $c_2$ because economies don't generally lose elections, whereas presidents often do. This type of inference is common in textual discourses because authors assume a shared knowledge

base with their readers. This knowledge base contains information about events and their typical participants (e.g., the fact that presidents lose elections but economies do not).

Inspired by the above observations, this chapter presents a preliminary exploration of the interaction that occurs between implicit arguments. The interaction (or joint) model relies on a knowledge base constructed by automatically mining semantic propositions from Internet webpages using the TextRunner information extraction system.[1] The primary goal of this chapter is to assess whether these propositions can help identify likely joint implicit argument configurations. In the following section, I review work on joint inference within semantic role labeling. In Sections 5.3 and 5.4, I present the joint implicit argument model and its features. Evaluation results for this model are given in Section 5.5. The joint model contains many simplifying assumptions, which I address in Section 5.6. I conclude in Section 5.7.

## 5.2 Related work

### Joint models for SRL

A number of recent studies have shown that semantic arguments are not independent and that system performance can be improved by taking argument dependencies into account. Consider the following examples, which were discussed in Section 2.2.3:

(5.3) [*Temporal* The day] that [$arg_0$ the ogre] [*Predicate* cooked] [$arg_1$ the children] is still remembered.

(5.4) [$arg_1$ The meal] that [$arg_0$ the ogre] [*Predicate* cooked] [*Beneficiary* the children] is still remembered.

These examples (due to Toutanova et al. (2008)) demonstrate the importance of inter-argument dependencies. The fact that the sentential subject is headed by *meal* in 5.4 instead of *day* causes a dramatic change in the interpretation of the constituent following the predicate. Toutanova et al. first generated an $n$-best list of argument labels for a predicate

[1] http://www.cs.washington.edu/research/textrunner/reverbdemo.html

instance. They then re-ranked this list using joint features that describe multiple arguments simultaneously. For example, one of the features captures the argument label sequence as follows for Examples 5.3 and 5.4, respectively:

(5.5) [voice:passive, lemma:cook, Temporal, $arg_0$, Predicate, $arg_1$]
(5.6) [voice:passive, lemma:cook, $arg_1$, $arg_0$, Predicate, Beneficiary]

The label sequences in 5.5 and 5.6 help rule out globally invalid configurations such as the following:

(5.7) [voice:passive, lemma:cook, $arg_1$, $arg_0$, Predicate, $arg_0$]

The label sequence in 5.7 violates a commonly used constraint that allows a single constituent to be given each argument label.

The unique label constraint just mentioned is also important to the work of Punyakanok et al. (2008), who formulate a variety of constraints on argument labels. Punyakanok et al. treat these constraints as binary variables within an integer linear program, which is optimized to produce the final labeling. Other constraints include the following (for a complete list, see p. 267 of the cited work):

- **Arguments cannot overlap the predicate.** This constraint was used by the nominal SRL model presented in Chapter 2, where candidate arguments were not allowed to overlap the predicate.[2]

- **Arguments cannot overlap each other.** This constraint was also used by the nominal SRL system, which applied post-processing heuristics to remove argument overlap.

Although the work of Punyakanok et al. focuses on SRL within single sentences, the key finding is pertinent to multi-sentence implicit argumentation: semantic arguments should not be predicted independently of each other.

[2]The exception to this constraint for the nominal SRL system of Chapter 2 is that incorporated arguments may overlap the predicate. See page 33 for details concerning incorporated arguments.

Following this line of work, Ritter et al. (2010) investigated joint selectional preferences. Traditionally, a selectional preference model provides the strength of association between a predicate-argument position and a specific textual expression. Returning to Examples 5.1 and 5.2, one sees that the selectional preference for *president* and *economy* in the $iarg_0$ position of *loss* should be high because each expression denotes an entity capable of losing something. The traditional selectional preference model was used in Chapter 4 as a source of information for identifying implicit arguments. Ritter et al. extended this single-argument model using a joint formulation of Latent Dirichlet Allocation (LDA) (Blei et al., 2003). In the generative version of joint LDA, text for the argument positions is generated from a common hidden variable. This approach reflects the intuition behind Examples 5.1 and 5.2 and would help identify *president* as the $iarg_0$. Training data for the model was drawn from a large corpus of two-argument tuples extracted by the TextRunner system, which I describe next.

### The TextRunner information extraction system

Both Ritter et al.'s model and the model described in this chapter rely heavily on information extracted by the TextRunner system (Banko et al., 2007). The TextRunner system extracts tuples from Internet webpages in an unsupervised fashion. One key difference between TextRunner and other information extraction systems is that TextRunner does not use a closed set of relations (compare to the work described by ACE (2008)). Instead, the relation set is left open, leading to the notion of Open Information Extraction (OIE). Although OIE often has lower precision than traditional information extraction, it is able to extract a wider variety of relations at precision levels that are often useful (Banko and Etzioni, 2008). Returning again to Examples 5.1 and 5.2, one can query TextRunner in the following way:

**TextRunner Query**

$arg_0$: ?

Predicate: lose

$arg_1$: election

In the TextRunner system, $arg_0$ typically indicates the *Agent* and $arg_1$ typically indicates the *Theme*. TextRunner provides many tuples in response to this query, two of which are shown below:

(5.8) Usually, [$arg_0$ the president's party] [*Predicate* loses] [$arg_1$ seats in the mid-term election]

(5.9) [$arg_0$ The president] [*Predicate* lost] [$arg_1$ the election].

The tuples present in these sentences (and many others) suggest that presidents are capable of losing elections. This was one possible inference for Examples 5.1 and 5.2. The other possible inference - that the economy might lose the election - is not supported as strongly by tuples returned for the TextRunner query. Given all of the returned tuples, only a single one involves *economy* in the $arg_0$ position:

(5.10) Any president will take credit for [$arg_0$ a good economy] or [*Predicate* lose] [$arg_1$ an election] over a bad one.

In 5.10, TextRunner has not analyzed the arguments correctly (*president* should be the $arg_0$, not *economy*). Later in this chapter, I show how evidence from the tuple lists can be aggregated such that correct analyses (5.8 and 5.9) are favored over incorrect analyses (5.10). Given the tuple-based preference for *president* in the $arg_0$ of *lose* where the $arg_1$ is *election*, the system would hopefully select $c_1$ (*The president*) as the $arg_0$ in Examples 5.1 and 5.2. The primary contribution of this chapter is an exploration of how such tuple-based preferences can be computed and applied to the task of implicit argument identification.[3]

## 5.3 Joint model formulation

To simplify the experimental setting, the model described in this section targets the specific situation where a predicate instance $p$ takes an implicit $iarg_0$ as well as an implicit $iarg_1$.

[3]Thanks to Robert Bart and Alan Ritter at the University of Washington for their assistance with the TextRunner system.

Whereas the model in the previous chapter classifies candidates for these positions independently, the model in this chapter classifies joint structures by evaluating the following binary prediction function:

$$P(+|\left\langle p, iarg_0, c_i, iarg_1, c_j \right\rangle) \tag{5.11}$$

Equation 5.11 gives the probability of the joint assignment of $c_i$ to $iarg_0$ and $c_j$ to $iarg_1$. Given a set of $n$ candidates $c_1, \ldots, c_n \in C$, the best labeling is found by considering all possible assignments of $c_i$ and $c_j$:

$$\underset{(c_i,c_j)\in C\text{x}C \text{ s.t. } i\neq j}{\arg\max} P(+|\left\langle p, iarg_0, c_i, iarg_1, c_j \right\rangle) \tag{5.12}$$

Consider modified versions of Examples 5.1 and 5.2:

(5.13) [$c_1$ The president] is currently struggling to manage [$c_2$ the country's economy].
(5.14) If he cannot get it under control before [$c_3$ the next election], a [$p$ loss] might result.

In this case, we are looking for the $iarg_0$ as well as the $iarg_1$ for the *loss* predicate. Three candidates $c_1$, $c_2$, and $c_3$ are marked. The joint model evaluates the following probabilities, taking the highest scoring to be the final assignment:

$$P(+|\left\langle \text{loss}, iarg_0, \text{president}, iarg_1, \text{economy} \right\rangle)$$
$$^*P(+|\left\langle \text{loss}, iarg_0, \text{president}, iarg_1, \text{election} \right\rangle)$$
$$P(+|\left\langle \text{loss}, iarg_0, \text{economy}, iarg_1, \text{president} \right\rangle)$$
$$P(+|\left\langle \text{loss}, iarg_0, \text{economy}, iarg_1, \text{election} \right\rangle)$$
$$P(+|\left\langle \text{loss}, iarg_0, \text{election}, iarg_1, \text{president} \right\rangle)$$
$$P(+|\left\langle \text{loss}, iarg_0, \text{election}, iarg_1, \text{economy} \right\rangle)$$

Intuitively, only the starred item should have a high probability. As described in the previous section, TextRunner might be capable of modeling such intuitions if the tuple data can be aggregated in the right way. In the following section, I describe such an aggregation method, which forms the basis for features used to estimate the above probabilities.

## 5.4 Joint model features based on TextRunner

The TextRunner system has been extracting massive amounts of knowledge in the form of tuples such as the following:

$$\langle \text{president}, \text{lose}, \text{election} \rangle$$

The database of tuples can be queried by supplying a value for one or more of the tuple arguments. For example, the following is a partial result list for the query $\langle \text{president}, \text{lose}, ? \rangle$:

$\langle$Kenyan president, lose, election$\rangle$
$\langle$party of president, lose seat in, election$\rangle$
$\langle$president, lose, ally$\rangle$
$\langle$President Bush, lose support for, mission$\rangle$

The final argument in each of these tuples provides a single answer to the question "What might a president lose?". In order to aggregate these answers, I first generalize each to its WordNet synset (the WordNet gloss for each synset is shown after the arrow):

$\langle$Kenyan president, lose, election$\rangle$ $\rightarrow$ vote to select the winner of a position
$\langle$party of president, lose seat in, election$\rangle$ $\rightarrow$ vote to select the winner of a position
$\langle$president, lose, ally$\rangle$ $\rightarrow$ a friendly nation
$\langle$President Bush, lose support for, mission$\rangle$ $\rightarrow$ an organization of missionaries

In cases where the answer argument is a phrase, the phrase's syntactic head is mapped to a WordNet synset. In cases where the answer argument can be mapped to multiple synsets (i.e., it has more than one sense), the argument is mapped to the most common sense as

listed in the WordNet database. The final mapping above shows the effect of sense ambiguity, where *mission* in the sense of *war* is mapped to *mission* in the sense of *religion*. This type of error is inevitable because the mapping process selects the most common sense instead of applying a more sophisticated sense disambiguation model; however, the negative effects of sense ambiguity are mitigated by the aggregation process described below.

Having mapped the answer argument of each tuple to its WordNet synset, each synset is ranked according to the number of answer arguments that it covers. For the query $\langle \text{president}, \text{lose}, ? \rangle$, this produces the following ranked list of WordNet synsets:

1. election (77)
2. war (51)
3. vote (39)
4. people (34)
5. support (26)

   ...

In the list above, I have provided a one-word paraphrase for each of the top five synsets. The number in parentheses indicates how many answer arguments are covered by the synset. These synsets indicate likely answers to the original question of "What might a president lose?".

In a similar manner, one can answer a question such as "What might lose an election?" using tuples extracted by TextRunner. The procedure described above produces the following ranked list of WordNet synsets to answer this question:

   ...

9. people (62)
10. Republican (51)
11. Republican party (51)
12. Hillary (50)
13. president (49)

    ...

In this case, the expected answer (*president*) ranks 13th in the list of answer synsets. It is important to note that lower ranked answers are not necessarily incorrect answers. It is a simple fact that a wide variety of entities can lose an election. Items 9-13 are all perfectly reasonable answers to the original question of what might lose an election. The features described later in this section will accommodate this observation.

The two symmetric questions defined and answered above are closely connected to the implicit argument situation discussed previously and reproduced below:

(5.15) [$c_1$ The president] is currently struggling to manage [$c_2$ the country's economy].
(5.16) If he cannot get it under control before [$c_3$ the next election], a [$p$ loss] might result.

In Example 5.16, one is searching for the implicit $iarg_0$ and $iarg_1$ to the *loss* predicate. Candidates $c_i$ and $c_j$ that truly fill these positions should be compatible with questions in the following forms:

Question: What did $c_i$ lose?
Answer: $c_j$

Question: What entity lost $c_j$?
Answer: $c_i$

If either of these question-answer pairs is not satisfied, then the joint assignment of $c_i$ to $iarg_0$ and $c_j$ to $iarg_1$ should be considered unlikely. Using the first question-answer pair above as an example, satisfaction is determined in the following way:

1. Resolve anaphoric expressions and normalize named entities in the sentences from which $c_i$ and $c_j$ originate.[4]

2. Query TextRunner for $\langle c_i, \text{lose}, ? \rangle$, retrieving the top $n$ tuples.

3. Map the final argument of each tuple to its WordNet synset and rank the synsets by frequency, producing the ranked list $A$ of answer synsets.

[4]I used gold-standard anaphora annotations from Weischedel and Brunstein (2005) and the automatic named entity extractor created by Bikel et al. (1999) for this purpose.

4. Map $c_j$ to its WordNet synset $synset_{c_j}$ and determine whether $synset_{c_j}$ exists in $A$. If it does, the question-answer pair is satisfied.

Some additional processing is required to determine whether $synset_{c_j}$ exists in $A$. This is due to the hierarchical organization of WordNet. For example, suppose that $synset_{c_j}$ is the synset containing "primary election" and $A$ contains synsets paraphrased as follows:

1. election
2. war
3. vote
4. people
5. support

   ...

$synset_{c_j}$ does not appear directly in this list; however, its existence in the list is implied by the following hypernymy path within WordNet:

$$\text{primary election} \xrightarrow{\text{is-a}} \text{election}$$

Intuitively, if $synset_{c_j}$ is connected to a highly ranked synset in $A$ by a short path, then one has evidence that $synset_{c_j}$ answers the original question. The evidence is weaker if the path is long, as in the following example:

$$\text{open primary} \xrightarrow{\text{is-a}} \text{direct primary} \xrightarrow{\text{is-a}} \text{primary election} \xrightarrow{\text{is-a}} \text{election}$$

Additionally, a path between more specific synsets (i.e., those lower in the hierarchy) indicates a stronger relationship than a path between more general synsets (i.e., those higher in the hierarchy). These two situations are depicted in Figure 5.1. The synset similarity metric defined by Wu and Palmer (1994) combines the path length and synset depth intuitions into

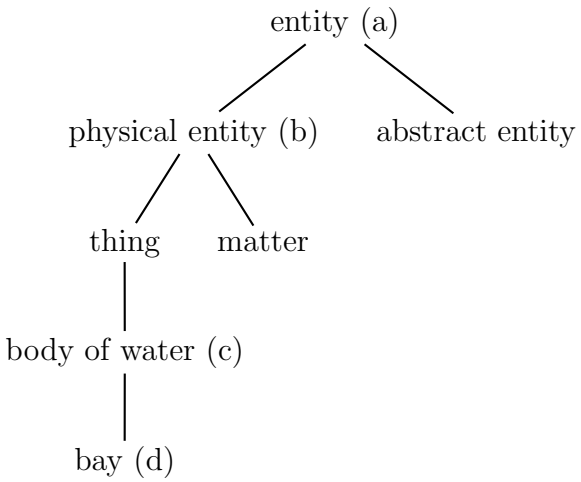



Figure 5.1: Effect of depth on WordNet synset similarity. All links indicate *is-a* relationships. Although the link distance from (a) to (b) equals the distance from (c) to (d), the latter are more similar due to their lower depth within the WordNet hierarchy.

a single numeric score that is defined as follows:

$$sim(synset_1, synset_2) = \frac{2 * depth(lca(synset_1, synset_2))}{depth(synset_1) + depth(synset_2)} \tag{5.17}$$

In Equation 5.17, *lca* returns the lowest common ancestor of the two synsets within the WordNet hierarchy.

To summarize, Equation 5.17 indicates the strength of association between $synset_{c_j}$ (e.g., primary election) and a ranked synset $synset_a$ from $A$ that answers a question such as "What might a president lose?". If the association between $synset_{c_j}$ and $synset_a$ is weak, then the assignment of $c_j$ to $iarg_1$ is unlikely. The process works similarly for assessing $c_i$ as the filler of $iarg_0$. In what follows, I quantify this intuition for use in estimating the joint probability defined in Equation 5.11, which is reproduced below:

$$P(+|\left\langle p, iarg_0, c_i, iarg_1, c_j \right\rangle) \tag{5.18}$$

In order to estimate Equation 5.18 using LibLinear, one must extract numeric features from the conditioning information. I describe these features below.

**Feature 1: Maximum TextRunner association strength.** Given the conditioning variables in Equation 5.18, there are two questions that can be asked:

Question: What did $c_i$ $p$?
Answer: $c_j$

Question: What entity $p$ $c_j$?
Answer: $c_i$

Each of these questions produces a ranked list of answer synsets using the approach described previously. The synset for each answer string will match zero or more of the answer synsets, and each of these matches will be associated with a similarity score as defined in Equation 5.17. Feature 1 considers all such similarity scores and selects the maximum. A high value for this feature indicates that one (or both) of the candidates ($c_i$ or $c_j$) is likely to be an answer to its associated question and is likely to fill its associated implicit argument position.

**Feature 2: Maximum TextRunner reciprocal rank.** Of all the answer matches described for Feature 1, Feature 2 selects the highest ranking and forms the reciprocal rank. Thus, values for Feature 2 are in [0,1] with larger values indicating matches with higher ranked answer synsets.

**Feature 3: Number of TextRunner matches.** This feature records the number of matches from either of the questions described for Feature 1.

**Feature 4: Summed TextRunner reciprocal rank.** Feature 2 considers answer synset matches from either of the posed questions; ideally, each question-answer pair should have some influence on the probability estimate in Equation 5.18. Feature 4 looks at the answer

synset matches from each question individually. The match with highest rank for each question is selected, and the reciprocal rank $\frac{2}{r_1 + r_2}$ is computed. The value of this feature is zero if either of the questions fails to produce a matching answer synset.

**Features 5 and 6: Local classification scores.** The joint model described in this chapter does not replace the local prediction model presented in the previous chapter. The latter uses a wide variety of important features that cannot be ignored. Like previous joint models (e.g., the one described by Toutanova et al. (2008)), the joint model works on top of the local prediction model, whose scores are incorporated into the joint model as feature-value pairs. Given the local prediction scores for the $iarg_0$ and $iarg_1$ positions in Equation 5.18, the joint model forms two features: (1) the sum of the scores for $c_i$ filling $iarg_0$ and $c_j$ filling $iarg_1$, and (2) the product of these two scores.

## 5.5 Evaluation

I evaluated the model described in this chapter over the manually annotated implicit argument data used elsewhere in this dissertation. As mentioned in Section 5.3, all joint model experiments were conducted using predicate instances that take an $iarg_0$ and $iarg_1$ in the ground-truth annotations. I reused the ten-fold cross-validation setup from the previous chapter as well as the evaluation metrics (see Section 4.5, p. 96 for more details). For each evaluation fold, features were selected using only the corresponding training data. I used the forward feature subset selection algorithm from Section A.8 (p. 144) for this purpose.

For comparison with the model from the previous chapter, I also evaluated the local prediction model on the evaluation data. Because this model predicted implicit arguments independently, it continued to use the conflict resolution heuristics described on page 94. However, the prediction threshold $t$ was eliminated because the system could safely assume that a true filler for each of the $iarg_0$ and $iarg_1$ positions existed.

Table 5.1 presents the evaluation results. The first thing to note is that these results are

| | | Local model | | | Joint model | | |
|---|---|---|---|---|---|---|---|
| | # Imp. args. | $P$ | $R$ | $F_1$ | $P$ | $R$ | $F_1$ |
| price | 40 | 65.0 | 65.0 | 65.0 | 67.5 | 67.5 | 67.5 |
| sale | 34 | 86.5 | 86.5 | 86.5 | 84.3 | 84.3 | 84.3 |
| plan | 30 | 60.0 | 60.0 | 60.0 | 56.7 | 56.7 | 56.7 |
| bid | 26 | 66.7 | 66.7 | 66.7 | 78.2 | 78.2 | 78.2 |
| fund | 18 | 83.3 | 83.3 | 83.3 | 83.3 | 83.3 | 83.3 |
| loss | 14 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| loan | 12 | 63.6 | 58.3 | 60.9 | 50.0 | 50.0 | 50.0 |
| investment | 8 | 57.1 | 50.0 | 53.3 | 62.5 | 62.5 | 62.5 |
| Overall | 182 | 72.6 | 71.8 | 72.2 | 73.1 | 73.1 | 73.1 |

Table 5.1: Joint implicit argument identification evaluation results. The second column indicates the number of filled implicit argument positions for the corresponding predicate(s). For comparison, the full implicit argument annotation data contain approximately 1000 filled implicit argument positions (see Table 4.1 on page 77).

not comparable with the results of the previous chapter. In general, performance is much higher because predicate instances reliably took implicit arguments in the $iarg_0$ and $iarg_1$ positions. The overall performance increase was relatively small (approximately 1 percentage point). The *bid* and *investment* predicates showed larger gains; however, due to the small size of the test collection, the differences in $F_1$ between the local and joint models were not significant at $p = 0.10$ when using the bootstrap resampling procedure described by Efron and Tibshirani (1993).

## 5.6 Discussion

### 5.6.1 Example improvement versus local model

The *bid* and *investment* predicates show the largest increase for the joint model versus the local model. Below, I give an example of the *investment* predicate for which the joint model correctly identified the $iarg_0$ and the local model did not.

(5.19) [Big investors] can decide to ride out market storms without jettisoning stock.

(5.20) Most often, [*c* they] do just that, because stocks have proved to be the best-performing long-term [*Predicate* investment], attracting about $1 trillion from pension funds alone.

Both models identified the $iarg_1$ as *money* from a prior sentence (not shown). The local model incorrectly predicted *$1 trillion* in Example 5.20 as the $iarg_0$ for the *investment* event. This mistake demonstrates a fundamental limitation of the local model: it cannot detect simple incompatibilities in the predicted argument structure. It does not know that "money investing money" is a rare or impossible event in the real world.

For the joint model's prediction, consider the constituent marked with *c* in Example 5.20. This constituent is resolved to *Big investors* in the preceding sentence. Thus, the two relevant questions are as follows:

Question: What did big investors invest?
Answer: money

Question: What entity invested money?
Answer: big investors

The first question produces the following ranked list of answer synsets (the number in parentheses indicates the number of answer arguments that mapped to the synset):

money (71)
amount (38)
million (38)
billion (22)
capital (21)

As shown, the answer string of *money* matches the top-ranked answer synset. The second question produces the following ranked list of answer synsets:

company (642)
people (460)
government (275)

business (75)
investor (70)

In this case, the answer string *Big investors* matches the fifth answer synset. The combined evidence of these two question-answer pairs allows the joint system to successfully identify *Big investors* as the $iarg_0$ of the *investment* predicate in Example 5.20.

### 5.6.2 Test collection size

The performance improvements for the joint model versus the local model were not found to be statistically significant at $p = 0.10$. Other studies of joint models for SRL (e.g., the one by Toutanova et al. for verbal SRL (2008)) have shown slightly larger gains (2.8 $F_1$ points). These gains, although modest, were statistically significant because of the larger test sample size. There are at least two ways in which the evaluation test sample in this chapter can be expanded. First, one could annotate additional implicit argument data. This is technically straightforward; however, implicit argument annotation is an expensive process. Alternatively, one could add to the test sample predicate instances that are not constrained to take both an $iarg_0$ and an $iarg_1$. This approach makes the modeling task more difficult, but the difficulty is one that needs to be addressed in order for the system to be practically applicable. Below, I discuss other issues surrounding the joint model and its wider application.

### 5.6.3 Toward a generally applicable joint model

The joint model presented in this chapter assumes that all predicate instances take an $iarg_0$ and an $iarg_1$. This assumption clearly does not hold for real data (these positions are often not expressed in the text), but relaxing it will require investigation of the following issues.

1. **Explicit arguments** should also be considered when determining whether a candidate $c$ fills an implicit argument position $iarg_n$. The motivation here is similar to

that given elsewhere in the current chapter: arguments (whether implicit or explicit) are not independent. This is demonstrated by the example in the beginning of this chapter (p. 110), where *election* is an explicit argument to the predicate and affects the implicit argument inference. The model developed in this chapter only considers jointly occurring implicit arguments.

2. **Other implicit argument positions (e.g., $iarg_2$, $iarg_3$, etc.)** need to be accounted for as well. This will present a challenge when it comes to extracting the necessary propositions from TextRunner. Currently, TextRunner only handles tuples of the form $\langle arg_0, p, arg_1 \rangle$. Other argument positions are not directly analyzed by the system; however, because TextRunner also returns the sentence from which a tuple is extracted, these additional argument positions could be identified in the following way:

   (a) For an instance of the *sale* predicate with an $arg_0$ of *company*, to find likely $arg_2$ fillers (the entity purchasing the item), query TextRunner with $\langle \text{company}, sell, ? \rangle$.

   (b) Perform standard verbal SRL on the sentences for the resulting tuples, identifying any $arg_2$ occurrences.

   (c) Cluster and rank the $arg_2$ fillers according to the method described in this chapter.

   This approach combines Open Information Extraction with traditional information extraction (i.e., verbal SRL).

3. **Computational complexity and probability estimation** is a problem for many joint models. The model presented in this chapter quickly becomes computationally intractable when the number of candidates and implicit argument positions becomes moderately large. This is because Equation 5.12 (p. 115) considers all possible assignments of candidates to implicit argument positions. With as few as thirty candidates and five argument positions (not uncommon), one must evaluate $30!/25! = 17,100,720$ possible assignments. Although this particular formulation is not tractable, one based on dynamic programming or heuristic search might give reasonable results.

## 5.7 Conclusions

Previous chapters of this dissertation have investigated the nature and recovery of semantic arguments for nominal predicates. Throughout these chapters, the models have assumed that the arguments are independent of each other. This assumption simplifies the computational modeling of semantic arguments, but it ignores the joint nature of natural language. In order to take advantage of the information provided by jointly occurring arguments, the local prediction models must be enhanced.

The current chapter has presented a preliminary investigation into the joint modeling of implicit arguments for nominal predicates. The model relies heavily on information extracted by the TextRunner extraction system, which pulls propositional tuples from millions of Internet webpages. These tuples encode world knowledge that is necessary for resolving semantic arguments in general and implicit arguments in particular. This chapter has proposed methods of aggregating tuple knowledge to guide implicit argument resolution. The aggregated knowledge is applied via a re-ranking model that operates on top of the local prediction model described in the previous chapter.

In general, the model and results presented in this chapter are exploratory. The performance gains of the joint model versus the local model were not found to be statistically significant, with a large factor being the small size of the testing corpus. It is possible to identify cases of improvement; however, a significant amount of future work will be required to make the model more effective and applicable in a wider usage context. Additional predicates and argument positions need to be accounted for, in turn requiring more efficient computational approaches. This effort will hopefully lead to better performance using an approach that more accurately reflects the joint properties of natural language.

# CHAPTER 6

# Summary of contributions and future work

## 6.1 Summary of contributions

### 6.1.1 A nominal SRL system for real-world use

This dissertation has addressed a number of preexisting issues surrounding nominal semantic role labeling. Most basic among these was the need for a nominal SRL system that is capable of handling unstructured textual input. Original work in nominal SRL assumed the existence of certain pieces of the SRL structure (i.e., nominal predicates). It has always been clear that this assumption does not hold when working on raw text; however, it was unclear whether, and to what extent, removal of this assumption would affect nominal SRL performance. This dissertation has confirmed that predicate identification is a crucial part of nominal SRL for many frequent predicates. Without a predicate identification model, the nominal SRL system presented in Chapter 2 suffers an argument $F_1$ loss of approximately 8%.

The predicate identification model allows the nominal SRL system to move out of its simplified experimental environment and into real-world settings such as the processing of raw newswire text, intelligence reports, and other sources of important information. Genre changes will undoubtedly have a negative impact on performance, but a tremendous amount of text is created in a form very similar to the training genre of Wall Street Journal newswire. Given the frequency and semantic importance of nominal predicates within this genre, it is clear that the nominal SRL system has the potential to enhance automatic understanding of important textual resources. The nominal SRL system is freely available for non-commercial

use.[1]

### 6.1.2 A focused, data-driven analysis of implicit arguments

Traditional SRL approaches such as the one just mentioned limit the search for arguments to the sentence containing the predicate of interest. Many systems take this assumption a step further and restrict the search to the predicate's local syntactic environment; however, predicates and the sentences that contain them rarely exist in isolation. As shown throughout this dissertation, they are usually embedded in a semantically rich discourse that contains complex phenomena such as coreference, coherence, and rhetorical structure. This dissertation has endeavored to make implicit arguments part of the discourse landscape. As a first step, this dissertation presents a manually annotated corpus of implicit arguments that complements the only other currently available corpus (Ruppenhofer et al., 2010). Analyses of this data reveal a number of insights.

First, one finds a correspondence between a verb and its nominal form in terms of implicit arguments. For the predicates considered in this dissertation, if an argument is required by the verb form and missing from the nominal form, then it is likely that the argument is present in the discourse that surrounds the nominal form. Second, the data show that implicit arguments contribute a substantial amount of novel information to the text. This information is not provided by arguments to verbs, which are another primary source of semantic information. Third, implicit arguments tend to be located in or nearby the sentence containing the predicate of interest. This is an important property when implicit argument identification is considered; without it, the search space could easily become unmanageable.

### 6.1.3 A novel model for implicit argument identification

Researchers formulated the task of implicit argument identification more than two decades ago; however, the task has received relatively little attention since that time. This disser-

[1]Please contact the author at gerber.matthew@gmail.com for more information.

tation presents a novel model for the implicit argument identification task and evaluates the model using the manually constructed corpus described above. The model draws evidence from a variety of sources, many of them created specifically for the task. In general, the most informative features are derived from semantic sources instead of the syntactic sources commonly used by standard SRL systems. Using this information, the system is able to recover implicit arguments with an overall $F_1$ score of 50%. This result represents the state-of-the-art, as there are no other results to compare it to.

Lastly, this dissertation contributes a preliminary exploration of joint modeling for implicit arguments. The other models in this dissertation assume that arguments are independent, regardless of whether they are implicit or not. The joint implicit argument model re-ranks the output of the independent model using knowledge extracted from millions of Internet webpages. This knowledge helps to identify likely joint occurrences of implicit arguments. Overall gains from this approach are small; however, the experiments and discussion constitute a starting point for future work in this direction.

## 6.2 Summary of future work

The models in this dissertation (with the exception of the model described in Chapter 5) apply an argument independence assumption. Under this assumption, each argument can be identified independently of each other argument. A wide range of psycholinguistic evidence suggests that this assumption does not reflect the true nature of human sentence comprehension, which builds up joint semantic structures at the sentence and discourse level (see Section 4.2 for details). Experimentally, researchers have found that joint models of semantic arguments can improve automatic identification for verbal predicates. Based on these findings, it seems natural to formulate a joint model for nominal SRL (the standard, non-implicit task described in Chapters 2 and 3).

The implicit argument model described in Chapter 4 can be improved in a variety of ways that are not directly related to joint modeling. As shown in Section 4.6, many implicit

argument identification errors were caused by the absence of true implicit arguments within the set of candidate constituents. More intelligent windowing strategies in addition to alternate candidate sources might offer some improvement. Although I consistently observed development gains from using automatic coreference resolution, this process creates errors that need to be studied more closely. It will also be important to study implicit argument patterns of non-verbal predicates such as the partitive *percent*. These predicates are among the most frequent in the TreeBank and are likely to require approaches that differ from the ones we pursued.

The implicit argument model developed in Chapter 4 is not generally applicable. It is limited to the ten predicates for which there exist manually annotated training data. Additional data will be required in order to extract implicit arguments for all predicates. An entirely manual annotation project is feasible; however, it will be complicated by the fact that implicit argument annotation is labor intensive and potentially error prone. This is because the annotation process requires both argument and coreference identification, each of which is difficult by itself. Thus, it might be productive to combine additional manual implicit argument annotation with semi-supervised learning from labeled and unlabeled data. Similar approaches have been applied to the standard verbal SRL task (for recent examples, see Lang and Lapata (2010), Fürstenau and Lapata (2009), Deschacht and Moens (2009), Abend et al. (2009), and Swier and Stevenson (2004)).

Regardless of the training corpus size, the implicit argument model will inevitably encounter previously unseen predicates. The negative effects of these predicates can be mitigated by designing features that transfer well. For example, recall Feature 11 (p. 138). This feature has a value of *true* when the candidate implicit argument is an argument to a predicate that is in the same VerbNet class as the predicate being filled. This particular feature does not depend on the exact predicates under consideration; it only checks whether they are in the same class. As such, this feature would transfer well to predicates that were not observed in the training data. VerbNet, with its verb classes and network structure, is

likely to be a good source of information.

The joint implicit argument model developed in Chapter 5 produced reasonable gains for two of the predicates, but overall results showed smaller gains. Additional work will be required in order to fully understand the potential of joint modeling for implicit arguments. The model will need to be extended to argument positions other than $iarg_0$ and $iarg_1$. This, in turn, will require a method of evaluating the possible assignments of candidates to more than two argument positions - a potentially intractable problem if done by brute force. Dynamic programming and heuristic search are possible answers to this problem.

As mentioned in the preceding section, it is possible to evaluate the standard nominal SRL model (no implicit arguments) on textual resources other than the Wall Street Journal, upon which this dissertation is based. The implicit argument model can be evaluated similarly, as long as the test cases involve only the ten predicates for which the model is designed. Such experiments are important because they test the ability of the model to generalize across domains. Domains with specialized vocabularies - biomedicine, for example - will pose significant challenges. Many of the features used throughout this dissertation are lexical in nature; that is, they depend on the actual word content of a phrase. Specialized vocabularies will not be accounted for in the training data. A genre such as standard news reporting will pose less of a problem, but one should expect performance to drop nonetheless. Domain adaptation techniques such as those presented by Daumé et al. (2010) should have something to offer in this respect.

# APPENDIX

## A.1 Support verb identification

Support verbs link long-distance arguments to nominal predicates. For example:

(A.1) [$Arg_0$ John] [*Support* took] a [*Predicate* walk].

In Example A.1, *took* does not have the usual meaning of forcibly changing possession; rather, this verb's purpose is to bring in *John* as the $Arg_0$ (walker) of *walk*. I created a binary logistic regression model to automatically identify support verb tokens. The model uses the features shown in the table at the end of this page. I set the LibLinear model parameters as follows: bias=1, c=4, $w+ = 1$. A prediction threshold of $t = 0.294$ was used at testing time. Overall support verb $F_1$ for this model was 53.36%.

| # | Feature value description |
|---|---|
| 1 | First word subsumed by $n$. |
| 2 | Semantic head of $n$'s right sibling. |
| 3 | Context-free grammar rule that expands $n$'s right sibling. |
| 4 | Syntactic head of $n$'s left sibling. |
| 5 | Context-free grammar rule that expands $n$'s grandparent. |
| 6 | Context-free grammar rule that expands $n$'s parent. |
| 7 | Last word of $n$'s right sibling. |
| 8 | Head word of $n$'s right sibling. |
| 9 | Context-free grammar rule that expands $n$'s left sibling. |
| 10 | The object head of the prepositional phrase that follows $n$. |
| 11 | Parse tree path to nearest passive verb. |
| 12 | Part of speech (POS) of the head word of $n$'s right sibling. |
| 13 | The POS of $n$'s parent's head word. |
| 14 | $n$'s parent's head word. |
| 15 | The syntactic category of $n$'s right sibling. |
| 16 | $n$'s syntactic category. |
| 17 | The POS of the syntactic head word of $n$'s left sibling. |
| 18 | Context-free grammar rule that expands $n$'s great-grandparent. |

Table A.1: Features used for support verb identification, sorted by feature selection rank. All features were based on automatically identified syntactic parse trees.

## A.2 Nominal predicate features

| # | Feature value description |
|---|---|
| 1* | $n$'s ancestor grammar rules. |
| 2 | $n$'s stemmed text. |
| 3 | Syntactic category of $n$'s right sibling. |
| 4 | First word of $n$'s left sibling. |
| 5* | Parse tree path from $n$ to previous nominal, with lexicalized source. |
| 6 | The stemmed content words in a one-word window around $n$. |
| 7 | $n$'s morphological suffix. |
| 8 | Parse tree path from $n$ to closest support verb, with lexicalized destination. |
| 9 | Parse tree path to nearest passive verb. |
| 10 | Number of left siblings of $n$. |
| 11 | Parse tree path to previous predicate node, with lexicalized source. |
| 12 | Semantic head word of $n$'s right sibling. |
| 13 | Parse tree path from $n$ to previous nominal with lexicalized source and destination. |
| 14 | Syntactic head word of $n$'s parent. |
| 15 | Head word of $n$'s left sibling. |
| 16* | PropBank markability score. |
| 17 | Signed token distance between $n$ and nearest support verb. |
| 18 | Parse tree path from $n$ to previous nominal. |
| 19 | The object head of the prepositional phrase that follows $n$. |
| 20 | Whether or not $n$ is followed by a prepositional phrase. |
| 21 | Semantic head word of $n$'s left sibling. |
| 22 | Whether or not $n$ surfaces before a passive verb. |
| 23 | Parse tree paths from $n$ to each support verb, including the support verb. |
| 24 | Syntactic category of $n$'s left sibling. |
| 25 | Context-free grammar rule of $n$'s left sibling. |
| 26 | Whether or not $n$ is the head of it's parent node. |
| 27 | Whether or not the previous term in the lexicon is the previous predicate. |
| 28 | Context-free grammar rule of $n$'s grandparent. |
| 29 | Parse tree path from $n$ to previous nominal, with lexicalized destination. |
| 30 | Number of right siblings of $n$. |
| 31 | First word of $n$'s right sibling. |
| 32 | Last word of $n$'s right sibling. |
| 33 | Signed token distance between $n$ and the previous predicate. |
| 34 | Part of speech of the syntactic head of $n$'s right sibling. |

Table A.2: Nominal predicate features, sorted by gain in selection algorithm. & denotes feature concatenation. Features marked with an asterisk are explained on page 45. Johansson and Nugues (2008) used features similar to 2 and 26.

## A.3 Nominal argument features

| # | Feature value description | New |
|---|---|---|
| 1 | 12 & 26. | |
| 2 | Position of $n$ relative to $p$ (beingfore/after) & 26. | * |
| 3 | First word subsumed by $n$. | |
| 4* | 12 & Position of $n$ relative to $p$ (before/after). | |
| 5 | 12 & 14. | |
| 6 | Head word of $n$'s parent. | |
| 7 | Last word subsumed $n$. | |
| 8 | $n$'s syntactic category & length of 26. | |
| 9 | First word of $n$'s right sibling. | * |
| 10* | Context-free grammar rule that expands the parent of $p$. | |
| 11 | Head word of the right-most NP in $n$ if $n$ is a PP. | |
| 12 | Stem of $p$ according to a Porter stemmer. | |
| 13 | Parse tree path from $n$ to the lowest common ancestor (LCA) of $n$ and $p$. | |
| 14 | Head word of $n$. | |
| 15 | 12 & $n$'s syntactic category. | |
| 16 | Context-free grammar rule that expands $n$'s parent. | * |
| 17* | Parse tree path from $n$ to the nearest support verb. | * |
| 18 | Last part of speech (POS) subsumed by $n$. | |
| 19 | Context-free grammar rule that expands $n$'s left sibling. | * |
| 20 | Head word of $n$, if the parent of $n$ is a PP. | |
| 21 | The POS of the head word of the right-most NP under $n$ if $n$ is a PP. | |
| 22 | Last word of $n$'s left sibling. | |
| 23 | Syntactic category of $n$. | |
| 24 | Whether or not $n$ comes before a passive verb. | * |
| 25 | Context-free grammar rule that expands $n$'s right sibling. | * |
| 26* | Parse tree path from $n$ to $p$. | |
| 27 | Whether or not $n$ is under an NP headed by $p$. | |
| 28 | First POS subsumed by $n$. | |
| 29 | Whether or not $n$ is an NP headed by $p$ and is also adjacent to a VP. | |
| 30 | Signed token distance from $n$ to $p$. | * |
| 31 | Tree depth of the LCA of $n$ and $p$. | * |
| 32 | Syntactic category of the LCA of $n$ and $p$. | * |

Table A.3: Nominal argument features, sorted by gain in selection algorithm. $n$ indicates the candidate argument node being classified. $p$ indicates the predicate under consideration. & denotes feature concatenation. Features marked with an asterisk are explained on page 29. Asterisks in the last column indicate features that were not used by Jiang and Zhai (2006) or Liu and Ng (2007).

## A.4 Role sets for the annotated predicates

Listed below are the role sets for the ten predicates used in Chapters 4 and 5.

Role set for *bid*:
$Arg_0$: bidder
$Arg_1$: thing being bid for
$Arg_2$: amount of the bid

Role set for *sale*:
$Arg_0$: seller
$Arg_1$: thing sold
$Arg_2$: buyer
$Arg_3$: price paid
$Arg_4$: beneficiary of sale

Role set for *loan*:
$Arg_0$: giver
$Arg_1$: thing given
$Arg_2$: entity given to
$Arg_3$: loan against (collateral)
$Arg_4$: interest rate

Role set for *cost*:
$Arg_1$: commodity
$Arg_2$: price
$Arg_3$: buyer
$Arg_4$: secondary commodity

Role set for *plan*:
$Arg_0$: planner
$Arg_1$: thing planned
$Arg_2$: beneficiary of plan
$Arg_3$: secondary plan

Role set for *investor*:
$Arg_0$: investor
$Arg_1$: thing invested
$Arg_2$: thing invested in

Role set for *price*:
$Arg_0$: seller
$Arg_1$: commodity
$Arg_2$: price
$Arg_3$: secondary commodity

Role set for *loss*:
$Arg_0$: entity losing something
$Arg_1$: thing lost
$Arg_2$: entity gaining thing lost
$Arg_3$: source of loss

Role set for *investment*:
$Arg_0$: investor
$Arg_1$: thing invested
$Arg_2$: thing invested in

Role set for *fund*:
$Arg_0$: funder
$Arg_1$: thing funded
$Arg_2$: amount of funding
$Arg_3$: beneficiary

## A.5 Implicit argument features

Table A.4

| # | Feature value description | Score |
|---|---|---|
| 1 | For every $f$, $p_f$ & $arg_f$ & $p$ & $iarg_n$. | 8.2 |
| 2 | Sentence distance from $c$ to $p$. | 4.0 |
| 3 | For every $f$, the head word of $f$ & the verbal form of $p$ & $iarg_n$. | 3.6 |
| 4 | Same as 1 except generalizing $p_f$ and $p$ to their WordNet synsets. | 3.3 |
| 5 | For every $f$, the WordNet synset for the head of $f$ & the verbal form of $p$ & $iarg_n$. | 1.0 |
| 6 | Whether or not $c$ and $p$ are themselves arguments to the same predicate. | 1.0 |
| 7 | $p$ & the semantic head word of $p$'s right sibling. | 0.7 |
| 8 | Whether or not any $arg_f$ and $iarg_n$ have the same integer argument position. | 0.7 |
| 9* | Frame element path between $arg_f$ of $p_f$ and $iarg_n$ of $p$ in FrameNet (Baker et al., 1998). | 0.6 |
| 10 | Percentage of elements in $c'$ that are subjects of a copular for which $p$ is the object. | 0.6 |
| 11* | Whether or not the verb forms of $p_f$ and $p$ are in the same VerbNet class and $arg_f$ and $iarg_n$ have the same thematic role. | 0.6 |
| 12 | $p$ & the last word of $p$'s right sibling. | 0.6 |
| 13* | Maximum targeted PMI between $arg_f$ of $p_f$ and $iarg_n$ of $p$. | 0.6 |
| 14 | $p$ & the number of $p$'s right siblings. | 0.5 |
| 15 | Percentage of elements in $c'$ that are objects of a copular for which $p$ is the subject. | 0.5 |
| 16 | Frequency of the verbal form of $p$ within the document. | 0.5 |
| 17 | $p$ & the stemmed content words in a one-word window around $p$. | 0.5 |
| 18 | Whether or not $p$'s left sibling is a quantifier (e.g., many, most, all, etc.). Quantified predicates tend not to take implicit arguments. | 0.4 |
| 19 | Percentage of elements in $c'$ that are copular objects. | 0.4 |
| 20 | TF cosine similarity between words from arguments of all $p_f$ and words from arguments of $p$. | 0.4 |
| 21 | Whether the path defined in 9 exists. | 0.4 |
| 22 | Percentage of elements in $c'$ that are copular subjects. | 0.4 |
| 23* | For every $f$, the VerbNet class/role of $p_f/arg_f$ & the class/role of $p/iarg_n$. | 0.4 |
| 24 | Percentage of elements in $c'$ that are indefinite noun phrases. | 0.4 |
| 25* | $p$ & the syntactic head word of $p$'s right sibling. | 0.3 |
| 26 | $p$ & the stemmed content words in a two-word window around $p$. | 0.3 |
| 27* | Minimum selectional preference between any $f$ and $iarg_n$ of $p$. Uses the method described by Resnik (1996) computed over an SRL-parsed version of the Penn TreeBank and Gigaword (Graff, 2003) corpora. | 0.3 |

Continued on next page...

Table A.4 (cont'd)

| # | Feature value description | Score |
|---|---|---|
| 28 | $p$ & $p$'s synset in WordNet. | 0.3 |
| 29 | Same as 27 except using the maximum. | 0.3 |
| 30 | Average per-sentence frequency of the verbal form of $p$ within the document. | 0.3 |
| 31 | $p$ itself. | 0.3 |
| 32 | $p$ & whether $p$ is the head of its parent. | 0.3 |
| 33* | Minimum coreference probability between $arg_f$ of $p_f$ and $iarg_n$ of $p$. | 0.3 |
| 34 | $p$ & whether $p$ is before a passive verb. | 0.3 |
| 35 | Percentage of elements in $c'$ that are definite noun phrases. | 0.3 |
| 36 | Percentage of elements in $c'$ that are arguments to other predicates. | 0.3 |
| 37 | Maximum absolute sentence distance from any $f$ to $p$. | 0.3 |
| 38 | $p$ & $p$'s syntactic category. | 0.2 |
| 39 | TF cosine similarity between the role description of $iarg_n$ and the concatenated role descriptions of all $arg_f$. | 0.2 |
| 40 | Average TF cosine similarity between each $arg_n$ of each $p_f$ and the corresponding $arg_n$ of $p$, where $n$s are equal. | 0.2 |
| 41 | Same as 40 except using the maximum. | 0.2 |
| 42 | Same as 40 except using the minimum. | 0.2 |
| 43 | $p$ & the head of the following prepositional phrase's object. | 0.2 |
| 44 | Whether any $f$ is located between $p$ and any of the arguments annotated by NomBank for $p$. When *true*, this feature rules out false positives because it implies that the NomBank annotators considered and ignored $f$ as a local argument to $p$. | 0.2 |
| 45 | Number of elements in $c'$. | 0.2 |
| 46 | $p$ & the first word of $p$'s right sibling. | 0.2 |
| 47 | $p$ & the grammar rule that expands $p$'s parent. | 0.2 |
| 48 | Number of elements in $c'$ that are arguments to other predicates. | 0.2 |
| 49 | Nominal form of $p$ & $iarg_n$. | 0.2 |
| 50 | $p$ & the syntactic parse tree path from $p$ to the nearest passive verb. | 0.2 |
| 51 | Same as 37 except using the minimum. | 0.2 |
| 52 | Same as 33 except using the average. | 0.2 |
| 53 | Verbal form of $p$ & $iarg_n$. | 0.2 |
| 54 | $p$ & the first word of $p$'s left sibling. | 0.2 |
| 55 | Average per-sentence frequency of the nominal form of $p$ within the document. | 0.2 |
| 56 | $p$ & the part of speech of $p$'s parent's head word. | 0.2 |
| 57 | Same as 33 except using the maximum. | 0.2 |
| 58 | Same as 37 except using the average. | 0.1 |

Continued on next page...

Table A.4 (cont'd)

| # | Feature value description | Score |
|---|---|---|
| 59* | Minimum path length between $arg_f$ of $p_f$ and $iarg_n$ of $p$ within VerbNet (Kipper, 2005). | 0.1 |
| 60 | Frequency of the nominal form of $p$ within the document. | 0.1 |
| 61 | $p$ & the number of $p$'s left siblings. | 0.1 |
| 62 | $p$ & $p$'s parent's head word. | 0.1 |
| 63 | $p$ & the syntactic category of $p$'s right sibling. | 0.1 |
| 64 | $p$ & $p$'s morphological suffix. | 0.1 |
| 65 | TF cosine similarity between words from all $f$ and words from the role description of $iarg_n$. | 0.1 |
| 66 | Percentage of elements in $c'$ that are quantified noun phrases. | 0.1 |
| 67* | Discourse relation whose two discourse units cover $c$ (the primary filler) and $p$. | 0.1 |
| 68 | For any $f$, the minimum semantic similarity between $p_f$ and $p$ using the method described by Wu and Palmer (1994) over WordNet (Fellbaum, 1998). | 0.1 |
| 69 | $p$ & whether or not $p$ is followed by a prepositional phrase. | 0.1 |
| 70 | $p$ & the syntactic head word of $p$'s left sibling. | 0.1 |
| 71 | $p$ & the stemmed content words in a three-word window around $p$. | 0.1 |
| 72 | Syntactic category of $c$ & $iarg_n$ & the verbal form of $p$. | 0.1 |
| 73 | Nominal form of $p$ & the sorted integer argument indexes (the $n$s) from all $arg_n$ of $p$. | 0.1 |
| 74 | Percentage of elements in $c'$ that are sentential subjects. | 0.1 |
| 75 | Whether or not the integer position of any $arg_f$ equals that of $iarg_n$. | 0.1 |
| 76 | Same as 13 except using the average. | 0.1 |
| 77 | Same as 27 except using the average. | 0.1 |
| 78 | $p$ & $p$'s parent's syntactic category. | 0.1 |
| 79 | $p$ & the part of speech of the head word of $p$'s right sibling. | 0.1 |
| 80 | $p$ & the semantic head word of $p$'s left sibling. | 0.1 |
| 81 | Maximum targeted coreference probability between $arg_f$ of $p_f$ and $iarg_n$ of $p$. This is a hybrid feature that calculates the coreference probability of Feature 33 using the corpus tuning method of Feature 13. | 0.1 |

Table A.4: Features for determining whether $c$ fills $iarg_n$ of predicate $p$. For each mention $f$ (denoting a $f$iller) in the coreference chain $c'$, $p_f$ and $arg_f$ are the predicate and argument position of $f$. Unless otherwise noted, all argument positions (e.g., $arg_n$ and $iarg_n$) should be interpreted as the integer label $n$ instead of the underlying word content of the argument. The & symbol denotes concatenation; for example, a feature value of "$p$ & $iarg_n$" for the $iarg_0$ position of *sale* would be "sale-0". Features marked with an asterisk are explained in Section 4.4.2 (p. 84). The *Score* column gives a heuristic ranking score for the features across all evaluation folds (see page 103 for discussion).

## A.6 Per-fold results for implicit argument identification

| | | Baseline | Discriminative (LibLinear) | | | | Oracle |
|---|---|---|---|---|---|---|---|
| Fold | Features | $F_1$ (%) | $c$ | $w+$ | $t$ | $F_1$ (%) | $F_1$ (%) |
| 1 | 1, 2, 3, 11, 32, 8, 27, 22, 31, 10, 20, 53, 6, 16, 24, 40, 30, 38, 72, 69, 73, 19, 28, 42, 48, 64, 44, 36, 37, 12, 7 | 31.7 | 0.25 | 4 | 0.39260 | 47.1 | 86.7 |
| 2 | 1, 3, 2, 4, 17, 13, 28, 11, 6, 18, 25, 12, 56, 29, 16, 53, 41, 31, 46, 10, 7, 51, 15, 22 | 32 | 0.25 | 256 | 0.80629 | 51.5 | 86.9 |
| 3 | 4, 3, 2, 8, 7, 6, 59, 20, 9, 62, 37, 39, 41, 19, 10, 15, 11, 35, 61, 44, 42, 40, 32, 30, 16, 75, 33, 24 | 35.3 | 0.25 | 256 | 0.90879 | 55.8 | 88.1 |
| 4 | 1, 2, 5, 13, 8, 49, 6, 35, 34, 14, 15, 18, 36, 28, 20, 45, 3, 43, 24, 48, 10, 29, 12, 30, 33, 65, 31, 22, 61, 16, 27, 41, 60, 55, 64 | 27.8 | 0.25 | 4 | 0.38540 | 45.8 | 86.5 |
| 5 | 1, 2, 26, 3, 4, 23, 5, 63, 55, 6, 12, 44, 42, 65, 7, 71, 18, 15, 10, 14, 52, 34, 19, 24, 50, 58 | 25.8 | 0.125 | 1024 | 0.87629 | 45.9 | 88 |
| 6 | 1, 3, 2, 14, 23, 38, 25, 39, 16, 6, 21, 68, 70, 58, 9, 22, 18, 31, 60, 10, 64, 15, 66, 19, 30, 51, 56, 28 | 34.8 | 0.25 | 256 | 0.87759 | 55.4 | 90.8 |
| 7 | 1, 2, 4, 3, 47, 54, 43, 7, 33, 9, 67, 24, 36, 50, 40, 12, 21 | 22.9 | 0.25 | 256 | 0.81169 | 46.3 | 87.4 |
| 8 | 1, 3, 2, 4, 9, 7, 14, 12, 6, 46, 30, 18, 19, 36, 48, 42, 37, 45, 60, 56, 61, 51, 15, 10, 41, 40, 25, 31, 11, 39, 62, 69, 34, 16, 33, 8, 38, 20, 78, 44, 55, 80, 53, 50, 52, 49, 24, 28, 57 | 27.1 | 0.0625 | 512 | 0.92019 | 47.4 | 87.2 |
| 9 | 1, 5, 2, 4, 3, 21, 27, 10, 15, 9, 57, 35, 16, 25, 37, 33, 45, 24, 46, 29, 19, 34, 51, 50, 22, 48, 32, 11, 12, 58, 41, 8, 76, 18, 30, 40, 77, 6, 66, 44, 43, 79, 81, 20 | 23 | 0.0625 | 32 | 0.67719 | 54.1 | 85.5 |
| 10 | 4, 3, 2, 17, 1, 13, 29, 12, 11, 52, 10, 15, 6, 16, 9, 22, 7, 21, 57, 19, 74, 34, 45, 20, 66 | 28.4 | 0.0625 | 512 | 0.89769 | 53.2 | 88.5 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Table A.5: Per-fold results for implicit argument identification. Columns are defined as follows: (1) fold used for testing, (2) selected features in rank order, (3) baseline $F_1$, (4) LibLinear cost parameter, (5) LibLinear weight for the positive class, (6) implicit argument confidence threshold, (7) discriminative $F_1$, (8) oracle $F_1$. A bias of 1 was used for all LibLinear models.

## A.7 Examples of implicit argument identification

Below, I provide two additional examples of implicit argument identification. The examples are drawn from the output of the implicit argument model described in Chapter 4.

**Example 1: Within-sentence implicit argument using preference information**

Consider the following sentence from article wsj_0308 in the Penn TreeBank:

(A.2) Sea Containers Ltd., in a long-awaited move to repel a [*Purpose* hostile takeover] [*Predicate* bid], said it will sell $1.1 billion of assets and use some of the proceeds to buy about 50% of [$iarg_1$ its common shares] for $70 apiece.

This sentence is describing a situation in which Sea Containers is being pursued by an outside company. The outside company is considering making an unsolicited bid for shares of Sea Containers. This move would effectively transfer ownership of Sea Containers to the outside company, and Sea Contains is trying to avoid this by purchasing a large number of its own shares.

With respect to the *bid* predicate in sentence A.2, we are looking for the $iarg_1$ (the entity being bid for). The answer, although present in the same sentence as the predicate, is not local to the predicate and is not identified by the standard nominal SRL system. Identification of *its common shares* as the $iarg_1$ relies primarily on selectional preference information: shares are often bid for. This fact is observed when training the selectional preference model.

**Example 2: Inter-sentence implicit argument using coreference information**

Consider the following sentences from article wsj_0286 in the Penn TreeBank:

(A.3) Nissan has increased earnings more than market share by cutting costs and by taking advantage of a general surge in Japanese car sales.

(A.4) But Nissan expects to earn only 120 billion yen in the current fiscal year, a modest increase of 4.7%.

(A.5) The big reason: For all its cost-cutting, [$iarg_0$ Nissan] remains less efficient than Toyota.

(A.6) In its last fiscal year, Nissan's profit represented just 2.3% of [Predicate 1 sales], compared with 4.3% at Toyota.

These sentences are describing a situation in which Nissan has increased sales but has not experienced a commensurate increase in profit due to its inefficiencies.

With respect to the *sale* predicate in sentence A.6, we are looking for the $iarg_0$ (the entity performing the selling). The implicit argument model has identified *Nissan* from A.5 as the filler of this argument position. This is an interesting inference because sentence A.5 does not contain much supporting evidence. The key is coreference: the system has identified many coreferent mentions of Nissan throughout the text, and these mentions participate in events (e.g., *earn* in A.3 and A.4 and *profit* in A.6) that are related to the *sale* event in the final sentence.

## A.8 Forward floating feature subset selection algorithm

$F$: set of $n$ features to select from
$T$: set of training instances containing all $n$ features
$V$: set of validation instances containing all $n$ features

**Algorithm**

```
B ← {}                                  # Best feature subset from F
Score_B ← −∞                            # Score of best feature subset
P ← {}                                  # All features used in previous round
while |F| > 0 do
  b ← null; Score_b ← −∞                # Track best new feature
  for all f ∈ F do
    Score_f ← eval(P ∪ {f}, T, V)       # Evaluate using previous/new feats.
    if Score_f > Score_b then
      b ← f; Score_b ← Score_f          # Update best new feature/score
    end if
  end for
  P ← P ∪ {b}; Score_P ← Score_b        # Update previous features
  F ← F − {b}                           # Remove best feature from pool
  if Score_P > Score_B then
    while |P| > 1 do
      r ← null; Score_r ← Score_P       # Track best feature to remove
      for all f ∈ P do
        Score_f ← eval(P − {f}, T, V)   # Evaluate after removing feature f
        if Score_f > Score_r then
          r ← f; Score_r ← Score_f      # Update best feature to remove
        end if
      end for
      if Score_r > Score_P then
        P ← P − {r}; Score_P ← Score_r  # Remove feature and update score
        F ← F ∪ {r}                     # Return removed feature to pool
      else
        break                           # No improvement from backtracking
      end if
    end while
    B ← P; Score_B ← Score_P            # Update best feature subset
  end if
end while
return B                                # Return best feature subset
```

# REFERENCES

# REFERENCES


Abend, O., Reichart, R., and Rappoport, A. (2009). Unsupervised argument identification for semantic role labeling. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 28–36, Suntec, Singapore. Association for Computational Linguistics.

ACE (2007). The ACE 2007 Evaluation Plan. NIST, 1.3a edition.

ACE (2008). The ACE 2008 Evaluation Plan. NIST, 1.2d edition.

Adger, D. (2003). Core Syntax. Oxford.

Baker, C., Fillmore, C., and Lowe, J. (1998). The Berkeley FrameNet project. In Boitet, C. and Whitelock, P., editors, Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In Proceedings of the 20th International Joint Conference on Artificial Intelligence.

Banko, M. and Etzioni, O. (2008). The tradeoffs between open and traditional relation extraction. In Proceedings of ACL-08: HLT, pages 28–36, Columbus, Ohio. Association for Computational Linguistics.

Berger, A., Pietra, V., and Pietra, S. (1996). A maximum entropy approach to natural language processing. Computational Linguistics, 22:39–71.

Bhagat, R., Pantel, P., and Hovy, E. (2007). LEDIR: An unsupervised algorithm for learning directionality of inference rules. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 161–170, Prague, Czech Republic. Association for Computational Linguistics.

Bikel, D. M., Schwartz, R., and Weischedel, R. M. (1999). An algorithm that learns what's in a name. Mach. Learn., 34(1-3):211–231.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022.

Bos, J. (2005). Towards wide-coverage semantic interpretation. In Proceedings of the Sixth International Workshop on Computational Semantics, pages 42–53.

Burchardt, A., Frank, A., and Pinkal, M. (2005). Building text meaning representations from contextually related frames - a case study. In Proceedings of the Sixth International Workshop on Computational Semantics.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2:121167.

Carpenter, P. A., Miyake, A., and Just, M. A. (1995). Language comprehension: Sentence and discourse processing. Annu. Rev. Psychol., 46:91–120.

Carreras, X. and Màrquez, L. (2004). Introduction to the conll-2004 shared task: Semantic role labeling. In Proceedings of the Conference on Computational Natural Language Learning.

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling.

Chambers, N. and Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In Proceedings of the Association for Computational Linguistics, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Charniak, E., Goldwater, S., and Johnson, M. (1998). Edge-based best-first chart parsing. In Sixth Workshop on Very Large Corpora, pages 127–133.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.

Chen, B., Su, J., and Tan, C. L. (2010). Resolving event noun phrases to their verbal mentions. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 872–881, Cambridge, MA. Association for Computational Linguistics.

Chen, Z. and Ji, H. (2009). Graph-based event coreference resolution. In Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4), pages 54–57, Suntec, Singapore. Association for Computational Linguistics.

Chinchor, N., Lewis, D. D., and Hirschmant, L. (1993). Evaluating message understanding systems: An analysis of the third message understanding conference. Computational Linguistics, 19(3):409–450.

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1):3746.

Copestake, A. and Flickinger, D. (2000). An open-source grammar development environment and broad-coverage english grammar using hpsg. In Proc. LREC-2000.

Dahl, D. A., Palmer, M. S., and Passonneau, R. J. (1987). Nominalizations in pundit. In Proceedings of the 25th annual meeting on Association for Computational Linguistics, pages 131–139, Morristown, NJ, USA. Association for Computational Linguistics.

Dang, H. T., Kelly, D., and Lin, J. J. (2007). Overview of the trec 2007 question answering track. In TREC.

Daumé, H., Deoskar, T., McClosky, D., Plank, B., and Tiedemann, J., editors (2010). Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing. Association for Computational Linguistics, Uppsala, Sweden.

Deschacht, K. and Moens, M.-F. (2009). Semi-supervised semantic role labeling using the Latent Words Language Model. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 21–29, Singapore. Association for Computational Linguistics.

Di Eugenio, B. and Glass, M. (2004). The kappa statistic: a second look. Comput. Linguist., 30(1):95–101.

Dowty, D. (1991). Thematic proto-roles and argument selection. Language, 67:547–619.

Efron, B. and Tibshirani, R. J. (1993). An Introduction to the Bootstrap. Chapman & Hall, New York.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research, 9:1871–1874.

Fellbaum, C. (1998). WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press.

Fillmore, C. (1968). The case for case. In Bach, E. and Harms, R., editors, Universals in Linguistic Theory. Holt, Rinehart, and Winston.

Fillmore, C. (1976). Frame semantics and the nature of language. In Harnad, S., Steklis, H., and Lancaster, J., editors, Origins and Evolution of Language and Speech. The New York Academy of Sciences.

Fillmore, C. and Baker, C. (2001). Frame semantics for text understanding. In Proceedings of WordNet and Other Lexical Resources Workshop, NAACL.

Fürstenau, H. and Lapata, M. (2009). Graph alignment for semi-supervised semantic role labeling. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 11–20, Singapore. Association for Computational Linguistics.

Gerber, M. and Chai, J. (2010). Beyond NomBank: A study of implicit arguments for nominal predicates. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1583–1592, Uppsala, Sweden. Association for Computational Linguistics.

Gerber, M., Chai, J., and Meyers, A. (2009). The role of implicit argumentation in nominal SRL. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 146–154, Boulder, Colorado. Association for Computational Linguistics.

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

Gildea, D. and Palmer, M. (2001). The necessity of parsing for predicate argument recognition. In ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 239–246, Morristown, NJ, USA. Association for Computational Linguistics.

Gillick, D. (2009). Sentence Boundary Detection and the Problem with the U.S. In Proceedings of NAACL: Short Papers.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D. (2007). Semeval-2007 task 04: Classification of semantic relations between nominals. In Proceedings of the 4th International Workshop on Semantic Evaluations.

Gordon, A. and Swanson, R. (2007). Generalizing semantic role annotations across syntactically similar verbs. In Proceedings of ACL, pages 192–199.

Gorn, S. (1967). Explicit definitions and linguistic dominoes. In Hart, J., editor, Systems and Computer Science, pages 77–115. University of Toronto Press, Toronto Canada.

Graesser, A. C. and Clark, L. F. (1985). Structures and Procedures of Implicit Knowledge. Ablex Publishing Corporation.

Graff, D. (2003). English Gigaword. Linguistic Data Consortium, Philadelphia.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. Computational Linguistics, 21(2):203–225.

Gruber, J. (1965). Studies in Lexical Relations. PhD thesis, MIT.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Harris, Z. (1985). Distributional structure. In Katz, J. J., editor, The Philosophy of Linguistics, pages 26–47. New York: Oxford University Press.

Heim, I. and Kratzer, A. (1998). Semantics in Generative Grammar. Blackwell, Oxford.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2010). Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.

Hirst, G. (1987). Semantic Interpretation and the Resolution of Ambiguity. Cambridge University Press.

Hsu, C.-W., Chang, C.-C., , and Lin, C.-J. (2010). A practical guide to support vector classi cation. Technical report, National Taiwan University.

Hull, R. and Gomez, F. (1996). Semantic interpretation of nominalizations. In Proceedings of AAAI.

Iida, R., Komachi, M., Inui, K., and Matsumoto, Y. (2007). Annotating a Japanese text corpus with predicate-argument and coreference relations. In Proceedings of the Linguistic Annotation Workshop in ACL-2007, page 132139.

Imamura, K., Saito, K., and Izumi, T. (2009). Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 85–88, Suntec, Singapore. Association for Computational Linguistics.

Jiang, J. and Zhai, C. (2006). Exploiting domain structure for named entity recognition. In Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 74–81, Morristown, NJ, USA. Association for Computational Linguistics.

Jiang, Z. and Ng, H. (2006). Semantic role labeling of nombank: A maximum entropy approach. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.

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

Joshi, M., Pakhomov, S., Pedersen, T., Maclin, R., and Chute, C. (2006). An end-to-end supervised target-word sense disambiguation system. In Proceedings of the Twenty-first National Conference on Artificial Intelligence, pages 1941–1942.

Kaisser, M. and Webber, B. (2007). Question answering based on semantic roles. In ACL 2007 Workshop on Deep Linguistic Processing, pages 41–48, Prague, Czech Republic. Association for Computational Linguistics.

Kamp, H. and Reyle, U. (1993). From Discourse to Logic. Kluwer, Dordrecht.

Kingsbury, P. and Palmer, M. (2003). Propbank: the next level of treebank. In Proceedings of Treebanks and Lexical Theories.

Kipper, K. (2005). VerbNet: A broad-coverage, comprehensive verb lexicon. PhD thesis, Department of Computer and Information Science University of Pennsylvania.

Kipper, K., Dang, H. T., and Palmer, M. (2000). Class-based construction of a verb lexicon. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 691–696. AAAI Press / The MIT Press.

Krippendorff, K. (1980). Content Analysis: An Introduction to Its Methodology. Sage Publications.

Kučera, H. and Nelson, F. W. (1967). Computational Analysis of Present-day American English. Brown University Press, Providence, RI.

Lang, J. and Lapata, M. (2010). Unsupervised induction of semantic roles. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 939–947, Los Angeles, California. Association for Computational Linguistics.

Lapata, M. (2000). The automatic interpretation of nominalizations. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 716–721. AAAI Press / The MIT Press.

Levin, B. (1993). English verb classes and alternations: A preliminary investigation. Chicago University Press.

Lin, D. and Pantel, P. (2001). Discovery of inference rules for question-answering. Nat. Lang. Eng., 7(4):343–360.

Liu, C. and Ng, H. (2007). Learning predictive structures for semantic role labeling of nombank. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 208–215, Prague, Czech Republic. Association for Computational Linguistics.

Liu, X., Han, B., Li, K., Stiller, S. H., and Zhou, M. (2010). SRL-based verb selection for ESL. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 1068–1076, Cambridge, MA. Association for Computational Linguistics.

Macleod, C., Grishman, R., Meyers, A., Barrett, L., and Reeves, R. (1998). Nomlex: A lexicon of nominalizations. In Proceedings of the Eighth International Congress of the European Association for Lexicography.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn TreeBank. Computational Linguistics, 19:313–330.

May, J. and Knight, K. (2007). Syntactic re-alignment models for machine translation. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 360–368, Prague, Czech Republic. Association for Computational Linguistics.

Meyers, A. (2007a). Annotation guidelines for NomBank - noun argument structure for PropBank. Technical report, New York University.

Meyers, A. (2007b). Those other nombank dictionaries. Technical report, New York University.

Meyers, A., Macleod, C., Yangarber, R., Grishman, R., Barrett, L., and Reeves, R. (1998). Using nomlex to produce nominalization patterns for information extraction. In Proceedings of the COLING-ACL Workshop on the Computational Treatment of Nominals.

Mooney, R. J. (2007). Learning for semantic parsing. In Proceedings of the 8th International Conference, CICLing.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. Comput. Linguist., 34(2):193–224.

Nielsen, L. A. (2004). Verb phrase ellipsis detection using automatically parsed text. In COLING '04: Proceedings of the 20th international conference on Computational Linguistics, page 1093, Morristown, NJ, USA. Association for Computational Linguistics.

Nielsen, L. A. (2005). A corpus-based study of Verb Phrase Ellipsis Identification and Resolution. PhD thesis, King's College.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT), pages 149–160.

Padó, S., Pennacchiotti, M., and Sporleder, C. (2008). Semantic role assignment for event nominalisations by leveraging verbal data. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 665–672, Manchester, UK. Coling 2008 Organizing Committee.

Palmer, M. S., Dahl, D. A., Schiffman, R. J., Hirschman, L., Linebarger, M., and Dowding, J. (1986). Recovering implicit information. In Proceedings of the 24th annual meeting on Association for Computational Linguistics, pages 10–19, Morristown, NJ, USA. Association for Computational Linguistics.

Pantel, P., Bhagat, R., Coppola, B., Chklovski, T., and Hovy, E. (2007). ISP: Learning inferential selectional preferences. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pages 564–571, Rochester, New York. Association for Computational Linguistics.

Pantel, P. and Ravichandran, D. (2004). Automatically labeling semantic classes. In Susan Dumais, D. M. and Roukos, S., editors, HLT-NAACL 2004: Main Proceedings, pages 321–328, Boston, Massachusetts, USA. Association for Computational Linguistics.

Pizzato, L. A. and Mollá, D. (2008). Indexing on semantic roles for question answering. In Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering, pages 74–81, Manchester, UK. Coling 2008 Organizing Committee.

Pradhan, S. S., Ward, W., and Martin, J. H. (2008). Towards robust semantic role labeling. Comput. Linguist., 34(2):289–310.

Prasad, R., Lee, A., Dinesh, N., Miltsakaki, E., Campion, G., Joshi, A., and Webber, B. (2008). Penn discourse treebank version 2.0. Linguistic Data Consortium.

Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating search methods in feature selection. Pattern Recognition Letters, 15:1119–1125.

Punyakanok, V., Roth, D., and tau Yih, W. (2005). The necessity of syntactic parsing for semantic role labeling. In International Joint Conference on Artificial Intelligence.

Punyakanok, V., Roth, D., and Yih, W.-t. (2008). The importance of syntactic parsing and inference in semantic role labeling. Comput. Linguist., 34(2):257–287.

Pustejovsky, J. (1995). The Generative Lexicon. The MIT Press.

Resnik, P. (1996). Selectional constraints: An information-theoretic model and its computational realization. Cognition, 61:127–159.

Ritter, A., Mausam, and Etzioni, O. (2010). A latent dirichlet allocation method for selectional preferences. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.

Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C., and Palmer, M. (2009). Semeval-2010 task 10: Linking events and their participants in discourse. In Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009), pages 106–111, Boulder, Colorado. Association for Computational Linguistics.

Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C., and Palmer, M. (2010). Semeval-2010 task 10: Linking events and their participants in discourse. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 45–50, Uppsala, Sweden. Association for Computational Linguistics.

Sagae, K. (2009). Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09), pages 81–84, Paris, France. Association for Computational Linguistics.

Sagae, K. and Lavie, A. (2005). A classifier-based parser with linear run-time complexity. In Proceedings of the International Workshop on Parsing Technologies.

Sanford, A. J. (1981). Understanding Written Language. John Wiley & Sons Ltd.

Sasano, R., Kawahara, D., and Kurohashi, S. (2004). Automatic construction of nominal case frames and its application to indirect anaphora resolution. In Proceedings of Coling 2004, pages 1201–1207, Geneva, Switzerland. COLING.

Schank, R. C. and Abelson, R. P. (1977). Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures. L. Erlbaum, Hillsdale, NJ.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Swier, R. S. and Stevenson, S. (2004). Unsupervised semantic role labelling. In Lin, D. and Wu, D., editors, Proceedings of Empirical Methods in Natural Language Processing, pages 95–102, Barcelona, Spain. Association for Computational Linguistics.

Szpektor, I., Tanev, H., Dagan, I., and Coppola, B. (2004). Scaling web-based acquisition of entailment relations. In Proceedings of Empirical Methods in Natural Language Processing.

Taboada, M. and Mann, W. C. (2006). Rhetorical structure theory: looking back and moving ahead. Discourse Studies, 8:423–459.

Tonelli, S. and Delmonte, R. (2010). Venses++: Adapting a deep semantic processing system to the identification of null instantiations. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 296–299, Uppsala, Sweden. Association for Computational Linguistics.

Toutanova, K., Haghighi, A., and Manning, C. D. (2005). Joint learning improves semantic role labeling. In ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 589–596, Morristown, NJ, USA. Association for Computational Linguistics.

Toutanova, K., Haghighi, A., and Manning, C. D. (2008). A global joint model for semantic role labeling. Comput. Linguist., 34(2):161–191.

van Dijk, T. A. (1977). Semantic macro structures and knowledge frames in discourse comprehension. In Just, M. A. and Carpenter, P. A., editors, Cognitive Processes in Comprehension, pages 3–32. Lawrence Erlbaum.

van Dijk, T. A. and Kintsch, W. (1983). Strategies of Discourse Comprehension. Academic Press.

Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., and Pustejovsky, J. (2007). Semeval-2007 task 15: Tempeval temporal relation identification. In Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007), pages 75–80, Prague, Czech Republic. Association for Computational Linguistics.

Versley, Y., Ponzetto, S. P., Poesio, M., Eidelman, V., Jern, A., Smith, J., Yang, X., and Moschitti, A. (2008). BART: A modular toolkit for coreference resolution. In Proceedings of the 6th International Conference on Language Resources and Evaluation, Marrakech, Morocco.

Weischedel, R. and Brunstein, A. (2005). Bbn pronoun coreference and entity type corpus. Linguistic Data Consortium.

Whittemore, G., Macpherson, M., and Carlson, G. (1991). Event-building through role-filling and anaphora resolution. In Proceedings of the 29th annual meeting on Association for Computational Linguistics, pages 17–24, Morristown, NJ, USA. Association for Computational Linguistics.

Wilson, N. L. (1974). Facts, events, and their identity conditions. Philosophical Studies, 25:303–321.

Wu, Z. and Palmer, M. (1994). Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pages 133–138, Las Cruces, New Mexico, USA. Association for Computational Linguistics.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In Proceedings of EMNLP.

Yang, X., Su, J., and Tan, C. L. (2008). A twin-candidate model for learning-based anaphora resolution. Comput. Linguist., 34(3):327–356.

Yi, S., Loper, E., and Palmer, M. (2007). Can semantic roles generalize across genres? In Proceedings of NAACL HLT.

Zanzotto, F. M., Pennacchiotti, M., and Pazienza, M. T. (2006). Discovering asymmetric entailment relations between verbs using selectional preferences. In ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pages 849–856, Morristown, NJ, USA. Association for Computational Linguistics.