Semantic Role Labeling of Implicit Arguments for Nominal Predicates

Matthew Gerber*
University of Virginia

Joyce Y. Chai**
Michigan State University

Nominal predicates often carry implicit arguments. Recent work on semantic role labeling has focused on identifying arguments within the local context of a predicate; implicit arguments, however, have not been systematically examined. To address this limitation, we have manually annotated a corpus of implicit arguments for ten predicates from NomBank. Through analysis of this corpus, we find that implicit arguments add 71% to the argument structures that are present in NomBank. Using the corpus, we train a discriminative model that is able to identify implicit arguments with an F_1 score of 50%, significantly outperforming an informed baseline model. This article describes our investigation, explores a wide variety of features important for the task, and discusses future directions for work on implicit argument identification.

1. Introduction

Recent work has shown that semantic role labeling (SRL) can be applied to nominal predicates in much the same way as verbal predicates (Liu and Ng 2007; Johansson and Nugues 2008; Gerber, Chai, and Meyers 2009). In general, the nominal SRL problem is formulated as follows: Given a predicate that is annotated in NomBank as bearing arguments, identify these arguments within the clause or sentence that contains the predicate. As shown in our previous work (Gerber, Chai, and Meyers 2009), this problem definition ignores the important fact that many nominal predicates do not bear arguments in the local context. Such predicates need to be addressed in order for nominal SRL to be used by downstream applications such as automatic question answering, information extraction, and statistical machine translation.

Gerber, Chai, and Meyers (2009) showed that it is possible to accurately identify nominal predicates that bear arguments in the local context. This makes the nominal SRL system applicable to text that does not contain annotated predicates. The system does not address a fundamental question regarding arguments of nominal predicates, however: If an argument is missing from the local context of a predicate, might the argument be located somewhere in the wider discourse? Most prior work on nominal

* 151 Engineer's Way, University of Virginia, Charlottesville, VA 22904.
E-mail: matt.gerber@virginia.edu.
** 3115 Engineering Building, Michigan State University, East Lansing, MI 48824.
E-mail: jchai@cse.msu.edu.

Submission received: 4 August 2011; revised version received: 23 December 2011; accepted for publication: 7 February 2012.

and verbal SRL has stopped short of answering this question, opting instead for an approach that only labels local arguments and thus ignores predicates whose arguments are entirely non-local. This article directly addresses the issue of non-local (or implicit) argument identification for nominal predicates.

As an initial example, consider the following sentence, which is taken from the Penn TreeBank (Marcus, Santorini, and Marcinkiewicz 1993):

(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 (Meyers 2007) role set for *requirement* is shown here:

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 (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; a reasonable interpretation of the sentence, however, is that *SEC* is the entity that is requiring something.[1] This article refers to arguments such as *SEC* in Example (1) as implicit. In this work, the notion of implicit argument covers any argument that is not annotated by NomBank.[2]

Building on Example (1), consider the following sentence, which directly follows Example (1) in the corresponding TreeBank document:

(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 as follows:

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, Example (2) shows the local argument labels provided by NomBank. These labels only indicate that rules have been changed. For a full interpretation, Example (2) requires an understanding of Example (1). Without

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

2 NomBank annotates arguments in the noun phrase headed by the predicate as well as arguments brought in by so-called support verb structures. See Meyers (2007) for details.

the sentence from Example 1, the reader has no way of knowing that *the agency* in Example (2) actually refers to the same entity as *SEC* in Example (1). As part of the reader's comprehension process, this entity is identified as the filler for the arg_0 role in Example (2). This identification must occur in order for these two sentences to form a coherent discourse.

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 (1) and (2):

(3) Who changed the rules regarding reporting requirements?

Question (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 Question Answering (QA) Track (Dang, Kelly, and Lin 2007). Using the evaluation data from this track, Pizzato and Mollá (2008) showed that SRL can improve the accuracy of a QA system; a traditional SRL system alone, however, is not enough to recover the implied answer to Question (3): *SEC* or *the agency*. Successful implicit argument identification provides the answer in this case.

This article presents an in-depth study of implicit arguments for nominal predicates.[3] The following section surveys research related to implicit argument identification. Section 3 describes the study's implicit argument annotation process and the data it produced. The implicit argument identification model is formulated in Section 4 and evaluated in Section 5. Discussion of results is provided in Section 6, and the article concludes in Section 7.

2. Related Work

The research presented in this article is related to a wide range of topics in cognitive science, linguistics, and natural language processing (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 high marks in an evaluation (for examples, see Carreras and Màrquez 2005 and Surdeanu et al. 2008); implicit argumentation, however, 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.

2.1 Discourse Comprehension in Cognitive Science

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

3 This article builds on our previous work (Gerber and Chai 2010).

to form coherent textual discourses. Consider the following examples, borrowed from Sanford (1981, page 5):

(4) Jill came bouncing down the stairs.

(5) Harry rushed off to get the doctor.

Examples (4) and (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) and (5):

(6) Jill came bouncing down the stairs.

(7) Harry rushed over to kiss her.

The interpretation of Example (6) is vastly different from the interpretation of Example (4). In Example (4), Jill becomes injured whereas in Example (6) she is quite happy.

Examples (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 article 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 (4–7) 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) and (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, page 4), van Dijk and Kintsch (1983, page 303), Graesser and Clark (1985, page 14), and Carpenter, Miyake, and Just (1995). Inspired by these ideas, the model developed in this article 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.

2.2 Automatic Relation Discovery

Examples (4) and (5) in the previous section show that understanding the relationships between predicates is a key part of understanding a textual discourse. In this section, we 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:

(8) *X eats Y* ↔ *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*; the reader might need to infer this in order to make sense of the fact that *X* left a large tip for the waiter, however.

Lin and Pantel (2001) created mappings such as the one in Example (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 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 (8), the two dependency paths are as follows:

$$X \xleftarrow{subject} eats \xrightarrow{object} Y$$
$$X \xleftarrow{subject} likes \xrightarrow{object} Y$$

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 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, Pantel, and Hovy (2007) extended the work of Lin and Pantel (2001) to handle cases of asymmetric relationships. The basic idea proposed by Bhagat, Pantel, and Hovy 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 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, Pantel, and Hovy, the system built around this intuition is able to effectively identify the directionality of many inference rules.

Zanzotto, Pennacchiotti, and Pazienza (2006) presented another study aimed at identifying asymmetric relationships between verbs. For example, the asymmetric entailment relationship *X wins* → *X plays* holds, but the opposite (*X plays* → *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, Pennacchiotti, and Pazienza [2006]):

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

The underlying idea behind the authors' approach is that asymmetric relationships such as *X wins* → *X plays* are often entailed by constructions involving agentive, nominalized verbs as the logical subjects of the main verb. In Example (9), 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, Pennacchiotti, and Pazienza (2006) 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., Szpektor et al. 2004, Pantel et al. 2007) have been conducted that are similar to that work. 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 (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. 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 article, we propose methods of automatically acquiring these fine-grained relationships for verbal and nominal predicates using existing corpora. We also propose a method of using these relationships to recover implicit arguments.

2.3 Coreference Resolution

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:

(10) "Carpet King sales are up 4% this year," said owner Richard Rippe.

(11) 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 (11) refers to the same entity as *Richard Rippe* in Example (10). 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 (NIST 2008) has provided a test environment for systems designed to identify these and other coreference relations. Systems based on the ACE data sets typically take a supervised learning approach to coreference resolution in general (Versley et al. 2008) and pronominal anaphor in particular (Yang, Su, and Tan 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, Saito, and Izumi 2009). Sasano, Kawahara, and Kurohashi (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, Joshi, and Weinstein 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.

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 that 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. [1986]):

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

(13) Has select lock.

Example (13) does not specify precisely which entity has select lock. The domain knowledge, however, 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 (12) has select lock.

A similar line of work was pursued by Whittemore, Macpherson, and Carlson (1991), who offer the following example of implicit argumentation (page 21):

(14) Pete bought a car.

(15) The salesman was a real jerk.

In Example (14), the *buy* event is not associated with an entity representing the seller. This entity is introduced in Example (15) as *the salesman*, whose semantic properties satisfy the requirements of the *buy* event. Whittemore, Macpherson, and Carlson (1991) build up the event representation incrementally using a combination of semantic property constraints and DRT.

The systems developed by Palmer et al. (1986) and Whittemore, Macpherson, and Carlson (1991) 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, Mausam, and Etzioni (2010) being some of the most recent. The model developed in

the current article uses a variety of selectional preference measures to identify implicit arguments.

The implicit argument identification systems described herein 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. Subsequently, we 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. 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; the authors did not provide concrete methods to perform the analysis automatically, however.

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

(16) John kicked the ball.

(17) Bill [did], too.

The bracketed text in Example (17) is a placeholder for the verb phrase *kicked the ball* in Example (16), which has been elided (i.e., left out). Thus, in Example (17), *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 agent (Bill) would be recovered.[4] Nielsen (2004) created a system able to detect the presence of ellipses, producing the bracketing in Example (17). 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, Frank, and Pinkal (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, Frank, and Pinkal (2005) propose to learn frame element linking patterns from observed data; the authors did not implement and evaluate such a method, however. Building on the work of Burchardt, Frank, and Pinkal, this article presents a model of implicit arguments that uses a quantitative analysis of naturally occurring coreference patterns.

In our previous work, we demonstrated the importance of filtering out nominal predicates that take no local arguments (Gerber, Chai, and Meyers 2009). This approach leads to appreciable gains for certain nominals. The approach does not attempt to actually recover implicit arguments, however.

4 Identification of the implicit patient in Example (17) (the ball) should be sensitive to the phenomenon of sense anaphora. If Example (16) was changed to "a ball," then we would have no implicit patient in Example (17).

Most recently, Ruppenhofer et al. (2009) proposed 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 article 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. We 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, we describe the implicit argument annotation process and resulting data set.

3. Implicit Argument Annotation and Analysis

As shown in the previous section, the existence of implicit arguments has been recognized for quite some time. This type of information, however, was not formally annotated until Ruppenhofer et al. (2010) conducted their SemEval task on implicit argument identification. There are two reasons why we chose to create an independent data set for implicit arguments. The first reason is the aforementioned sparsity of the SemEval data set. The second reason is that the SemEval data set is not built on top of the Penn TreeBank, which is the gold-standard syntactic base for all work in this article. 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.

3.1 Data Annotation

3.1.1 Predicate Selection. Implicit arguments are a relatively new subject of annotation in the field. To effectively use our limited annotation resources and allow the observation of interesting behaviors, we decided to focus 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 article 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 predicate indicated by the "%" symbol). 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. We 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, we first calculated N_p, the average number of roles expressed by p in NomBank. We then calculated V_p, the average number of roles expressed by the verb form of p in PropBank. We 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:

(18) *John loaned (the money to Mary).

(19) *John invested (his money).

Examples (18) and (19) indicate that certain arguments must explicitly accompany *loan* and *invest*. In nominal form, these predicates can exist without such arguments and still be grammatical:

(20) John's loan was not repaid.

(21) John's investment was huge.

Note, however, that Examples (20) and (21) 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 criteria 3 and 4. We then selected the top ten, which are shown in the first column of Table 1. The role sets (i.e., argument definitions) for these predicates can be found in the Appendix on page 790.

3.1.2 Annotation Procedure. We annotated implicit arguments for instances of the ten selected nominal predicates. The annotation process proceeded document-by-document. For a document d, we 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 1.

Table 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.

			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)

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.[5]
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. When possible, match the textual bounds of an implicit argument to the textual bounds 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 article, we will use $iarg_n$ to refer to an implicit argument position n. We will use arg_n to refer to an argument provided by PropBank or NomBank. We will use p to mark predicate instances. Example (22) provides a sample annotation for an instance of the *investment* predicate:

(22) [$iarg_0$ 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; 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, however.

Of course, not all implicit argument decisions are as easy as those in Example (22). Consider the following contrived example:

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

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

Example (24) contains one mention of the $iarg_0$ (the agentive planner). It might be tempting to also mark *Coke* in Example (23) as an additional $iarg_0$; the only reasonable interpretation of *Coke* in 23 is as a consumable fluid, however. Fluids cannot plan things, so this annotation should not be performed. This is a case of sense ambiguity between *Coke* as a company and *Coke* as a drink. In all such cases, the annotator was asked to infer the proper sense before applying an implicit argument label.

Lastly, it should be noted that we 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, Roth, and Yih (2008) model this constraint explicitly to arrive at the final label assignment; as the

5 See Appendix A for the list of role sets used in this study.

following example shows, however, this constraint should not be applied to implicit arguments:

(25) 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 (25) is a perfectly reasonable implicit argument (the seller) for the marked predicate. Systems should be required to identify such arguments; thus, we included them in our annotations.

3.1.3 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 previously, we compared our annotations to those provided by an undergraduate linguistics student who, after a brief training period, re-annotated a portion of the data set. 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 our 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 1.

We 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 κ 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 κ will be negative. As noted by Di Eugenio and Glass (2004), researchers have devised different scales to assess κ. Many NLP researchers use the scale created by Krippendorff (1980):

$\kappa < 0.67$	low agreement
$0.67 \leq \kappa < 0.8$	moderate agreement
$\kappa \geq 0.8$	strong agreement

Di Eugenio and Glass (2004) note, however, that this scale has not been rigorously defended, even by Krippendorff (1980) himself.

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

$$p_o = \frac{\sum_{iarg_n} \text{agree}(iarg_n)}{N}$$

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

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 the argument label n to a filler, 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 (1), 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 these values for p_o and p_c, Cohen's kappa indicated an agreement of 64.3%. According to the scale of Krippendorff (1980), 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, Cohen'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. Subsequently, we demonstrate this situation with actual data. First, we present our annotations for two sentences from the same document:

(26) Shares of UAL, the parent of [$iarg_1$ United Airlines], were extremely active all day Friday, reacting to news and rumors about the proposed [$iarg_2$ $6.79 billion] buy-out of [$iarg_1$ the airline] by an employee–management group.

(27) And 10 minutes after the UAL trading halt came news that the UAL group couldn't get financing for [arg_0 its] [p bid].

In Example (27), the predicate is marked along with the explicit arg_0 argument. Our task is to locate the implicit $iarg_1$ (the entity bid for) and the implicit $iarg_2$ (the amount of the bid). We were able to locate these entities in a previous sentence (Example (26)). Next, we present the second annotator's (i.e., the student's) annotations for the same two sentences:

(28) Shares of UAL, the parent of [$iarg_1$ United Airlines], were extremely active all day Friday, reacting to news and rumors about the proposed $6.79 billion buy-out of [$iarg_1$ the airline] by an employee–management group.

(29) And 10 minutes after the UAL trading halt came news that the UAL group couldn't get financing for [arg_0 its] [p bid].

As shown in Example (28), the second annotator agreed with our identification of the $iarg_1$; the second annotator did not mark an implicit $iarg_2$, however, despite the fact that it can be inferred. We believe this type of error can be addressed with additional training. The student's annotations were only used to compute agreement. We performed all training and evaluation using randomized cross-validation over the annotations we created.

3.2 Annotation Analysis

We carried out this annotation process on the standard training (2–21), development (24), and testing (23) sections of the Penn TreeBank. Table 1 summarizes the results. In this section, we highlight key pieces of information found in this table.

3.2.1 Implicit Arguments are Frequent. Column (3) of Table 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 6).

3.2.2 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 (22) is 0/3 for NomBank-only arguments and 3/3 when the annotated implicit arguments are also considered. Returning to Table 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 we annotated.

3.2.3 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 (5) in Table 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 (5) and (6)). We hypothesized that this difference might indicate the presence of approximately one implicit argument per predicate instance. This hypothesis is confirmed by comparing columns (6) and (8): When considering implicit arguments, many nominal predicates express approximately the same number of arguments on average as their verbal counterparts.

3.2.4 Most Implicit Arguments Are Nearby. In addition to these analyses, we examined the location of implicit arguments in the discourse. Figure 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 4–6 sentences for resolution. These observations are important; they show that searching too far back in the discourse is likely to

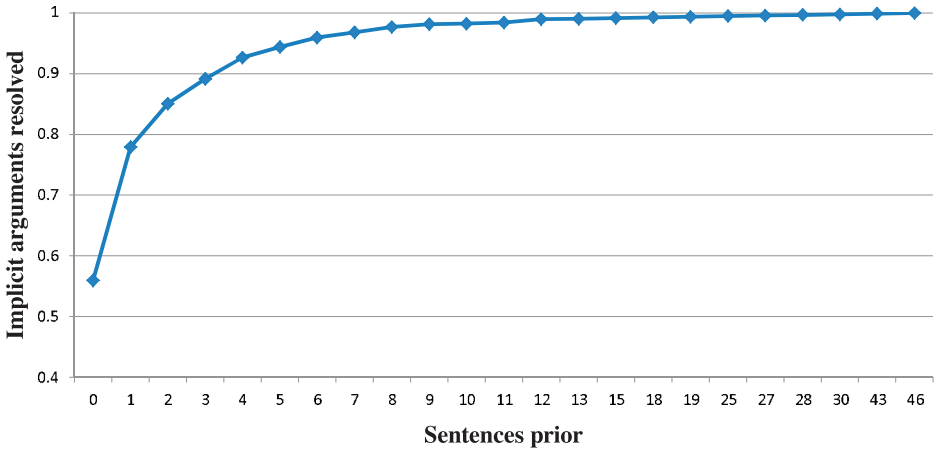

Figure 1
Location of implicit arguments. Of all implicitly filled argument positions, the *y*-axis indicates the percentage that are filled at least once within the number of sentences indicated by the *x*-axis (multiple fillers may exist for the same position).

produce many false positives without a significant increase in recall. Section 6 discusses additional implications of this skewed distribution.

4. Implicit Argument Model

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:

(30) [Mexico] desperately needs investment.

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

(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*,

Table 2
Primary feature groups used by the model. The third column gives the number of features in the group, and the final column gives the number of features from the group that were ranked in the top 20 among all features.

Feature group	Resources used	#	Top 20
(1) Textual semantics	PropBank, NomBank	13	4
(2) Ontologies	FrameNet, VerbNet, WordNet	8	4
(3) Filler-independent	Penn TreeBank	35	7
(4) Corpus statistics	Gigaword, Verbal SRL, Nominal SRL	9	1
(5) Textual discourse	Penn Discourse Bank	1	0
(6) Other	Penn TreeBank	15	4

one can draw evidence from other mentions in c's coreference chain. Example (30) states that Mexico needs investment. Example (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 (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.[6] We 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 article, we will refer to c as the **primary filler**, differentiating it from other mentions in the coreference chain c'. This distinction is necessary because our evaluation requires the model to select at most one filler (i.e., c) for each missing argument position. In the following section, we present the feature set used to represent each three-tuple within the classification function.

4.2 Model Features

Appendix Table B.1 lists all features used by the model described in the previous section. For convenience, Table 2 presents a high-level grouping of the features and the resources used to compute them. The broadest distinction to be made is whether a feature depends on elements of c'. Features in **Group 3** do not, whereas all others do. The features in **Group 3** characterize the predicate–argument position being filled (p and $iarg_n$), independently of the candidate filler. This group accounts for 43% of the features and 35% of those in the top 20.[7] The remaining features depend on elements of c' in some way. **Group 1** features characterize the tuple using the SRL propositions contained in the text being evaluated. **Group 2** features place the $\langle p, iarg_n, c' \rangle$ tuple into a manually constructed ontology and compute a value based on the structure of that ontology. **Group 4** features compute statistics of the tuple within a large corpus of semantically analyzed text. **Group 5** contains a single feature that captures the discourse structure properties of the tuple. **Group 6** contains all other features, most of which capture the syntactic relationships between elements of c' and p. In the following sections, we provide detailed examples of features from each group shown in Table 2.

6 We used OpenNLP for coreference identification: `http://opennlp.sourceforge.net`.
7 Features were ranked according to the order in which they were selected during feature selection (Section 5.3 for details).

4.2.1 Group 1: Features Derived from the Semantic Content of the Text. **Feature 1** was often selected first by the feature selection algorithm. This feature captures the semantic properties of the candidate filler c' and the argument position being filled. Consider the following Penn TreeBank sentences:

(33) [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 1 takes a value of $produce - arg_0 - ship - arg_0$. This value, which is derived from the text itself, asserts that producers are also shippers. To reduce data sparsity, we generalized the predicates to their WordNet synset IDs (creating Feature 4). We also generalized the predicates and arguments to their VerbNet thematic roles using SemLink (creating Feature 23). Although the generalized features rely on ontologies, they do so in a trivial way that does not take advantage of the detailed structure of the ontologies. Such structure is used by features in the next group.

4.2.2 Group 2: Features Derived from Manually Constructed Ontologies. **Feature 9** captures the semantic relationship between predicate–argument positions by examining paths between frame elements in FrameNet. SemLink[8] 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.

The value of Feature 9 has the following general form:

(34) $Frame_1.FE_1 \xrightarrow{Rel_1} Frame_2.FE_2 \xrightarrow{Rel_2} \ldots \xrightarrow{Rel_{n-1}} Frame_n.FE_n$

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:

(35) Sell.Goods $\xrightarrow{\text{Inherits}}$ Give.Theme $\xrightarrow{\text{Causes}}$ Get.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 give scenario (Give.Theme) that can also be viewed as a get scenario (Get.Theme) in which the buyer receives something (Buy.Goods). This complex world knowledge is represented compactly using the relationships defined in FrameNet. In our experiments, we searched all possible frame element paths of length five or less that use the following relationships:

- Causative–of

8 http://verbs.colorado.edu/semlink.

- Inchoative–of
- Inherits
- Precedes
- Subframe–of

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

(36) Consumers bought many [*c* cars] this year at reduced prices.

(37) [*p* Sales] are expected to drop when the discounts are eliminated.

In Example (37) we are looking for the $iarg_1$ (thing sold) of *sale*. The path shown in Example (35) indicates quite clearly that the candidate *cars* from Example (36), 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' contains multiple coreferential 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 59 is similar to Feature 9 (the frame element path) except that it captures the distance between predicate–argument positions within the VerbNet hierarchy. 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:

(38) 13.5.1.1 ↑ 13.5.1 ↑ 13.5 ↑ 13 ↓ 13.2

The path in Example (38) 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):

(39) [*c* Monsanto Co.] is expected to continue reporting higher [*p* earnings].

(40) The St. Louis-based company is expected to report that [*p* losses] are narrowing.

In Example (40) 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 (39), 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 (38), 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, recall that c' might contain multiple coreferential elements. In such a situation, the minimum path length is selected as the value for this feature.

4.2.3 Group 3: Filler-independent Features. Many of the features used by the model do not depend on elements of c'. These features are usually specific to a particular predicate. Consider the following example:

(41) 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 41 is *price:index*. Contrast this with the following:

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

The value of Feature 25 for Example (42) 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 B.1 depend only on the predicate and have values that take the form *predicate:feature_value*.

4.2.4 Group 4: Features Derived from Corpus Statistics. **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. We extended this PMI score to semantic arguments instead of syntactic dependencies. Thus, the value for this feature is computed as follows:

$$pmi(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) = log \frac{P_{coref\_pmi}(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle)}{P_{coref\_pmi}(\langle p_1, arg_i\rangle, *) P_{coref\_pmi}(\langle p_2, arg_j\rangle, *)} \quad (2)$$

We computed Equation (2) by applying verbal SRL (Punyakanok, Roth, and Yih 2008), nominal SRL (Gerber, Chai, and Meyers 2009), and coreference identification (OpenNLP) to the Gigaword corpus (Graff 2003); because these tools are not fast enough to process all 1,000,000 documents in the corpus, however, we selected subsets of the corpus for each p_1/p_2 combination observed in our implicit argument data. We first indexed the Gigaword corpus using the Lucene search engine.[9] We then queried this index using the simple boolean query "p_1 AND p_2," which retrieved documents relevant to the predicates considered in Equation (2). Assuming the resulting data have N coreferential pairs of arguments, the numerator in Equation (2) is defined as follows:

$$P_{coref\_pmi}(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle) = \frac{\#coref(\langle p_1, arg_i\rangle, \langle p_2, arg_j\rangle)}{N} \quad (3)$$

In Equation (3), *#coref* returns the number of times the given argument positions are found to be coreferential. In order to penalize low-frequency observations with

9 http://lucene.apache.org.

artificially high scores, we used the simple discounting method described by Pantel and Ravichandran (2004) resulting in the following modification of Equation (3):

$$x = \#coref(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle)$$
$$P_{coref\_pmi}(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle) = \frac{x}{N} * \frac{x}{x+1} \quad (4)$$

Thus, if two argument positions are rarely observed as coreferent, the value $\frac{x}{x+1}$ will be small, reducing the PMI score. The denominator in Equation (2) is computed with a similar discount factor:

$$x_1 = \#coref(\langle p_1, arg_i \rangle, *)$$
$$x_2 = \#coref(\langle p_2, arg_j \rangle, *)$$
$$P_{coref\_pmi}(\langle p_1, arg_i \rangle, *) P_{coref\_pmi}(\langle p_2, arg_j \rangle, *) = \frac{x_1 x_2}{(N^2)\frac{min(x_1,x_2)}{min(x_1,x_2)+1}} \quad (5)$$

Thus, if either of the argument positions is rarely observed as coreferent with other argument positions, the value $\frac{min(x_1,x_2)}{min(x_1,x_2)+1}$ will be small, making the denominator of Equation (2) large, reducing the PMI score. In general, the discount factors reduce the PMI score for argument positions that are not frequent in the corpus.

We refer to Equation (2) as a **targeted** PMI score because it relies on data that have been chosen specifically for the calculation at hand. Table 3 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 3 is useful in situations such as the following (contrived):

(43) Mary won [*c* the tennis match].

(44) [arg_0 John's] [*p* loss] was not surprising.

In Example (44) we are looking for the $iarg_1$ of *loss*. The information in Table 3 strongly suggests that the marked candidate *c*, being the arg_1 of *win*, would be a suitable filler

Table 3
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) for the discount factor.

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

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 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, we used the information–theoretic model proposed by Resnik (1996), which is defined as follows:

$$Pref(p, arg_n, s \in \text{WordNet}) = \frac{Pr(s|p, arg_n) log \frac{Pr(s|p, arg_n)}{Pr(s)}}{Z} \quad (6)$$

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

In Equation (6), *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 we 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 (1996). Note that a synset s that is not observed in the training data will receive a score of zero because $Pr(s|p, arg_n)$ will be zero.

Equation (6) computes the preference of a predicate–argument position for a synset; a single word can map to multiple synsets if its sense is ambiguous, however. 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:

$$Pref(p, arg_n, w) = \frac{\sum_{s_i} Pref(p, arg_n, s_i)}{n} \quad (7)$$

That is, the preference for a word is computed as the average preference across all possible synsets. The final value for Feature 27 is computed using the word-based preference score defined in Equation (7). Given a candidate implicit argument c' comprising the primary filler c and its coreferent mentions, the following value is obtained:

$$Pref(p, iarg_n, c') = \min_{f \in c'} Pref(p, arg_n, f) \quad (8)$$

In Equation (8), each f is the syntactic head of a constituent from c'. The value of Equation (8) 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, Frank, and Pinkal (2005) that implicit arguments might be identified using observed coreference patterns in a large corpus of text. Our 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, we identified coreferent arguments in this corpus using OpenNLP. Using this information, we calculated the probability of coreference between any two argument positions. As with Feature 13, we used

discounting to penalize low–frequency observations, producing an estimate of coreference probability as follows:

$$Coref_{joint} = \#coref(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle)$$
$$Coref_{marginal} = \#coref(\langle p_1, arg_i \rangle, *)$$
$$P_{coref}(\langle p_1, arg_i \rangle, \langle p_2, arg_j \rangle) = \frac{Coref_{joint}}{Coref_{marginal}} * \frac{Coref_{joint}}{Coref_{joint}+1} * \frac{Coref_{marginal}}{Coref_{marginal}+1} \quad (9)$$

In Equation (9), P_{coref} should be read as "the probability that $\langle p_1, arg_i \rangle$ is coreferential with $\langle p_2, arg_j \rangle$ given $\langle p_1, arg_i \rangle$ is coreferential with something." For example, we observed that the arg_1 for predicate *reassess* (the entity reassessed) is coreferential with six other constituents in the corpus. Table 4 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:

(45) Senators must rethink [*c* their strategy for the upcoming election].

(46) The [*p* reassessment] must begin soon.

In Example (46) we are looking for the $iarg_1$ of *reassess*. Table 4 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.

4.2.5 Group 5: Features Derived from the Discourse Structure of the Text. **Feature 67** identifies the discourse relation (if any) that holds between the candidate constituent *c* and the filled predicate *p*. Consider the following example:

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

(48) 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 second sentence. The coherence provided by this relation encourages

Table 4
Coreference probabilities between *reassess*.arg_1 and other argument positions. See Equation (9) for details on the discount factor.

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

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 our investigation, we used gold-standard discourse relations provided by the Penn Discourse TreeBank (Prasad et al. 2008).

4.2.6 Group 6: Other Features. A few other features that were prominent according to our feature selection process are not contained in the groups described thus for. **Feature 2** encodes the sentence distance from c (the primary filler) to the predicate for which we are filling the implicit argument position. The prominent position of this feature agrees with our previous observation that most implicit arguments can be resolved within a few sentences of the predicate (see Figure 1 on p. 770). **Feature 3** is another simple yet highly ranked feature. This feature concatenates the head of an element of c' with p and $iarg_n$. For example, in sentences (45) and (46), this feature would have a value of $strategy - reassess - arg_1$, asserting that strategies are reassessed. **Feature 5** generalizes this feature by replacing the head word with its WordNet synset.

4.2.7 Comparison with Features for Traditional SRL. The features described thus far are quite different from those used in previous work to identify arguments in the traditional nominal SRL setting (see the work of Gerber, Chai, and Meyers 2009). The most important feature used in traditional SRL—the syntactic parse tree path—is notably absent. This difference is due to the fact that syntactic information, although present, does not play a central role in the implicit argument model. The most important features are those that capture semantic properties of the implicit predicate–argument position and the candidate filler for that position.

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. Graphically:

	$iarg_0$	$iarg_1$
c_1	0.3	0.4
c_2	0.1	0.05
c_3	0.6	0.3

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.[10] Cells with probabilities less than 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.

5. Evaluation

5.1 Data

All evaluations in this study 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, we first randomized the annotated predicate instances. Following this, we 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. This evaluation set-up is an improvement versus the one we previously reported (Gerber and Chai 2010), in which fixed partitions were used for training, development, and testing.

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; otherwise, the candidate three-tuple was given a negative label. During testing, the system was presented with each predicate instance and was required to identify all implicit arguments for the predicate.

10 The threshold t is learned from the training data. The learning mechanism is explained in the following section.

Throughout the evaluation process we 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 4.1, the set of candidates for a missing argument position contains constituents labeled with a core (e.g., arg_0) PropBank or NomBank argument label. We used gold-standard PropBank and NomBank arguments; it is not the case that all annotated implicit arguments are given a core argument label by PropBank or NomBank, however. 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. Approximately 96% of implicit argument positions are filled by gold-standard PropBank or NomBank arguments.
- **Feature extraction.** Many of the features described in Section 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 our experiments, values for this feature contained no errors because gold-standard PropBank and NomBank labels were used. Note, however, that many features were derived from the output of an automatic SRL process that occasionally produced errors (e.g., Feature 13, which used PMI scores between automatically identified arguments). These errors were present in both the training and evaluation stages.

We also assumed the existence of gold-standard syntactic structure when possible. This was done in order to focus our investigation on the semantic nature of implicit arguments.

5.2 Scoring Metrics

We evaluated system performance using the methodology proposed by Ruppenhofer et al. (2010). 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. To give partial credit for inexact argument bounds, we scored predictions using the Dice coefficient, which is defined as follows:

$$Dice(Predicted, True) = \frac{2 * |Predicted \bigcap True|}{|Predicted| + |True|} \quad (10)$$

Predicted contains the tokens that the model believes fill the implicit 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*):

$$Score(Predicted) = \max_{True \in AF} Dice(Predicted, True) \tag{11}$$

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.[11] For example, consider the following true and predicted labelings:

(49) **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].

(50) **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 (49) there are three implicit argument positions to fill. The hypothetical system has made predictions for two of the positions. The prediction scores are:

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

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

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

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

We 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

11 Our evaluation methodology differs slightly from that of Ruppenhofer et al. (2010) in that we use the Dice metric to compute precision and recall, whereas Ruppenhofer et al. reported the Dice metric separately from exact-match precision and recall.

the predictions by systems A and B for each $iarg_n$, we 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, Lewis, and Hirschmant 1993).

5.3 LibLinear Model Configuration

Given a testing fold F_{test} and a training fold F_{train}, we performed floating forward feature subset selection using only the information contained in F_{train}. We used an algorithm similar to the one described by Pudil, Novovicova, and Kittler (1994). As part of the feature selection process, we 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). We 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.[12] For all folds, we used LibLinear's logistic regression solver and a candidate selection window of two sentences prior. As shown in Figure 1, 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}.

5.4 Baseline and Oracle Models

We compared the supervised model 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 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 4.2. We also evaluated an oracle model that made gold-standard predictions for candidates within the two-sentence prediction window.

12 See Appendix Table C.1 for a per-fold listing of features and model parameters.

5.5 Results

Table 5 presents the evaluation results for implicit argument identification. Overall, the discriminative model increased F_1 performance by 21.4 percentage 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 and 17.5 percentage 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 the Appendix for a per-fold breakdown of results and a listing of features and model parameters used for each fold.

We 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 5 because the assistant did not annotate the entire data set.

6. Discussion

6.1 Training Set Size

As described in Section 3.1, implicit argument annotation is an expensive process. Thus, it is important to understand whether additional annotation would benefit the ten predicates considered. In order to estimate the potential benefits, we measured the effect of training set size on system performance. We retrained the discriminative model for each evaluation fold using incrementally larger subsets of the complete training set for the fold. Figure 2 shows the results, which indicate minimal gains beyond 80% of the training set. Based on these results, we feel that future work should emphasize feature and model development over training data expansion, as gains appear to trail off significantly.

6.2 Feature Assessment

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

6.2.1 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 Appendix Table C.1). 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.

Table 5
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.

		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

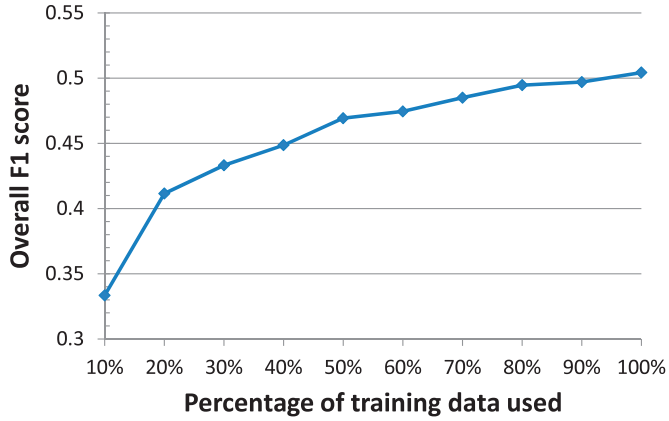

Figure 2
Effect of training set size on performance of discriminative model. The *x*-axis indicates the percentage of training data used, and the *y*-axis indicates the overall F_1 score that results.

6.2.2 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.

6.2.3 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; it remains a very challenging problem, however, 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 article used gold-standard discourse relations from the Penn Discourse TreeBank. As shown by Sagae (2009), these 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.

6.2.4 Relative Feature Importance. We extended earlier findings by assessing the relative importance of the features. We aggregated the feature rank information given in Appendix Table C.1. 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. We totaled these points across all folds, arriving at the values shown in the final column of Appendix Table B.1. The scores confirm the earlier findings. 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).

6.3 Error Analysis

Table 6 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.

Table 6
Implicit argument 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.

#	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 a candidate	21.1
4	A true filler scored highest but below the threshold	15.9
5	A true filler existed but not within the window	10.0

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; this came at the expense of increased type 3 errors, however. 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 5) 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, we 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. It is unclear whether either of these actions would increase overall performance, however.

6.4 The *Investment* and *Fund* Predicates

In Section 4.2, we discussed the *price* predicate, which frequently occurs in the "[*p* price] index" collocation. We 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 5). 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 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. It might be helpful to distinguish eventive nominals from non-eventive ones, given the observation that some non-eventive nominals rarely take arguments. Additional investigation is needed to address this type of error.

6.5 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:

(51) Mr. Rogers recommends that [*p* investors] sell [$iarg_2$ takeover–related stock].

Neither NomBank nor the baseline heuristic associate the marked predicate in Example (51) with any arguments; the feature-based model was able to correctly identify the marked $iarg_2$ as the entity being invested in, however. 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.

We conclude our discussion with an example of a complex extra-sentential implicit argument. Consider the following adjacent sentences:

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

(53) [arg_0 Olivetti] reportedly began [*p* shipping] these tools in 1984.

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

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

In Example (55), we are looking for the $iarg_0$ of *sale*. As shown, the discriminative model was able to correctly identify *Olivetti* from Example (54) as the implied filler of this argument position. The inference involved two key steps. First, the model identified coreferent mentions of *Olivetti* in Examples (52) and (53). 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.

6.6 Comparison with Previous Results

In a previous study, we reported initial results for the task of implicit argument identification (Gerber and Chai 2010). This article presents two major advancements versus our prior work. First, this article presents a more rigorous evaluation set-up, which was not used in our previous study. Our previous 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 article alleviated both problems. The F_1 difference between training and testing was approximately 10 percentage 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; the methodology in the current article is preferable for the reasons mentioned, however.

Second, this article presents a wider range of features compared with the features described in our previous study. In particular, we experimented with corpus statistics derived from sub-corpora that were specifically tailored to the predicate instance under consideration. See, for example, Feature 13 in Appendix B, which computed PMI scores between arguments found in a custom sub-corpus of text. This feature was ranked highly by a few of the evaluation folds (see Appendix B for feature rankings).

7. Conclusions

Previous work provided a partial solution to the problem of nominals with implicit arguments (Gerber, Chai, and Meyers 2009). The model described in that work is able to accurately identify nominals that take local arguments, thus filtering out predicates whose arguments are entirely implicit. This increases standard nominal SRL performance by reducing the number of false positive argument predictions; all implicit arguments remain unidentified, however, leaving a large portion of the corresponding event structures unrecognized.

This article presents our investigation of implicit argument identification for nominal predicates. The study was based on a manually created corpus of implicit arguments, which is freely available for research purposes. Our results show that models can be trained by incorporating information from a variety of ontological and corpus-based sources. 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, we 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 article 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; a direct application of the model to all NomBank predicates will require a substantial annotation effort, however. 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 (and verb-based nouns). Additionally, the model would benefit from a deeper understanding of the relationships that obtain between predicates in close textual proximity. Often, predicates themselves head arguments to other predicates, and, as a result, borrow arguments from those predicates following certain patterns. The work of Blanco and Moldovan (2011) addresses this issue directly with the use of composition rules. These rules would be helpful for implicit argument identification. Lastly, it should be noted that the prediction model described in this article 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. We have performed a preliminary investigation of joint implicit argument structures (Gerber, Chai, and Bart 2011); as described in that work, however, many issues remain concerning joint implicit argument identification.

Appendix A: Role Sets for the Annotated Predicates

Listed here are the role sets for the ten predicates used in this article.

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

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

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 *price*:
Arg_0: seller
Arg_1: commodity
Arg_2: price
Arg_3: secondary commodity

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 *loss*:
Arg_0: entity losing something
Arg_1: thing lost
Arg_2: entity gaining thing lost
Arg_3: source of loss

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

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

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

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

Appendix B: Implicit Argument Features

Table B.1
Features for determining whether c fills $iarg_n$ of predicate p. For each mention f (denoting a filler) 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 4.2. Features marked with a dagger (†) require external text corpora that have been automatically processed by existing NLP components (e.g., SRL systems). The final column gives a heuristic ranking score for the features across all evaluation folds (see Section 6.2 for discussion).

#	Feature value description	Importance 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*	Same as 3 except generalizing f to its WordNet synset.	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, Fillmore, and Lowe 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 (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
28	p **&** p's synset in WordNet.	0.3
29†	Same as 27 except using the maximum.	0.3

Table B.1
(*continued*)

#	Feature value description	Importance score
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 ns 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
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

Table B.1
(*continued*)

#	Feature value description	Importance score
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 ns) 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

Appendix C: Per-fold Implicit Argument Identification Results

Table C.1
Per-fold implicit argument identification results. 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.

		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)

Acknowledgments

We would like to thank the anonymous reviewers for their many insightful comments and suggestions. This work was partially supported by NSF grants IIS-0347548 and IIS-0840538.

References

Baker, Collin, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet project. In *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, CA.

Bhagat, Rahul, Patrick Pantel, and Eduard Hovy. 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.

Blanco, Eduardo and Dan Moldovan. 2011. A model for composing semantic relations. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS 2011)*, pages 45–54, Oxford.

Burchardt, Aljoscha, Anette Frank, and Manfred Pinkal. 2005. Building text meaning representations from contextually related frames—a case study. In *Proceedings of the Sixth International Workshop on Computational Semantics*, Tilburg.

Carpenter, Patricia A., Akira Miyake, and Marcel Adam Just. 1995. Language comprehension: Sentence and discourse processing. *Annual Review of Psychology*, 46:91–120.

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

Chambers, Nathanael and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics*, pages 789–797, Columbus, OH.

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

Chen, Desai, Nathan Schneider, Dipanjan Das, and Noah A. Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 264–267, Uppsala.

Chen, Zheng and Heng Ji. 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.

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

Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Dang, Hoa Trang, Diane Kelly, and Jimmy J. Lin. 2007. Overview of the TREC 2007 question answering track. In *Proceedings of the Fifteenth TREC*. Available at `trec.nist.gov/pubs/trec15/t15_proceedings.html`.

Di Eugenio, Barbara and Michael Glass. 2004. The kappa statistic: a second look. *Computational Linguistics*, 30(1):95–101.

Efron, Bradley and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

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

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, Cambridge, MA.

Fillmore, C. J. and C. F. Baker. 2001. Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*, Pittsburgh, PA.

Gerber, Matthew and Joyce Chai. 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.

Gerber, Matthew, Joyce Chai, and Robert Bart. 2011. A joint model of implicit arguments for nominal predicates. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 63–71, Portland, OR.

Gerber, Matthew, Joyce Chai, and Adam Meyers. 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, CO.

Graesser, Arthur C. and Leslie F. Clark. 1985. *Structures and Procedures of Implicit Knowledge*. Ablex Publishing Corporation, New York.

Graff, David. 2003. English Gigaword. Linguistic Data Consortium, Philadelphia, PA.

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

Harris, Zellig. 1985. Distributional structure. In J. J. Katz, editor, *The Philosophy of Linguistics*. Oxford University Press, New York, pages 26–47.

Heim, Irene and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell, Oxford.

Iida, Ryu, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop in ACL-2007*, pages 132–139, Prague.

Imamura, Kenji, Kuniko Saito, and Tomoko Izumi. 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.

Johansson, Richard and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Kipper, Karin. 2005. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Krippendorff, Klaus. 1980. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Thousand Oaks, CA.

Lin, Dekang and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Liu, Chang and Hwee Ng. 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.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn TreeBank. *Computational Linguistics*, 19:313–330.

Meyers, Adam. 2007. Annotation guidelines for NomBank—noun argument structure for PropBank. Technical report, New York University.

Nielsen, Leif Arda. 2004. Verb phrase ellipsis detection using automatically parsed text. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 1093–1099, Geneva.

Nielsen, Leif Arda. 2005. *A corpus-based study of Verb Phrase Ellipsis Identification and Resolution*. Ph.D. thesis, King's College, London.

NIST, 2008. *The ACE 2008 Evaluation Plan*. National Institute of Standards and Technology, Gaithersburg, MD.

Palmer, Martha S., Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 10–19, Morristown, NJ.

Pantel, Patrick, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 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, NY.

Pantel, Patrick and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, MA.

Pizzato, Luiz Augusto and Diego Mollá. 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.

Prasad, Rashmi, Alan Lee, Nikhil Dinesh, Eleni Miltsakaki, Geraud Campion, Aravind Joshi, and Bonnie Webber. 2008. Penn discourse treebank version 2.0. Linguistic Data Consortium, University of Pennsylvania, Philadelphia.

Pudil, P., J. Novovicova, and J. Kittler. 1994. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125.

Punyakanok, Vasin, Peter Koomen, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005 Shared Task*, pages 181–184, Ann Arbor, MI.

Punyakanok, Vasin, Dan Roth, and Wen-Tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Resnik, Philip. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61:127–159.

Ritter, Alan, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Uppsala.

Ruppenhofer, Josef, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 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, CO.

Ruppenhofer, Josef, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 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.

Sagae, Kenji. 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.

Sanford, A. J. 1981. *Understanding Written Language*. John Wiley & Sons Ltd, Hoboken, NJ.

Sasano, Ryohei, Daisuke Kawahara, and Sadao Kurohashi. 2004. Automatic construction of nominal case frames and its application to indirect anaphora resolution. In *Proceedings of COLING 2004*, pages 1201–1207, Geneva.

Schank, Roger C. and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Lawrence Erlbaum, Hillsdale, NJ.

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

Szpektor, Idan, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling Web-based acquisition of entailment relations. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 41–48, Barcelona.

Tonelli, Sara and Rodolfo Delmonte. 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.

van Dijk, T. A. 1977. Semantic macro structures and knowledge frames in discourse comprehension. In M. A. Just and P. A. Carpenter, editors, *Cognitive Processes in Comprehension*. Lawrence Erlbaum, Hillsdale, NJ, pages 3–32.

van Dijk, Teun A. and Walter Kintsch. 1983. *Strategies of Discourse Comprehension*. Academic Press, Waltham, MA.

Versley, Yannick, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 9–12, Marrakech.

Whittemore, Greg, Melissa Macpherson, and Greg Carlson. 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.

Wilson, N. L. 1974. Facts, events, and their identity conditions. *Philosophical Studies*, 25:303–321.

Wu, Zhibiao and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, NM.

Yang, Xiaofeng, Jian Su, and Chew Lim Tan. 2008. A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356.

Zanzotto, Fabio Massimo, Marco Pennacchiotti, and Maria Teresa Pazienza. 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.