ROBUST SEMANTIC ROLE LABELING USING PARSING VARIATIONS AND SEMANTIC CLASSES

Szu-ting Yi

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2007

Martha Stone Palmer
Supervisor of Dissertation

Rajeev Alur
Graduate Group Chairperson

Acknowledgements

I am most grateful to my advisor, Martha Palmer. Throughout my graduate study years at Penn, she has not only helped me become an independent scientific researcher but provided me valuable advice, supporting me through various crossroads I have encountered during my life. Without her encouragement and guidance, I would never have finished this dissertation.

I also want to thank my colleagues and fellow graduate students, Olga Babko-Malaya, Jinying Chen, Susan Converse, Hoa Trang Dang, Yuan Ding, Ryan Gabbard, Alexandra Kinyon, Karin Kipper, Seth Kulick, Edward Loper, Ryan McDonald, Libin Shen and Nianwen Xue, for their witty and inspirational discussion both on research and on other events. They have provided many helpful suggestions on my research and made my years at Penn delightful.

I thank Mitch Marcus, Aravind Joshi, Maria-Isabel Romero, and Dan Roth for being on my committee. Their sharp criticism and research pointers helped me avoid spending time on unnecessary experiments.

Last but not least, I want to thank my family. My parents in Taiwan have patiently waited for me to finish my study. My older brother Wen-tau, who is also in computer science, has reviewed numerous conference paper drafts and given me useful suggestions. I am especially in debt to my husband, Jeremy LaCivita, who has supported me with his unconditional love and has proofread all my papers and dissertation drafts even though he is not necessary interested in my thesis topic. He is the reason that I was able to keep persistent and finish my writing.

ABSTRACT

ROBUST SEMANTIC ROLE LABELING USING PARSING VARIATIONS AND SEMANTIC CLASSES

Szu-ting Yi

Martha Stone Palmer

Correctly identifying semantic entities and successfully disambiguating the relations between them and their predicates is an important and necessary step for successful natural language processing applications, such as text summarization, question answering, and machine translation. Researchers have studied this problem, semantic role labeling (SRL), as a machine learning problem since 2000, after large-scale corpora annotated for arguments for a broad range of predicates became available. However, after using an optimal global inference algorithm to combine several SRL systems, the growth of SRL performance seems to have reached a plateau.

SRL systems typical rely on an upstream syntactic parser to gather argument candidates. We believe that this one-way relationship is the bottleneck of semantic role labeling, and we attempt to tackle it by training parsers more suitable for the SRL task. We incorporated semantic role annotation directly into the parse tree annotation, and trained different types of parsers on this data. We found that our maximum entropy style parser (Ratnaparkhi, 1999) derived more benefit from the additional features than our Collins style parser, based on Dan Bikel's implementation (Collins, 1999; Bikel, 2004). It also demonstrated better adaptability when ported to a different genre (the Brown corpus), outscoring the Collins style parser on Brown SRL by 10%.

A thorough error analysis indicated that a better route to creating a suitable syntactic parser for the task of semantic role labeling is to create training data which is more consistent and less contradictory. We then carefully examined different types of Treebank/PropBank mismatches and both Treebank and PropBank made changes in order to reach synchronization.The preliminary assessment on the merged data by

comparing SRL performance on the old 300k and new 300k data indicate that the noisy data problem might still exist because the synchronization is not yet complete.

In order to achieve system robustness, we create a new set of semantic roles by transforming verb-specific PropBank roles to less verb-dependent thematic roles based on the mapping between PropBank and VerbNet. Our hypothesis is that a set of less verb-dependent roles should be easier to learn and port better to different genres. We compared SRL system performance trained on different sets of semantic roles, and the results confirm the hypothesis. The new system ports better to novel text. On a subtask of comparing one overloaded PropBank role to its mapped thematic roles, the new system trained on the WSJ corpus gains a 6% performance improvement on the test set extracted from WSJ, and a 10% performance improvement on the new genres from the Brown corpus.

Syntactic parsing is the bottleneck of the task of semantic role labeling and robustness is the ultimate goal. In this thesis, we investigate ways to train a better syntactic parser and increase SRL system robustness. We demonstrate that parse trees augmented by semantic role markups can serve as suitable training data for training a parser for an SRL system. Furthermore, we show that by resolving the discrepancies between Penn Treebank and PropBank, it is possible to create a cleaner training corpus both for training the parsers and the SRL systems. For system robustness, we propose that it is easier to learn a new set of semantic roles transformed from the original argument roles based on the mapping between VerbNet and PropBank. The new roles are less verb-dependent than the original PropBank roles. As a result, the SRL system trained on the new roles achieves significantly better robustness than the original system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Semantic parsing, identifying and classifying the semantic entities in context and the relations between them, potentially has great impact on its downstream applications, such as text summarization, question answering, and machine translation. Therefore, semantic parsing could be an important intermediate step for natural language comprehension. Recently, researchers have started to tackle this problem using machine learning techniques, because corpora annotated for argument structure for a broader range of predicates have been made available. The PropBank project (Palmer et al., 2005b) and the FrameNet project (Baker et al., 1998) share the goal of documenting the syntactic realization of arguments of the predicates of the general English lexicon by annotating corpora with semantic roles. These resources have been used to develop an automatic system that identifies all the semantic roles for a wide variety of predicates in unrestricted text (Gildea and Jurafsky, 2002; Fleischman and Hovy, 2003; Carreras and Márquez, 2004; Carreras and Márquez, 2005).

In this thesis, we investigate the task of Semantic Role Labeling (SRL). We explore how syntactic alternations and different granularities of the predicates and head nouns would influence SRL performance, and study the impact of different syntactic parsing frameworks on the task results. Semantic role labeling is defined as: Given a verb in a sentence, the goal is to locate the constituents which are arguments

of the verb, and assign them appropriate semantic roles, such as Agent, Patient, and Theme. Given that the PropBank annotates the entire Wall Street Journal section of the Penn Treebank II (Marcus et al., 1994) with predicate argument structures of verbs and has a broader and more realistic coverage than FrameNet, we select the PropBank corpus as our training and test corpus.

Many researchers have studied Semantic Role Labeling (SRL) as a machine learning problem since 2000 (Gildea and Jurafsky, 2000; Gildea and Palmer, 2002; Gildea and Jurafsky, 2002; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Hacioglu et al., 2003; Moschitti, 2004; Pradhan et al., 2004; Punyakanok et al., 2004; Yi and Palmer, 2004; Pradhan et al., 2005b; Punyakanok et al., 2005; Toutanova et al., 2005). For two years, the CoNLL workshop made this problem the shared task (Carreras and Márquez, 2004; Carreras and Márquez, 2005). Various machine learning algorithms have been tried, and many lexical features have been proposed; however, most of the systems share a common pipeline framework. An SRL system consists of two basic components, Argument Identification and Argument Classification, and contains optional pre- and post- processing components. An SRL system first takes as input the candidate constituents which are generated by processing a given sentence with either a syntactic parser or with several shallow parsing components, such as a POS tagger, a chunker and a clause identifier. These constituents then flow through an Argument Identification component, which is a binary classifier that determines whether the constituent in question is an argument of the verb being processed. If a verb argument is detected, it will be handed to an Argument Classification component to receive its role assignment.

Since an SRL system takes input from its upstream syntactic parsers without questioning the quality of candidate arguments, the performance of an SRL system is highly restricted by the parser. Many researchers have noticed this problem and common strategies include using k-best parses to start with or combining different systems. If an optimal global inference algorithm is applied while combining different

systems, a 2% boost on the final SRL performance is experienced (Punyakanok et al., 2005). However, if we do not break the one-way relationship between parsers and SRL systems, current state-of-the-art SRL systems seem to have reached their performance upper bound.

We attempt to tackle this bottleneck by training parsers more suitable for the SRL task. Traditionally statistical parsers were trained on almost purely syntactic information, because at that time only syntactic annotations were available. The only vaguely semantic feature might be the head word, which proved to be extremely useful. Nowadays, we have annotations on syntactic bracketing and predicate argument structure; however, we are still using parsers trained on only syntactic bracketing information. If we can train a parser which learns from both syntactic and semantic information, this parser might be more suitable for the SRL task.

Another issue we pay close attention to is system robustness which is essential to any NLP application. When we select the underlying parsing framework for the SRL task, we evaluate our system both on the WSJ corpus and the PropBank-style annotated Brown corpus, which has articles of different genres from the WSJ.

We first create a full-grown SRL system based on (Xue and Palmer, 2004) and evaluate the SRL performance using three different state-of-the-art syntactic parsers: The Collins parser (Collins, 1999), Charniak's parser (Charniak, 2000) and Ratnaparkhi's maximum-entropy parser (Ratnaparkhi, 1999). With a simple post-processing algorithm based on majority votes, the system combination of two SRL systems using Charniak's parser and the maximum-entropy parser demonstrates state-of-the-art performance.

Given that the maximum-entropy parser shows the most robustness and has excellent flexibility, we select it as our base parsing model for further experiments. We created our new training corpus by merging the PropBank argument information and the Penn TreeBank constituent information. If a constituent is an argument, we attach a sub-label to its constituent label. Among the three newly trained parsers,

the parser that is trained with training data in which only the adjunct-like arguments are marked, performs the best. We refer to this parser as the AM-parser, and compared to the base maximum-entropy parser which is only trained on the Penn TreeBank, the AM-parser performs better in terms of SRL performance on the WSJ test data and on labeling core arguments on the Brown corpus without sacrificing Parseval scores.

After the experiments of training parsers on parse trees augmented by semantic role markups, we performed a thorough error analysis in order to investigate the strengths and weaknesses of different parsers, especially the AM-parser and the Collins parser. We found out that while tested on the WSJ corpus, these two parsers have comparable results, on the Brown corpus the AM-parser seems to perform significantly better on constituent boundary detection and constituent attachment. We therefore tried to combine two parsers by feeding the Collins parser partial parse trees generated by the AM-parser; however, we had worse SRL performance using the resulting parse trees. We realized that in order to obtain a solid upstream syntactic parser for the task of semantic role labeling, the key solution is to resolve the discrepancies between Penn Treebank and the PropBank. Even if revolutionary changes are made to the implementation of current state-of-the-art parsers, we still need a clean training set which is not inconsistent and contradictory.

We therefore carefully examined the discrepancies between the Penn Treebank and PropBank and classified mismatching cases in different categories and proposed approaches to reconcile them. In order to synchronize these two corpora, Treebank added a new empty category to signal raising verbs and control verbs, and drew a cleaner distinction between verbs which take small clauses and secondary predicates; PropBank re-defined its policy on tagging links; the links are categorized into three classes and are tagged at different stages: syntactic chains, direct semantic links (co-reference), and indirect semantic type links. We evaluated the quality of the current 300k of the synchronized data by comparing two SRL systems trained on

the original data and the new data. The preliminary assessment on the merged data by comparing SRL performance on the old 300k and new 300k data indicate that the noisy data problem might still exist because the synchronization is not yet complete.

For SRL system robustness, we ask the question which has stimulated debates over decades in the linguistics and the NLP community: What is the most appropriate set of semantic roles? Current SRL systems which are trained on the PropBank corpus output the following labels: Arg[0-5], ArgA, and ArgM. However, PropBank uses these argument labels in order to avoid theoretical controversy, and these argument labels are defined on a verb-by-verb basis; that is, these PropBank argument roles are very verb specific. As a result, except for Arg0 and Arg1 which in general correspond to Agent and Patient, other numbered arguments, Arg2-5, are fairly overloaded. This creates noise in the training data and therefore poses a barrier to a robust SRL system.

Recently new mappings of different lexical resources have been made available, and we take the mapping between PropBank and VerbNet and transform the PropBank semantic roles to VerbNet thematic roles. Our hypothesis is that the VerbNet thematic roles are less verb dependent and therefore are easier to learn to port to other genres. We trained and compared SRL systems trained on the original PropBank roles and transformed VerbNet thematic roles. We found out that although the new system suffered from a data sparseness problem because the number of roles increased, the SRL system performance of the new system is still significantly better than the original system. This verifies our hypothesis.

The task of semantic role labeling faces two major challenges: 1) seeking a suitable syntactic parser; 2) accomplishing system robustness. A syntactic parser is the upstream component of an SRL pipeline; therefore it naturally acts as a bottleneck for the task of semantic role labeling. While current SRL systems seem to have reached a performance ceiling on the WSJ corpus, they do not port well to other

genres. However, robustness is the ultimate goal of the task of semantic role labeling. An SRL system without robustness cannot serve as a useful semantic processing tool.

In this thesis, we investigate ways to train a better syntactic parser and increase SRL system robustness. We demonstrate that parse trees augmented by semantic role markups can serve as suitable training data for training a parser for an SRL system. Furthermore, we show that by resolving the discrepancies between Penn Treebank and PropBank, it is possible to create a cleaner training corpus both for training the parsers and the SRL systems. For system robustness, we propose that it is easier to learn a new set of semantic roles transformed from the original argument roles based on a mapping between VerbNet and PropBank. The new roles are less verb-dependent compared to the original PropBank roles. As a result, the SRL system trained on the new roles achieves significantly better robustness than the original system.

Chapter 2

Semantic Representation

Data representation is always an important issue in the Natural Language Processing (NLP) community. In order for an NLP application to interpret and reason upon natural language input, it requires an adequate data representation to model real-world knowledge. For instance, a question answering system needs to identify what, when, who, why, where, and how.

While a small database of templates might fulfill the need of a domain-specific NLP application, most other NLP applications, depending on their scale and their purpose, have data representations which generally model aspects from a subset of the following natural language properties: morphology, syntax, semantics, discourse and so on. Recent years, with the rapid development of machine learning, and the improvement in syntactic parsing, there is a growing demand for deeper and broader semantic data representation and annotations.

2.1 Deep Semantic Representations and Applications

Deep semantic representation is not an easy task. Semantic entities need to be identified, relations between entities and their predicates (usually referred to as semantic

roles) have to be drawn, and there is little consensus among linguists on defining semantic roles (Baker, 1985; Dowty, 1989; Fillmore, 1968; Gruber, 1967; Jackendoff, 1972; Jackendoff, 1987; Marantz, 1984; Nishigauchi, 1984; Rappaport and Levin, 1988; Riemsdijk and Williams, 1986; Rozwadowska, 1988; Talmy, 1985). Nonetheless, a popular assumption that is shared by many proposals involves a closed set of semantic roles; and among different proposed sets, these roles are probably the most cited ones: Agent, Patient, Theme, Goal, Source, Experiencer etc. Furthermore, linguists exploit semantic roles to serve as an interface between syntax and semantics (Belletti and Rizzi, 1988; Di Sciullo and Williams, 1987; Grimshaw, 1990; Levin and Rappaport, 1986; Marantz, 1984; Rappaport and Levin, 1988; Williams, 1981; Zubizarreta, 1987). Given below are descriptions of the semantic roles previously mentioned (The description of these semantic roles are adopted from (Allen, 1995; Sanfilippo et al., 1999; Schuler, 2005))

Agent: A participant which does or causes something to happen, possibly intentionally. Example: *[John] broke the window.*

Patient: A participant which has something happen to it or a participant which is affected by what happens to it. Example: *John killed [Mary's son].*

Theme: A participant which has a change of position or condition such as possession. Example: *John gave [the vase] to his mother.*

Goal: Participant to which motion proceeds. Example: *Mary went to [New York].*

Source: Participant from which motion proceeds. Example: *John left [London] yesterday.*

Experiencer: A participant who is aware of something or experiencing something. Example: *[John] admires Mary.*

In the following sections, we will depict several data representations and corresponding applications, and briefly discuss the advantages and disadvantages of each approach.

2.1.1 Conversational Systems

With the working domain refined and the goal of the application clearly specified, early conversational systems reached decent performance using simple template slots or semantic frames to represent semantic entities (Rudnicky et al., 1999; Stallard, 2000; Seneff and Polifroni, 2000; Ward and Pellom, 1999). For example, Rudnicky et al. (1999) uses Phoenix, which is a rule-based text extraction tool that retrieves text segments based on given a frame mechanism tailored to the task (Ward and Issar, 1994). Stallard (2000) maps words to some predefined slots, such as DEST_CITY, ORIG_CITY, DEPART_TIME and ARRIVE_TIME.

When the application is small, templates suffice but in general it is hard to port the system to another domain. This problem is also addressed within the development of TRIPS (Allen et al., 2001). TRIPS is a spoken dialogue system that has been implemented for various domains; it performs spontaneous interactions with users who stay focused on the task in question. In order to enhance its portability, the developers of TRIPS look into smoothing the transformation between domain-specific semantic representation and a domain-independent general lexicon and parser (Dzikovska et al., 2003). Every entry in the general lexicon is expressed in a LF (Logic Form) representation, which includes type, semantic feature vector, and argument structure along with selectional restrictions. The design of the representation is inspired by Framenet (Johnson and Fillmore, 2000), EuroWordnet (Vossen, 1997), and the work of Lascarides and Copestake (1998). Furthermore, they are currently using VerbNet (Schuler, 2005) to expand lexicon coverage (Swift, 2005).

2.1.2 ACE and MUC

Attempts are being made to make information extraction tasks more domain independent by defining template slots or roles that are generic for most of the domains and by annotating more training data. For example, the Message Understanding Conference (MUC) (Chinchor, 1998) annotates named entities and a few relations, and the Automated Context Extraction Project (ACE)[1] expands the entity annotation to include nominal entities and defines more relations.

MUC-7 (Chinchor, 1998) annotates named entities of proper names and quantities: Person, Organization, Location, Date, Time, Percentage, and Monetary Amount. There are only 3 annotated relations: employee_of, product_of, and location_of, all indicating a relationship with an organization.

The annotations provided in the training data of ACE-II consisted of a type for each entity, together with offset annotations describing the locations (or mentions) of entities in the text. Similarly, relation objects describing the relationships between entities were annotated with a type and subtype and mentions of these relationships in the text were described by reference to the respective entity mentions. Entity types consist of person (PER), organization (ORG), facility (FAC), location (LOC), and geo-political entity (GPE). The type of relations consist of AT, NEAR, PART, ROLE and SOC; as for relation subtypes, one example is that AT has subtypes which include Based-In, Located and Residence.

2.1.3 ACQUILEX

With machine translation systems as target applications, the aim of the ACQUILEX project (Copestake, 1992; Copestake et al., 1992; Sanfilippo and Poznanski, 1992; Sanfilippo et al., 1999) was to construct a multilingual lexicon by linking lexical entries in several monolingual lexicons. Each monolingual lexicon is built upon extractions from machine-readable dictionaries and textual corpora. These extractions

[1] http://www.ldc.upenn.edu/Projects/ACE/

are then converted into attribute-value pairs to express syntactic and semantic information. The representation formalism is called Typed Feature Structure (TFS). Within the ACQUILEX Lexical Knowledge Base, Feature Structures might inherit from other Feature Structures, therefore, lexical entries form a hierarchical structure.

A typical ACQUILEX-style monolingual lexical entry includes but is not limited to the following information:

- Semantic relations such as hyponymy, synonymy, and meronymy.
- Semantic attributes characterizing a word. For example, Made-Of, Color, Size, and Shape.
- Noun Qualia Structure loosely based on Pustejovsky's work (Pustejovsky, 1991).
- Case Marking. For example, verb causativity/inchoativity.
- Lexical information such as collocations, selectional preferences, subcategorization frames, and predicate argument structures.

The ACQUILEX project works on representing a wide-spectrum of deep linguistic information, but this poses a potential threat to linking mismatched translations. In addition to representing detailed information about lexical semantics, the ACQUILEX project includes many syntactic and semantic properties of lexical items to ensure compatibility in order to have effective treatment when translation mismatches occur; however, this also causes the linking between two monolingual lexicons to be exponentially intractable when the gap of a translation mismatch increases.

The use of a formal representation, TFS, in ACQUILEX is to facilitate future NLP applications; however, its lack of flexibility generates problems in incorporating some important properties, especially word meanings which are inherently vague (Sanfilippo et al., 1999).

In addition to the difficulties in constructing an ACQUILEX-style data representation, the ACQUILEX project only focuses on selected areas of interest. The lack of broad coverage impedes production of end-to-end machine translation systems.

2.1.4 LCS

Lexical Conceptual Structure (LCS) is a rich semantic representation which was designed not only within a linguistic perspective but also a cognitive one (Jackendoff, 1983; Jackendoff, 1990; Jackendoff, 1996). LCS has strong relations with syntax and its most important notions are thematic relations which are defined through primitive operators. The graphic structure of an LCS resembles a semantic net or a directed graph. A lexical item in an LCS constitutes a node, and every node in an LCS is depicted with the following attributes: conceptual category, primitive and field.

A conceptual category is selected from a small set of ontological categories, which include: *thing, event, state, place, path, property, purpose, manner, amount, time.* A root category may consist of sub-categories, for example, *human, animal*, and *object* are sub-categories of *thing.* There are two general classes of primitives: a smaller class of primitives which covers various concepts: BE (state), GO (event), STAY (duration of a state), CAUSE (causality), INCH (inchoative interpretations), EXT (spatial extension), REACT, EXCH (exchange), ORIENT (orientation), etc., and a larger class of primitives which describe prepositions: AT, IN, ON, TOWARD, FROM, TO, BEHIND, UNDER, VIA, etc. The main fields in LCS consist of localization (+loc), time (+temp), possession (+poss), etc. For example, GO+loc depicts a change of location.

Because LCS retains a level of abstraction and language-independence, people have built large-scale LCS-based lexicons and several projects have used it as a semantic representation (Palmer, 1990) or as an interlingua for machine translation and other tasks (Dorr, 1997; Dorr et al., 1997). Some applications also used LCS to learn the behavior of verb constructions or to cluster similar verbs (Fujita et al.,

2004; Resnik and Diab, 2000).

2.2 Broad Semantic Representations and Applications

Syntactic parsing has managed to scale up to broad coverage applications through the use of machine learning techniques applied to annotated corpora (Charniak, 2000; Collins, 1999). There are currently a lot of proposals and annotations attempting to do the same thing for semantic representations, including The Proposition Bank (PropBank) (Palmer et al., 2005a), the FrameNet project (Baker et al., 1998), the VerbNet lexicon (Schuler, 2005), and the Interlingua Annotation of Parallel Corpora (Dorr et al., 2004).

2.2.1 PropBank

The PropBank (Palmer et al., 2005a) annotates the entire Wall Street Journal section of the Penn Treebank II (Marcus et al., 1994) with predicate argument structures of verbs. Semantic roles associated with their predicate structure are defined on a verb-by-verb basis. For a given verb, the PropBank annotates all the example sentences. The PropBank annotates all the verbs except for copula verbs.

PropBank-style annotations for languages other than English include the Penn Chinese PropBank (Xue and Palmer, 2003) and the forthcoming Korean PropBank [2]. A Parallel PropBank for Chinese and English is also constructed (Palmer et al., 2005b); this is a valuable resource for research regarding interlingua building and machine translation.

[2] http://www.cis.upenn.edu/ xtag/koreantag/

The PropBank Semantic Role Labels

In PropBank, semantic roles are defined on a verb-by-verb basis. An individual verb's semantic arguments are simply numbered, beginning with 0. Polysemous verbs have several Framesets, corresponding to a relatively coarse notion of word sense, with a separate set of numbered roles (a roleset) defined for each Frameset. For instance, *leave* has both a DEPART Frameset ([ARG0 John] left [ARG1 the room]) and a GIVE Frameset, ([ARG0 I] left [ARG1 my pearls] [ARG2 to my daughter-in-law] [ARGM-LOC in my will].) While most Framesets have three or four numbered roles, as many as six can appear, in particular for certain verbs of motion.

Besides the numbered arguments, which loosely correspond to verb complements, verbs can take any of a set of 13 optional arguments (ARGMs): EXT, DIR, LOC, TMP, REC, PRD, NEG, MOD, ADV, MNR, CAU, PNC, and DIS. Some of the ARGMs are adjunct-like arguments, such as EXT (extent), DIR (direction), LOC (location), TMP (temporal), MNR (manner), PRD (predication), DIS (discourse connectives), and PRP (purpose). Some of the ARMs mark argument features, for example, NEG (negation), MOD (modal), and ADV (adverbial).

Another major argument is ARGA. ARGA is the agent of an induced action. For example, [ARGA John] jumped [ARG0 the horse] [AGRM over the fence]. ARG0 is roughly equivalent to Agent and ARG1 is usually similar to Theme or Patient. However, argument labels are not necessarily consistent across different verb meanings of the same verb, or different verbs, as thematic roles are usually taken to be. There was an attempt, however, to maintain consistency across members of the same verb class (Kipper et al., 2000).

An argument is not always a continuous constituent. For example, in the following sentence, the ARG1 of the verb *say* is split into two constituents: [ARG1 Absent other working capital], he said, [C-ARG1 the RTC would be forced to delay other...]. We use C-ARGX in the example sentence to represent split arguments.

2.2.2 FrameNet

The Framenet Project (Baker et al., 1998) annotates selected sentences from the British National Corpus (BNC). Its goal is to construct evidence for the study of syntactic and semantic generalizations. The Framenet Project starts with constructing a hierarchical ontology of semantic domains selected by interest. Each semantic domain has a frame file in which the name, the inheritance property and a general description of this frame and its member predicates are stated. Most importantly, a frame file includes frame elements which represent the conceptual structure of its target semantic domain. The concept of frame elements are similar to that of thematic roles, however, instead of attempting to establish generalizations at the very top level of a natural language, frame elements work within the constraints of their designated semantic domain. After the preparations of the frame files, example sentences are extracted from the BNC corpus and are annotated with frame elements. The Framenet Project annotates predicates including verbs, nouns, and adjectives.

Frames and Frame Elements

The top semantic domains covered are: HEALTH CARE, CHANCE, PERCEPTION, COMMUNICATION, TRANSACTION, TIME, SPACE, BODY, MOTION, LIFE STAGES, SOCIAL CONTEXT, EMOTION, and COGNITION. Under these domains, there are sub-domains. For example, TRANSPORTATION has direct child domains such as DRIVING and RIDING-1. A child domain inherits the frame elements from its parent domain and has additional frame elements or has additional meanings associated with the frame elements inherited.

The TRANSPORTATION frame has the frame elements: MOVER(S), MEANS, and PATH. The DRIVING frame inherits the TRANSPORTATION frame and it has the following frame elements: DRIVER (=MOVER), VEHICLE (=MEANS), RIDER(S) (=MOVER(S)), CARGO (=MOVER(S)). We can see that in the TRANSPORTATION frame, the MOVER(S) are not necessarily the ones who operate the

MEANS; however, in the DRIVING frame, there must be a principal vehicle operator who is also a MOVER but there might be some other riding-along MOVER(S). The frame element PATH is specified in the TRANSPORTATION frame although is not specified in the DRIVING frame. Due to the parent-child relationship of these two frames, the DRIVING frame also has PATH as one of its frame elements.

Example Annotations

Below we list a few examples of the predicate *drive* and we make efforts to exemplify all the frame elements suggested by the DRIVING frame:

1. Yesterday [DRIVER my husband] *drove* [RIDER his friend] [PATH back to his residence].

2. [DRIVER Mr. Smith] always *drives* [VEHICLE his Porsche] to visit his girlfriend.

3. On Saturday, [VEHICLE the Bridal Wagon] will *drive* [PATH down to center city], following Main street.

2.2.3 VerbNet

VerbNet verb classes are constructed based on the syntactic alternation behaviors of individual verbs (Schuler, 2005). VerbNet is a hierarchical verb lexicon with syntactic and semantic information for English verbs, referring to Levin verb classes (Levin, 1993) for systematic construction of lexical entries. This lexicon exploits the systematic link between syntax and semantics that motivates these classes, and thus provides a clear and regular association between syntactic and semantic properties of verbs and verb classes (Dang et al., 2000; Dang et al., 1998; Kipper et al., 2000). The entry for each verb in a class is associated with the most suitable WordNet sense(s) (Miller, 1990), with the same verb in a different class typically receiving a different WordNet assignment.

The components of a verb class entry consist of the following:

1. The name of the class
2. The position in the hierarchy
3. The thematic roles this verb class takes
4. The selectional preferences applied upon the arguments
5. The frame structures

2.2.4 Interlingua Annotation of Parallel Corpora

Dorr et al. (2004) and Farwell et al. (2004) propose and describe an on-going project to build an interlingua for multiple languages. The underlying representation of the interlingua would include both syntactic realization and deep semantic information. Moreover, in order for different languages to agree upon the interlingua, the representation would be normalized to reduce vagueness and redundancy in the specification of meaning. A small set of parallel articles in seven languages (including English) have been annotated as a demonstration. The ultimate goal will be to use the representation in applications such as machine translation, question answering, text summarization, etc.

The interlingua annotation consists of three levels of representation; later levels build on previous levels. The first level is a deep syntactic dependency representation, constructed by hand-correcting the output of a dependency parser. The second level is an intermediate semantic representation; in this level, concepts drawn from the Omega ontology (Hovy et al., 2003) are assigned to nouns, verbs, adjectives, and adverbs. Also in this level, thematic roles (AGENT, THEME, GOAL, etc) are assigned to events or states to replace syntactic relations, similar to the Framenet and the PropBank. The projected third and final level would be meaning representation. Its objective would be to normalize over conversives, non-literal language usage, and

extended paraphrases in order to reconcile or merge intermediate semantic representations which are meaning equivalent (Dorr et al., 2004).

2.2.5 Applications

Representations and annotations are used to train automatic taggers in order to annotate new data. Gildea and Jurafsky (2002) design a statistical semantic role tagger trained on the Framenet corpus; Fleischman and Hovy (2003) apply a maximum entropy approach to FrameNet Tagging. The shared task of both CoNLL 2004 and CoNLL 2005 (Carreras and Márquez, 2004; Carreras and Márquez, 2005) is to train and optimize an automatic semantic role labeler based on the PropBank corpus.

The Framenet and the PropBank annotated corpora, and VerbNet, the broad-coverage verb lexicon, have also been applied to various NLP tasks. Swift (2005) exploits the VerbNet to expand an LCS-based lexicon utilized by a conversational system. Surdeanu et al. (2003) use predicate argument structure information derived from the PropBank to assist information extraction. Dang and Palmer (2005) investigate how semantic role information influences word sense disambiguation.

2.3 Semantic Representation for Machine Learning

Both deep semantic representations and broad semantic representations have advantages and disadvantages; however, when selecting an appropriate corpus from the perspective of NLP applications, at present a broad semantic representation is usually preferred. To serve well for machine learning, a corpus needs to be informative and representative; moreover, the amount of the data and the consistency of the data are equally important. The following discussion provides some pointers for selecting a semantic representation for machine learning:

Whether it contains enough information? As long as we can utilize it, the more information, the better. However, there are always compromises between the amount of the information a semantic representation intends to cover, the difficulty of the annotation process, and the amount and the quality of the resulting corpus. Although a deep semantic representation, such as ACQUILEX and LCS, contains much information, the complexity of the data structures make large-scale annotation unattainable. As a result, there may not be sufficient annotation for a machine learning system.

Whether it is easy to formalize? Machine learning systems that utilize an annotated corpus can be as straight-forward as an automatic tagger to further annotate more data, or they can be an intermediate natural language processing tool that supplies extracted information to its down-stream components. Regardless of the type of the machine learning system implemented, there is always a data preprocessing phase. A good semantic representation smooths out the transformation to a machine-readable data structure, minimizing the loss of information during the process.

Whether it represents real world knowledge? To implement an unbiased machine learning system, we need annotations that parallel real world data. In other words, there should be enough training data for a machine learning algorithm to converge; furthermore, the distribution of different example types in the corpus should be similar to the natural occurrence of those example types in real world data.

To conclude, to experiment with different machine learning techniques, we need to have a significant amount of labeled data to start with. In recent years, corpora of annotations on predicate argument structures have been developed and are matured enough for general purpose natural language processing systems. The most widely used predicate argument structure corpora are the Framenet project and the

PropBank project. The following section compares these two corpora in more detail.

2.3.1 Comparison between the Framenet and the PropBank

Both Framenet and PropBank locate the predicate and its arguments in a sentence and assign the semantic relations to each predicate-argument pair. However, the approaches and the methodologies these two corpora use have many different aspects:

Corpora:

The PropBank is another semantic layer on top of the Penn Treebank corpus, while the Framenet Project uses the British National Corpus (BNC). The BNC is a more balanced corpus than the Penn Treebank, since the former consists of articles from different genres and the latter consists of many articles on financial or economical issues from the WSJ. However, the FrameNet project only selects a few example sentences to annotate for every predicate, while the PropBank annotates the entire Penn TreeBank, therefore, the PropBank corpus reflects the real-world data more closely. Another advantage of using the Penn Treebank is that it is a syntactically bracketed corpus, and this facilitates the annotation process and minimizes the errors in defining the boundaries of semantic fragments. The BNC does not possess syntactic information, therefore the Framenet Project has to use automatic syntax-analyzing tools, such as a POS tagger.

Predicates:

The PropBank annotates all the verbs except for the copula verbs from the Penn TreeBank, and the Framenet Project annotates selected verbs, nouns and adjectives, which are members of the semantic domains in which they are interested. While the Framenet Project covers predicates with more syntactic categories, the PropBank covers a broader range of verbs and is a much bigger corpus than the Framenet Project.

Semantic roles:

The Framenet Project adopts a top-down approach, which defines semantic roles on the level of semantic classes, while the PropBank takes a bottom-up approach, which defines them on a verb-by-verb basis. The PropBank uses numbered arguments instead of named arguments in order to avoid the confusion the names might suggest; the FrameNet tries to give every semantic role which they refer to as frame elements a self-explanatory name.

Both the FrameNet Project and the PropBank have their strengths and shortcomings. Although the corpus the PropBank annotates is not a balanced corpus, we use the PropBank for the experiments presented in later chapters. This is because the PropBank corpus has a larger size and a broader coverage and the fact that the PropBank annotates every verb gives us a realistic distribution of natural language phenomena.

Merging these two corpora has been proposed. Regardless of the discrepancies between the two corpora, the similarities shared make this a practical suggestion. Hopefully in the near future, a corpus which covers predicates with several syntactic categories and annotates more data will be available to the computational linguistics community.

Chapter 3

Semantic Role Labeling

Researchers of the natural language processing community have been able to create highly accurate syntactic analysis tools, such as POS taggers or NP chunkers. In addition, the current state-of-the-art syntactic parsers have also reached almost 90% on labeled precision and recall. A natural next step is to construct applications which predict semantic entities and relations between them.

Correctly identifying semantic entities and successfully disambiguating the relations between them and their predicates is an important and necessary step for successful natural language processing applications, such as text summarization, question answering, and machine translation. As large-scale corpora annotated for arguments for a broad range of predicates are now available, such as the Penn PropBank project (Palmer et al., 2005a) and the FrameNet project (Baker et al., 1998), we are able to create statistical semantic relation identifiers based on state-of-the-art machine learning techniques which not only avoid the human effort of compiling hand-crafted rules for symbolic systems, but should also improve system performance and robustness.

In this chapter, we detail the task of Semantic Role Labeling (SRL): given a verb in a sentence, the goal is to locate the constituents which are arguments of the verb, and assign them appropriate semantic roles, such as, Agent, Patient, and

Theme. Many researchers have studied this problem as a machine learning problem since 2000 (Gildea and Jurafsky, 2000; Gildea and Palmer, 2002; Gildea and Jurafsky, 2002; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Hacioglu et al., 2003; Moschitti, 2004; Pradhan et al., 2004; Punyakanok et al., 2004; Yi and Palmer, 2004; Pradhan et al., 2005b; Punyakanok et al., 2005; Toutanova et al., 2005). For two years, the CoNLL workshop has made this problem the shared task (Carreras and Márquez, 2004; Carreras and Márquez, 2005). Various machine learning algorithms have been tried, and many lexical features have been proposed.

We will start with the formalization of SRL as a machine learning problem in Sec 3.1, followed by an introduction of the general framework of an SRL system in Sec 3.2. Sec 3.3 will discuss the trade-offs of extracting features from the output of shallow parsing applications or full syntactic parsers. Sec 3.4 details the features used by SRL systems and Sec 3.5 discusses different machine learning algorithms and grammars adopted by different systems. Sec 3.6 presents how the SRL community exploits global inference and system combination to promote SRL performance. Finally Sec 3.7 discusses challenges and further directions for the SRL task.

3.1 Semantic Role Labeling as a Machine Learning Problem

In this chapter, all the systems discussed use the PropBank as their training and testing data. The PropBank annotates the entire Wall Street Journal section of the Penn Treebank II (Marcus et al., 1994) with predicate argument structures of verbs.

Semantic role labels in the PropBank include numbered arguments (ARG0-5) and ARGA which are called core arguments as opposed to the adjunct-like arguments ARGMs. There are 13 sub-labels of the ARGMs, such as ARGM-ADV (general purpose), ARGM-CAU (cause), and ARGM-DIR (direction). For a detailed description of the PropBank, please refer to Sec 2.2.1.

The standard practice has Sec 23 of the Penn Treebank and the PropBank reserved for testing purposes, and Secs 02-21 for training and Sec 00 or Sec 22 or Sec 24 for development purposes.

The evaluation of SRL systems is typically expressed by precision, recall and the F1-measure. Precision is the number of correct arguments predicted by a system divided by the total number of arguments proposed. Recall is the number of correct arguments divided by the number of the total number of arguments in the Gold Standard Data. F1 computes the harmonic mean of precision and recall.

We provide the inter annotator agreement (ITA) rates of the PropBank corpus here to serve as the baselines of SRL in order to understand the difficulty of this task as a machine learning problem: the ITA rate for all numbered arguments (ARG0-5) is 84%, and the ITA rate for all arguments (ARG0-5, ARGA and ARGM) is 82%.

3.2 General Framework of an SRL System

Although different SRL systems have applied different machine learning approaches and have used different lexical features, in general they share a pipeline framework: Pre-processing, Argument Identification, Argument Classification and Post-processing.

3.2.1 Pre-processing

Pre-processing includes analyzing the data with syntactic tools (a POS tagger, a chunker or a full parser) and discarding highly improbable argument candidates. For systems which use shallow parsing tools only, they usually incorporate this phase with an Argument Identification phase. We will detail this when we describe the Argument Identification phase.

For systems which use a full syntactic parser, there is usually a filter or a pruning component which discards highly improbable candidates. This is because the

number of constituents predicted by a syntactic parser is large, and among all the constituents, only a very small portion can be an argument. This phenomenon that negative examples vastly outnumber positive examples makes it difficult to train a binary classifier to distinguish arguments from non-arguments.

Among all the different filter and pruning schemes, the algorithm presented in (Xue and Palmer, 2004) is the most successful one and it has been widely adopted by later SRL systems. This algorithm, which is inspired by linguistic theory, first spots the verb and then extracts all the sister nodes of the nodes along the verb spine (including the verb) of the parse tree. Yi and Palmer (2005) modified the algorithm to expand the coverage following this condition: if the node extracted by the original algorithm is an S, ADVP, PP and NP node, extract all its immediate children. This pruning component generally prunes out about 80% of the constituents given by a syntactic parser.

3.2.2 Argument Identification

If an SRL system makes use of a syntactic parser, an Argument Identification component is simply a binary classifier which determines whether a constituent given by the previous pre-processing component is an argument or not an argument.

If an SRL system does not have access to a syntactic parser, the objective of an Argument Identification component is to identify the boundaries of an argument. In general, there are two kinds of approaches: (1) the BIO approach, and (2) the Start-End approach.

The BIO approach marks the arguments in the training data with 3 sub-labels: B indicates a token which is the first word of an argument, I indicates a token which is not the first word but a participant of an argument, and O indicates a token which is not in any arguments. A classifier therefore is trained to give candidate arguments. For example, the lexical items in (1) will get the BIO marking: B I O B I I.

(1) [The boy] *broke* [the crystal vase].

The start-end approach trains two classifiers. One classifier marks the beginning of an argument and the other classifier tags the end of an argument. The output of the two classifiers therefore is combined to give argument candidates. Before the argument candidates are sent to an Argument Classification component, they are usually further sifted by heuristics which filter out highly improbable ones.

3.2.3 Argument Classification

An Argument Classification component is a multi-class classifier, and its objective is to assign semantic role labels to the argument candidates passed by an Argument Identification component. In addition to the target semantic roles, many SRL systems have their Argument Classification component keep null (non-argument) as one of the tags in order to catch those cases which are mistakenly treated as an argument in the Argument Identification phase but filtered out by the Argument Classification component.

3.2.4 Post-processing

Since every constituent is classified individually based on its local features and its relationship with the predicate, the correlations between the constituent and other arguments of the same predicate or the interactions between arguments of different verbs in the same sentence are usually overlooked. Therefore, there are usually competing arguments after the Argument Classification phase. For example, arguments are overlapping, or arguments of the same predicate are competing for the same numbered argument labels. Only under certain constructions are repeating numbered arguments allowed; for example, as in a sentence which uses coordination (this sentence is extracted from the WSJ corpus):

(2) Harley-Davidson's complaint *claims* that [ARG1 the group, led by investor

Malcolm I. Glazer, violated securities laws by failing to disclose plans to purchase 15% of the company's shares outstanding] and that [ARG1 when the required Hart-Scott-Rodino filing eventually was made, it didn't disclose the group's alleged earlier violation of the so-called prior-notice requirements of the law].

Many systems add on another post-processing component to resolve the above problems. A post-processing component can simply be a set of heuristics to remove arguments with respect to some constraints, and it can also be a statistical approach which takes global constraints into consideration to sort out competing arguments. There are several interesting approaches which apply global inference, and we will discuss them in more detail in Sec 3.6. In addition, a global inference approach is not necessary as a post-processing component.

3.3 Shallow Parsing versus Full Parsing

At first glance, it is more beneficial to supply an SRL system syntactic information with a full syntactic parser rather than several shallow parsing applications, because in order to correctly label semantic roles, the dependency of constituents is very important. However, because the nature of syntactic parsing is far more complicated than shallow parsing, currently syntactic parsers are not as accurate and robust as shallow parsing applications. In addition, a full syntactic parser is computationally more expensive than a shallow parser. All these factors have triggered researchers to conduct experiments which explore the trade-offs between using a full parser and several shallow parsing applications.

Gildea and Palmer (2002) prepared two kinds of syntactic representation for a sentence, one of which is a full parse tree, and the other one is a chunk-based representation. Their results suggest that although current syntactic parsers still have much space to improve, they provide better structural information and head

word information than chunk-based sentences.

Punyakanok et al. (Punyakanok et al., 2005) argue that the chunk-based representation used in (Gildea and Palmer, 2002) is in fact too shallow, because there is other shallow information to take into account, such as clauses. Furthermore, (Gildea and Palmer, 2002) neglect possible arguments which consist of more than two chunks, which results in a significant drop of recall, and causes lower general system performance. Therefore, in order to perform a fair comparison, (Punyakanok et al., 2005) represent a shallow representation with several shallow parsing applications, such as a POS tagger, a chunker, and clause identifier, and use the Start-End approach (Sec 3.2.2) to extract argument candidates. They observe the system's performance after each phase to better understand the causes and effects.

The results suggest that syntactic parsing might help in the Argument Identification phase; however, if the shallow parsing applications are accurate enough, the difference is marginal. In addition, if correct argument boundary information is given, there is no difference in the performance of the Argument Classification task. However, the final result still benefits from having a full syntactic tree as input. They conclude that the cumulative effects of a theoretically-inspired pruning scheme and syntactic parsing improve latter stages (Argument Identification and Classification). In other words, an SRL system starts with a better pool of candidate arguments to work with.

In later text, we focus on SRL systems which work with a full syntactic parser.

3.4 Lexical Features

Only six lexical features were used when Gildea and Jurafsky (2002) first established their SRL system. Chen and Rambow (2003) further augmented the feature set to include deep syntactic information represented by a set of binary features describing the transformation of a syntactic tree. The improvement of their system

indicated much room for improvement with respect to feature engineering. Other SRL systems (Surdeanu et al., 2003; Pradhan et al., 2005a) also attempted to extend the feature set by expanding the context window and including more content words other than the head word. Xue and Palmer (2004) reminded us that there should be different feature sets for the Argument Identification and the Argument Classification tasks respectively, since the two tasks are very different in nature. They thoroughly investigated what features were suitable for each task and proposed several combined features to replace or enhance stand-alone features. The feature set has become fairly stable, and most of the SRL systems use many of the features proposed by these systems.

The common approach of most SRL systems is to assign a semantic role label to an argument candidate given the predicate for that argument. The features fall into four categories: (1) features associated with the candidate constituent alone, (2) features associated with the predicate alone, (3) features associated with the relations between the candidate constituent and the predicate, and (4) combined features. Table 3.4 is a general listing of lexical features. In the table, GP is the abbreviation of Gildea and Jurafsky (2002), SP represents Surdeanu et al. (2003) and Pradhan et al. (2005b), and XP stands for Xue and Palmer (2004).

3.5 Variations in Learning Algorithms and Grammars

Many machine learning algorithms have been exploited for the SRL task. The most popular ones are Maximum-Entropy and Support Vector Machines. Systems that apply Support Vector Machines have chosen different kernel functions, including a standard polynomial kernel (Pradhan et al., 2004; Pradhan et al., 2005b; Pradhan et al., 2005a), a Gaussian kernel (Yi and Palmer, 2004) and a customized tree kernel (Moschitti, 2004). Other learning algorithms include SNoW (Punyakanok

	GJ	SP	XP
Features associated with the candidate constituent alone			
Phrase type of the constituent	X		
Head word of the constituent	X		
POS tag of the head word		X	
Phrase type of the parent node and the sibling nodes		X	
Head word of the parent node and the sibling nodes		X	
POS tag of the head word of the parent node and the sibling nodes		X	
The first content word of the constituent		X	
The last content word of the constituent		X	
The left or right tokens surrounding the constituent (token can be a word, chunk, or a phrase)		X	
If the parent node of the constituent is a PP node, then the head word of the PP node			X
The governing category	X		
Name entity of the constituent		X	
Features associated with the predicate alone			
Predicate (verb lemma)	X		
POS of the predicate			X
Voice	X		
Sub-categorization frame	X		
Features associated with the relations between the candidate constituent and the predicate			
Relative position of the constituent with respect to the target verb	X		
The distance between the constituent and the verb			X
The path from the constituent to the target verb	X		
Syntactic frame which describes the sequential pattern of the noun phrases and the verb in the sentence			X
Combined features			
The verb and the phrase type of the constituent			X
The verb and the head word of the constituent			X
If the parent node of the constituent is a PP node, then the verb and the head word of the PP node			X

Table 3.1: Lexical Features

et al., 2004; Punyakanok et al., 2005), Decision Trees (Chen and Rambow, 2003), AdaBoost (Márquez et al., 2005; Surdeanu and Turmo, 2005) and Memory-Based Learning (Sang et al., 2005).

According to the 2005 CoNLL shared-task on SRL (Carreras and Márquez, 2005), the results of the 19 participating systems did not show a preference for any particular learning algorithm. Since the choice of learning algorithm is not the only aspect which makes one system different from another, until we control all other aspects such as lexical features, and system framework, we cannot conclude that the SRL task has a preferred machine learning mechanism. However, in Chapter 4, we will demonstrate that with a careful model selection process, an SVM semantic role tagger can outperform other systems, some of which are trained on more features.

Most SRL systems assume a Head-based Probabilistic Context Free Grammar (HPCFG), because they extract lexical features from a syntactic parser (Charniak, 2000; Collins, 1999; Ratnaparkhi, 1999). Chen and Rambow (2003) use features called "deep syntactic information" which are represented as a feature vector that conveys syntactic transformation information according to a Tree-Adjoining grammar (TAG) extracted from the WSJ corpus (Chen and Vijay-Shanker, 2000). Gildea and Hockenmaier (2003) present an SRL system which extracts features from a statistical parser based on Combinatory Categorial Grammar (CCG), and their results suggest that CCG performs better than PCFG in explaining long-distance dependencies.

3.6 Global Inference and System Combination

Although some systems treat SRL as a sequential tagging problem (Márquez et al., 2005), most systems treat SRL as a classification problem, and assign a semantic role label to a candidate constituent. When a classification-based SRL system makes a decision, it focuses on the relationship between the candidate constituent and the

target predicate and neglects any interactions with other arguments of the same predicate.

Several approaches have been proposed to apply patches to this problem: Punyakanok et al. (2004) formulate SRL global inference as an integer linear programming (ILP) that takes as input the confidence corresponding to each type of argument supplied by the argument classifier. By maximizing the sum of the confidence level while respecting constraints posed by argument structures, the set of argument labels output is an optimal solution. Toutanova et al. (2005) implement a joint model that captures dependencies among arguments of a predicate. The joint model takes as input the argument assignments output by a local model and re-ranks the assignments according to global features and constraints.

Besides the relations between arguments of a predicate, relations between arguments of different predicates can also be a key factor of global inference. For example, in (3), *Ann* is an argument of the verbs, *persuade* and *go*. Currently, none of the SRL systems have incorporated this sort of dependency in the global inference process.

3. *Peter persuaded Anne to go.*

Another bottleneck faced by current SRL systems is the use of a not-quite perfect syntactic parser. As shown by the common architecture of SRL systems (Sec 3.2), the performance of an SRL system relies heavily on the quality of the upstream syntactic parser. Previous research has shown that if an SRL system is tested on gold standard parses, the accuracy of Argument Classification can reach about 95% and the system performance in terms of the F1 measure is close to 90 (Pradhan et al., 2005a; Xue and Palmer, 2004). However, when testing on automatic parses, since a syntactic parser typically produces less than 90% correct constituents, the system performance is brought down dramatically to below 80 in F1 measure. Meanwhile, the system is given a recall upper bound which is equivalent to the percentage of correct constituents the parser generates.

Many researchers have noticed this problem, and common strategies include: (1) Take k-best parses output by a syntactic parser and re-rank the combinations of parse tree and semantic role assignment (Toutanova et al., 2005; Sutton and McCallum, 2005); (2) Combine several SRL systems.

There are several ways to combine systems. Tsai et al. (2005) implement two argument classifiers, and have heuristics to evaluate every possible assignment based on the confidence level output by a maximum-entropy classifier and the ranking output by an SVM classifier. Most researchers have their SRL systems take as input different syntactic parsers and merge all the possible assignments according to a simple Voting algorithm (Yi and Palmer, 2005) or according to global inference (Punyakanok et al., 2004; Punyakanok et al., 2005).

The use of global constraints and system combination provides the overall SRL performance about a 2% improvement in F-1 measure (Punyakanok et al., 2005). A special discussion on system combination presented in the Conll 2005 shared task session revealed that an appropriate system combination can bring the overall SRL performance slightly beyond 80 in terms of F-1 measure after evaluating 4,096 possible ways to combine different systems.

3.7 Discussion

Current SRL development seems to have hit a bottleneck. Although system combination improves the overall SRL performance to a level not reached by any individual system, problems remain because an improper combination can bring down the overall performance: (1) how do we select the participating systems? (2) How many systems should we include? In addition, system combination does not make any structural changes to individual systems, so that the improvement gained by system combination is in fact limited by the individual systems themselves.

The question is what can we do to improve an individual SRL system? If we keep

the current one-directional pipelined framework, we need to start with a quality upstream component, in other words a better syntactic parser. However, as we have discussed in the previous chapters, syntax and semantics are highly correlated. Traditionally statistical parsers were trained on almost purely syntactic information, because back then only syntactic annotations were available. The only vaguely semantic feature might be the head word, which proved to be extremely useful. Nowadays, we have annotations on syntactic bracketing and predicate argument structure; however, we are still using parsers trained on only syntactic bracketing information. If we can train a parser which learns from both syntactic and semantic information, this parser might be more suitable for the SRL task.

Another issue which SRL systems will finally encounter is system robustness. This problem is actually a general problem for every natural language processing application. Robustness evaluates whether a natural language processing system performs well not only on similar domains to its training domain, but also on domains which are very different from the training domain. Moreover, a robust natural language application should be able to perform decently when it is ported to another language. The 2005 CoNLL shared task has addressed this issue by evaluating participating systems on a test set extracted from the Brown corpus which is very different from the WSJ corpus that was used for training. The results suggest that there is much work to be done in order to improve system robustness.

Chapter 4

Pilot Experiments

The discussion in the previous chapter suggests that the semantic role labeling (SRL) task has reached a performance upper bound because most SRL systems, instead of communicating with the upstream syntactic parsers, simply accept the parse trees given by them. In order to break the bottleneck, interactions between syntax processing and semantic tagging should be explored.

Another issue in SRL is how we represent the target predicate and the head noun of the constituent in question. A predicate, which is usually a verb, has different senses and therefore has different predicate argument structures. Therefore, a verb predicate should be represented by its sense instead of the surface text. Furthermore, verb and noun lemmas usually pose a data sparseness problem: previous research indicated a significant performance drop when testing the semantic role tagger on different genres other than WSJ stories (Carreras and Márquez, 2005; Pradhan et al., 2005a) possibly due to poor coverage of some lexical features, especially predicates and head nouns.

In this chapter, we present some preliminary experiments on both issues. A verb-argument model, which was inspired by Levin classes (Levin, 1993), clusters verbs and nouns based on their semantic relations and syntactic alternations. Training on the PropBank data, this model is used to tag Arg0 and Arg1, which are both the most

significant roles and the least verb-specific to demonstrate the interactions between syntax and semantics. Furthermore, we incorporate unlabeled data to explore the possibilities of using unlabeled data to augment insufficient labeled data.

We then implement a full-grown pipelined SRL system as a base system to investigate how altering the granularity of the predicates and head nouns would influence the SRL performance. The base system is implemented on Support Vector Machines with only the baseline features introduced by Gildea and Jurafsky (2002).

4.1 Stochastic Argument Labeling Using Labeled and Unlabeled Training Data

This section describes the transformation of a verb-argument model, based on observations of the syntactic alternation behavior of verbs (Gildea, 2002), into a semantic role predictor for tagging Arg0 and Arg1, which traditionally correspond to Agent and Patient. Not only are Agent and Patient the most frequent semantic roles, but also these two roles are less verb-specific than other roles. In addition, with the availability of the PropBank corpus, we can turn the hidden variable *role* from the original model into an observed variable. We ran experiments on the PropBank data to evaluate the smoothing power of the model. We bootstrapped the model on the British National Corpus to explore the strength of the model while being ported to text of different genres.

4.1.1 The Model

Many verbs exhibit alternations in their syntactic behavior, as shown by the following examples:

1. *The Federal Reserve increased rates by 1.4%.*

2. *Interest rates have increased sharply over the past year.*

The noun *rates* appears as the syntactic object of the verb *increase* in the first sentence, but as its subject in the second sentence, where the verb is used intransitively.

The phenomenon of verb argument alternations has been most comprehensively studied by Levin (1993), who categorizing over 3,000 verbs into classes according to which alternations they participate in. A central thesis of Levin's work is that a verb's syntactic alternations are related to its semantics, and that semantically related verbs will share the same alternations. For example, the alternation of examples 1 and 2 is shared by other verbs such as *decrease* and *diminish*.

We expect verbs that take similar sets of argument fillers to be semantically related, and to participate in the same alternations. This idea has been used by McCarthy (2000), Merlo and Stevenson (2001), and Schulte im Walde (2000) on verb classification tasks, with the comprehensive verb classes of Levin (1993) often serving as a Gold Standard.

Another strand of research has focused on automatic clustering algorithms for verbs and their arguments, with the goal of finding groups of semantically related words (Pereira et al., 1993; Rooth et al., 1999), without focusing specifically on alternation behavior. Gildea (2002) combined these two approaches to develop a verb-argument model which explicitly captures alternation behaviors in an attempt to accurately distinguish semantic classes.

Figure 4.1 illustrates the structure of the model. The model has five variables, which are cluster c , verb v , noun n , slot s and role r . The cluster variable represents the semantic class: in the following experiments we assume cluster c has 64 values; i.e. 64 possible classes. Slot s has 3 possible values: subject or object of an intransitive verb, or subject of a transitive verb. Role r has 2 possible values: Arg0 or Arg1. The dependency relations of the variables are shown in the Figure 4.1. The role r is dependent on both the cluster c to which our verb-noun pair belongs, and the syntactic slot s in which the noun is found. The probability of an observed

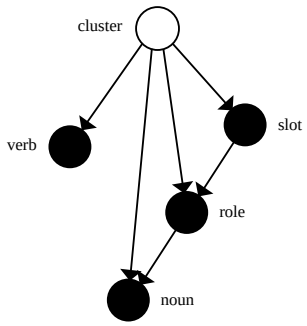

Figure 4.1: First Model: shading represents observed variables, arrows probabilistic dependencies.

sequence $P(v, s, n, r)$ is estimated as: $\sum_c P(c)P(v|c)P(s|c)P(r|c, s)P(n|r, c)$

The noun is independent of the verb given the cluster variable, and the noun is independent of the syntactic slot s given the cluster c and the semantic role r . The semantic role variable r takes two values, with $P(r|c, s)$ representing the mapping from syntactic slot to semantic role for a cluster of verbs.

The Expectation Maximization algorithm (Dempster et al., 1977) is used to train the model from the corpus, iterating over an Expectation step in which the expected value for the unobserved variable c is calculated for each observation in the training data, and a Maximization step in which the parameters of each of the five distributions $P(c)$, $P(v|c)$, $P(s|c)$, $P(r|c, s)$, and $P(n|r, c)$ are set to maximize the likelihood of the data given the expectation for c .

4.1.2 Data Preparation

For our experiments we extracted (v, s, n, r) sequences from the PropBank corpus as labeled data and (v, s, n) triples from a version of the British National Corpus parsed with the statistical parser of Collins (Collins, 1999) as unlabeled data. Subject and direct object relations were extracted by searching for NP nodes dominated by S and VP nodes respectively. The head words of the resulting subject and object nodes were found using the deterministic headword rules employed by the parsing model. The individual observations of our dataset are noun-verb pairs of three types: direct object, subject of a verb with an object, subject of a verb without an object. As a result, the subject and object relations of the same original sentence are considered independently by all of the models we examine.

Both verbs and nouns were lemmatized using the XTAG morphological dictionary developed by the XTAG Research Group (2001). The resulting unlabeled dataset has a vocabulary of 8,197 verbs and 50,158 nouns, and consists of 1,015,103 triples of verb, noun and syntactic relation. The resulting PropBank dataset has a vocabulary of 1,556 verbs and 7,985 nouns, and consists of 69,386 4-ary tuples of verb, noun, syntactic relation and semantic role. Of those, 90% were used as training material, 5% were used as a development set in order to determine the stop point of EM iterations, and the remaining 5% were used as test data.

4.1.3 First Set of Experiments

We compared the system's performance in a number of experiments in order to explore the smoothing power of the clustering model and the impact of combining labeled data and unlabeled data.

Baseline. This model simply applies the Maximum Likelihood Estimation method to acquire probabilities $P(r|v, s, n)$, $P(r|v, s)$, and $P(r|n, s)$ by counting the

frequencies of the training examples and uses the probabilities to tag test examples with the following back-off order: $P(r|v,s,n)$, $P(r|v,s)$, and $P(r|n,s)$. If both verb and noun of the test example are unseen in the training data, then we tag the test example with semantic role 'Arg0' if the syntactic slot indicates a subject and 'Arg1' otherwise.

Supervised Clustering. This model is depicted by Figure 4.1, using the EM algorithm to smooth the probabilities. Given a test example, $P(v,s,n)$ is a constant, thus we only have to compare $P(r,v,s,n)$ to tag semantic roles, since $P(r|v,s,n) = P(r,v,s,n)/P(v,s,n)$.

$P(r,v,s,n)$ is estimated as: $\sum_c P(c)P(v|c)P(s|c)P(r|c,s)P(n|r,c)$

If a verb or noun is unseen in the training data, we ignore $P(v|c)$ or $P(n|r,c)$ while computing the above equation.

Partially Supervised Clustering. This approach was inspired by the work of Nigam et al. (2000). The Supervised Clustering model first iterates on the labeled data, and obtains the smoothed probabilities. It then tags the unlabeled data probabilistically to create a set of labeled BNC data. The Supervised Clustering model merges the labeled PropBank data with the labeled BNC data as the new training set, and iterates on it until convergence. The new smoothed probabilities are used to tag test examples.

Discussion

The results are presented with respect to the accuracy of predicting the semantic roles of Arg0 and Arg1 of the held-out test data.

Table 4.1 shows the performance of the Baseline model, giving the number of instances for which each conditional probability was used. In experiments where we compared the performance of just $P(r|v,s)$ and $P(r|n,s)$, we found verb information

to be more relevant than noun information for the prediction of semantic roles. Notice that our baseline is extremely high at 96.20%.

Table 4.1: Baseline model performance: rows indicate levels of back-off. Test example count: the number of test examples which are labeled under different back-off levels

	Baseline Model	
Back-off order (top to bottom)	Test example count	Accuracy (%)
$P(r\|v,s,n)$	1,484	98.72%
$P(r\|v,s)$	1,953	94.36%
$P(r\|n,s)$	30	90%
$P(r\|s)$	3	100%
Total	3,470	96.20%

Table 4.2 and Table 4.3 describe the performance of the Supervised Clustering and Partially Supervised Clustering models. Adding the BNC data actually hurt our overall performance, although not significantly. The coverage of test examples labeled in terms of $P(r|v,s,n)$ more than doubled with both approaches, but the accuracy is also significantly worse than the Baseline model.

4.1.4 Revised Model

In hopes of improving our performance, we reexamined our model. The dependency relation between the noun n and the role r seems much less important to the overall task than the dependence between the cluster variable c, the slot s and the role r.

We revised the probabilistic model by taking off the dependency between noun n and role r , letting noun n and role r be conditionally independent given cluster c. $P(v,s,n,r)$ of the revised model is estimated as: $\sum_c P(c)P(v|c)P(n|c)P(s|c)P(r|c,s)$

The Expectation Maximization algorithm iterates over an Expectation step in which the expected value for the unobserved variable c is calculated for each observation in the training data, and a Maximization step in which the parameters of each of the five distributions $P(c)$, $P(v|c)$, $P(n|c)$, $P(s|c)$, and $P(r|c,s)$ are set to maximize the likelihood of the data given the expectation for c.

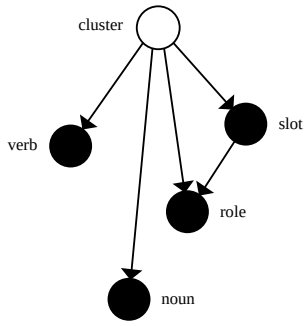

Figure 4.2: Revised Model: shading represents observed variables, arrows probabilistic dependencies.

Figure 4.2 is the graphical representation of the revised model and Table 4.2 and Table 4.3 report the accuracy numbers of the Supervised Clustering and Partially Supervised Clustering approaches on the revised clustering model.

The numbers are better. In fact our Supervised Clustering performance is only very slightly less (less than .3% difference) than the Baseline, although incorporating the BNC data still makes things marginally worse. However, there is no evidence at this point that EM clustering has any advantage over MLE.

4.1.5 Generalization to New Predicate

The advertised benefit of EM clustering is more robust performance over unseen data. According to Zipf's Law, in supervised training we will never gather a reliable distribution for most lexical items, therefore we need to use smoothing methods and

Table 4.2: Performance of Supervised Clustering Approach on the First Model and the Revised Model

	Supervised Clustering		
	Count	First Model	Revised Model
All info is seen	3,267	93.08%	96.02%
Verb unseen	1	100%	100%
Noun unseen	202	93.07%	94.06%
Both verb and noun unseen	0		
Total	3,470	93.08%	95.91%

Table 4.3: Performance of Partially Supervised Clustering Approach on the First Model and the Revised Model

	Partially Supervised Clustering (Bootstrapping on BNC)		
	Count	First Model	Revised Model
All info is seen	3,345	92.53%	95.61%
Verb unseen	0		
Noun unseen	125	92.08%	92.08%
Both verb and noun unseen	0		
Total	3,470	92.54%	95.50%

clustering models in order to learn unseen events by analyzing the behavior of similar events. Good clustering strategies should help us to extend our systems to different genres and to build domain independent language understanding applications.

However, as evidenced by our Baseline performance, over 40% of the test triples have already been seen in the training data. The PropBank corpus from which we extracted the dataset for the experiments is not a balanced corpus and does not contain a variety of genres. In fact, the Wall Street Journal articles have very similar topics. Since our 5% test data was drawn randomly from this repetitive source, we should not expect it to showcase the advantages of a clustering model with respect to predicting unseen events.

In order to investigate whether our clustering model is indeed capable of handling unseen events when analyzing the semantics of participant roles, we made a new division of the PropBank data into training and test sets with respect to verb information, ensuring that all of the verbs in the test data did not occur in the training data. We expected our model to learn a new verb's conditional probability based on the alternation behavior of verbs it is similar to.

Our new training dataset has 62,406 (verb, slot, noun) triples with 1,489 verb types; our new testing dataset has 3,510 triples with 67 verb types. The development set remained unchanged. The same experiments for the Baseline model and the Supervised Clustering were run using only the revised graphical model. In order to provide an appropriate comparison for the Partially Supervised Clustering approach, we designed a Bootstrapping Baseline method, which uses MLE to tag unlabeled BNC data probabilistically according to the conditional probabilities retrieved from PropBank training data. The Baseline model was then run on PropBank training data combined with probabilistically labeled BNC data to obtain new probability estimates and to tag the testing data.

4.1.6 Results and Discussion

Table 4.4 and Table 4.5 describe the performance of the Baseline and Supervised Clustering as well as their corresponding bootstrapping methods. The model accuracy is based on the percentage of the correct number of semantic role labels.

As can be seen from the table, the Partially Supervised Clustering approach obtained an estimation of conditional probability $P(r|v, s, n)$ for over 95% of the test examples, but the Bootstrapping Baseline has only about 40% to work with. The unreliability of the added BNC data worsens the overall MLE performance. However, the improvement in performance of the Partially Supervised Clustering model over the Baseline model is now statistically significant, validating both the ability of clustering to predict unseen events and the usefulness of incorporating unlabelled data given an appropriate probabilistic model.

Table 4.4: Baseline and Baseline Bootstrapping on new split of training and test data

	Number of labeled test examples / accuracy (%)			
	Baseline (PB)		Baseline Bootstrapping (PB+BNC)	
$P(r\|v, s, n)$			1,465	95.70%
$P(r\|v, s)$			2,045	87.29%
$P(r\|n, s)$	3,165	93.78%		
$P(r\|s)$	345	77.97%		
Total	3,510	92.22%	3,510	90.80%

4.2 Experiments on the Granularity of the Predicates and Head Nouns

In this paper, we develop a full-grown statistical semantic role labeler trained on the PropBank corpus using Support Vector Machines (SVM) (Vapnik, 1995) as a base system to perform experiments addressing system robustness. The SRL tagger only uses the baseline lexical features first introduced by Gildea and Jurafsky (2002):

Table 4.5: Supervised Clustering and Partially Supervised Clustering Approach of the revised model on new split of training and test data

	Number of labeled test examples / accuracy (%)			
	Supervised Clustering (Revised Model)		Partially Supervised Clustering (Bootstrapping on BNC)	
All info are seen	0		3,397	94.17%
Verb unseen	3,324	93.59%	0	
Noun unseen	0		113	86.73%
Both verb and noun unseen	186	83.87%	0	
Total	3,510	93.08%	3,510	93.93%

predicate, constituent type, head word, path, voice, position, and sub-categorization frame. By carefully tuning the parameters of the machine learning technique in question, which is SVM, our system achieves results that are better or comparable to the best published results using the same features.

Furthermore, previous research indicated a significant performance drop when testing the semantic role tagger on different genres other than WSJ stories (Pradhan et al., 2005a; Carreras and Márquez, 2005), possibly due to poor coverage of some lexical features, especially predicates and head nouns. An NLP system should be robust enough to handle test data from domains other than the training data domain in order to face real-world tasks. We use this system to investigate the influence of features based on verbs and head nouns, which, due to sparse data, usually lose coverage when moving to a different domain. We perform the following additional experiments altering the granularity of the predicates and head nouns in an attempt to address this problem:

Verb Sense Experiment: In order to evaluate the contribution of verb sense disambiguation information, we used the more specific "Frameset" information provided by the PropBank annotation as a feature instead of using the surface form of the verb.

Verb Class Experiment: We investigated the possibility of predicting semantic arguments for unseen verbs by consulting VerbNet classes (Kipper et al., 2000), which is a hand-crafted verb lexicon based on Levin classes (Levin, 1993).

Head Noun Class Experiment: Since nouns always cause the worst sparse data problem, we replaced the head nouns with their semantic classes, which are created automatically by an EM clustering algorithm based on syntactic alternations.

4.2.1 Base SRL System Overview and SVM Models

Our semantic role labeling is also a pipelined system, consisting of 3 components: pruning, argument identification, and argument classification. The Pruning component not only uses the heuristics presented by Xue and Palmer (2004) to extract candidate arguments, but also extracts all the children of a PP node and all the children of an S node if the S node appears to be a sister node of the predicate. For a detailed description of a general framework of an SRL system, please refer to Chapter 3.

We train our Argument Identification and Argument Classification classifiers based on support vector machines. Support vector machines are a supervised machine learning technique which maps a given set of binary labeled training data to a high-dimensional feature space and separate the two classes of data with a maximum margin hyperplane. SVM learning algorithm is attractive because of its outstanding generalization performance and its flexibility on handling the data sparseness problem (Burges, 1998).

We implemented our classifiers based on LIBSVM, which is a library for Support Vector Machines developed by Chang and Lin (2001). We chose the radial basis function (RBF) as the kernel function, and set the degree parameter to 3.

Before we trained the actual classifiers, we first performed a 5-fold cross-validation for model selection which determined the optimal combination of values for the

parameters: the gamma parameter γ for the kernel function and the cost parameter C. The selection of the combination of γ and C is determined by a grid search, which pairs up exponentially growing sequences of γ and C within a pre-defined range, for example, $\gamma = 2^{-15}, 2^{-13}, \cdots, 2^{3}$, $C = 2^{-5}, 2^{-3}, \cdots, 2^{15}$. We then trained our two SVM classifiers with the optimal parameters. The Argument Identification classifier has 512.0 as the cost parameter and 0.5 as the gamma parameter, and the Argument Classification classifier has 32.0 as the cost parameter and 0.0078125 as the gamma parameter.

SVM was originally designed for binary classification. In order to use SVM for a multi-class classification task, a common approach is to train multiple binary SVM classifiers and then assign the final classification based on a majority vote or some other heuristic. Multi-class classification using SVM is still an open research area. In this paper, we took a pair-wise approach rather than the one-versus-others approach to establish the binary classifiers and selected the final classification based on a majority vote.

4.2.2 Base System Results: Comparison and Discussion

Comparison

Table 4.6 summarizes the performance of our base system, SVM (RGB), as compared with some previous work. In the table, we compare 4 different systems: G&P (Gildea and Palmer, 2002), G&H (Gildea and Hockenmaier, 2003), Pradhan (Pradhan et al., 2004) and our base model. As shown in the table, our model outperforms the other systems on Argument Classification using the PropBank 2002 release.[1] Chen and Rambow's results suggest that using richer lexical information is beneficial. Our model uses a limited choice of features, but because of the statistical technique we chose, still has the best performance. As for the PropBank 2004 release, although

[1] (Chen and Rambow, 2003) reported error rates instead of accuracy of semantic role classification, and their results are not directly comparable.

the performance of the baseline system of Pradhan et al. (2004) which uses the same set of features as our system is not available, we can see from the table that our system has comparable performance to their best system, which compiles a rich set of lexical features. The additional rich lexical features were beneficial in boosting their 2002 release results from 81 to 87 (F1 measure). Given a similar boost for the 2004 data, our system would significantly outperform theirs based on the same feature set.

Table 4.6: A comparison of performance on the combined task of Argument Identification and Argument Classification and a comparison of accuracy (A) on only Argument Classification between the SVM model in question and previous works

Data	System	P(%)	R(%)	F1	A(%)
2002	G&P	71	64	67	77.0
2002	G&H	76	68	72	-
2002	Pradhan - baseline	83	79	81	87.9
2002	SVM(RGB)	-	-	-	93.1
2002	Pradhan - rich features	89	85	87	91.0
2004	SVM(RGB)	89	88	88	93.5
2004	Pradhan - rich features	90	89	89	93.0

Discussion

As the results suggest, SVM is a good choice for this kind of classification task. However, the behavior of an SVM model is pre-determined by the kernel function used with it. After the kernel function is decided, parameter selection also plays a critical part in the model's performance. Therefore a good strategy for model selection is critical for a high performing SVM model.

The gap in accuracy between our model and Pradhan et al. (2004) highlights the importance of the role played by kernel and parameter selection in Support Vector Machines. Based on current research in the SVM community, in general, we favor an RBF kernel over linear and polynomial kernels, because the RBF kernel maps training examples to a higher dimensional space which can handle nonlinear relations

between class labels (Chang and Lin, 2001). The RBF kernel also encounters less numerical problems than the polynomial kernel when the degree parameter gets higher. In addition, the right model selection can be more effective at generalizing over unseen events, as demonstrated below.

Another contributing factor might be the pair-wise approach we selected for building the multi-class SVM classifier. A pairwise approach constructs $\frac{n(n-1)}{2}$ binary classifiers for every possible pair of classes, where n is the number of classes. A one-versus-others approach constructs a binary classifier for every class, which treats the training instances of the class in question as positive examples and all other training examples as negative examples. Theoretically, a one-versus-others approach cannot always find a separable hyperplane for some classes. For example, suppose that the instances of a certain class lie in the boundary between the other two classes, then it cannot be distinguished by a hyperplane. Besides, a one-versus-others approach usually has to deal with the situation of negative examples wildly out-numbering positive examples (Krebel, 1999). Empirically, some experiments (Hsu and Lin, 2002; Krebel, 1999) reported that the pairwise approach outperforms the one-versus-others approach for problem types.

4.2.3 Adding Verb Sense Information

A given verb with different senses can have different syntactic and semantic behavior; therefore it is natural for such a verb to have multiple predicate argument structures. Consider the following example sentences extracted from the PropBank lexical guidelines for the verb *call*:

1. *Mary called John an idiot.*

2. *Ford called a Jaguar stockholders' meeting.*

In (1) the verb *call* means "to label something with some attribute," and as shown in the example, it has 3 core arguments, where *Mary* is ARG0, *John* is

ARG1, and *an idiot* is ARG2. In (2) the verb *call* means "to demand something," and under the circumstances, it has only two core arguments, where *Ford* is ARG0 and *a Jaguar stockholders' meeting* is ARG1. These two usages of call would have different Frameset tags.

Current approaches to semantic role labeling all use the surface text of the given verb as a feature without further disambiguating the verb senses. In our experiments, we replace the verb feature with a more specific "Frameset" feature, which loosely corresponds to verb sense information. The Argument Classification results on 8 labels improved from 93.45% to 93.74%, which is not significant.

4.2.4 Backing-off with VerbNet Classes for Unseen Verbs

In order to extend the predictive power of our semantic role tagger on the predicate arguments of unseen verbs, we tried a set of experiments with VerbNet verb classes. (For a detailed description of VerbNet, please read Sec 2.2.3.)

The VerbNet verb classes include 352 classes and cover 4,227 verbs. Among the 3,242 verbs that PropBank annotated, VerbNet verb classes only cover 1,837 verbs. For those verbs that have a class assignment in VerbNet, we use the verb class information to instantiate the predicate feature instead of the lemma. If the verb has multiple class assignments, then we pick the most frequently used one. For the 1,405 verbs that do not obtain verb class information, we use the original surface text for the predicate feature.

The overall accuracy for Argument Classification is 92.33%, which is about 1% worse than the model trained without verb class information. After a closer look at verb class performance on 28 unseen verbs in the testing data, we found that the accuracy for predicting the semantic roles for unseen verbs in the VerbNet classes is much better than the accuracy for unseen verbs that are not in VerbNet classes. The prediction accuracy of the 16 verbs that do not have VerbNet classes is 75%, and the accuracy on the 12 verbs that are in VerbNet classes is 84.62%.

The SVM model trained without VerbNet class information also did very well on unseen verbs, with an overall accuracy of 85.54%. It seems that our SVM model already has good generalization ability. But since the VerbNet classes only cover about half of the verbs PropBank annotated, that makes the predicate feature inconsistent and might cause some information loss. In future experiments, we would like to focus on the verbs that appear both in PropBank and VerbNet, and augment VerbNet classes with the verb distribution information of the WSJ corpus. Therefore for a verb that has multiple VerbNet class listings, we can take all the classes into account with their probabilities instead of picking a single representative class. We will also compare the usefulness of pre-existing class information for unseen verbs in test data from other genres.

4.2.5 Replacing Head Nouns with Semantic Classes

For building a statistical natural language understanding application, the data sparseness problem is always an issue. The more lexical information extracted for training a system, the worse the data sparseness problem will be. Among several commonly used lexical features, the head noun is the most notorious cause. To reduce this problem, we designed a clustering model, which is a revision of Gildea's verb argument language model (Gildea, 2002) (For the model details, please refer to Sec 4.1.4), and assigned all the head nouns to one of the 64 semantic classes the clustering model created. If the headword is not a noun, then we leave it untouched.

We trained our clustering model with (v, s, n, r) sequences extracted from the PropBank corpus. The result of Argument Classification on using headword semantic classes is 93.3%, which is slightly worse than 93.45%. We suspect that if we train the clustering model with more training data from different genres, our final model would have a better generalization power.

4.2.6 Discussion and Future Directions

Although experiments on feature selection did not show significant improvement with features of word sense information and semantic classes, we still believe these features would prove beneficial with different domains from the training data. Future directions might include the following: extend the EM clustering model with broader coverage both on verbs and nouns, and use it with a word sense disambiguator to create features that carry deeper semantic information, or integrate a state-of-the-art word sense disambiguator in the SRL system framework.

Chapter 5

Robust Semantic Role Labeling via Variation in Parser Approach

The performance of current SRL systems is restricted by the upstream component, the syntactic parser. When we collect argument candidates using the linguistic-based pruning scheme (Xue and Palmer, 2004) on the parse trees given by any state-of-the-art automatic syntactic parser, more than 10% of the arguments are missing. Some missing arguments are due to incorrect argument boundaries; Some missing arguments are because they do not attach to the correct anchor in the parse trees. Other missing arguments are due to POS tagging errors (the parser does not find the target verb, because it was not tagged as a verb).

The imperfection of a syntactic parse naturally generates a recall upper bound for the SRL task, and after errors propagate along the SRL system pipeline, the final performance in terms of F1-measure of an SRL system typically drops below 80 (Carreras and Márquez, 2005).

We need a better parser. We believe that a parser which is suitable for the SRL task should fulfill the following requirements:

1. The parser should be able to distinguish arguments and non-arguments.

2. The parser should be able to form correct dependencies.

3. The parser should be robust enough to handle language variation.

In this chapter, we present experiments aimed at selecting a better parser and improving a parser with semantic argument information. We build our base SRL system based on Xue and Palmer (2004), and had the SRL system process parses from different syntactic parsers. We then create several new parsers trained on a new training corpus created by merging Penn TreeBank (Marcus et al., 1994) constituent labels and PropBank (Palmer et al., 2005a) argument information. All the SRL experiments with different parsers are tested on the WSJ Sec 23.

In order to evaluate parsing robustness, we use the Brown corpus as test data; the Brown corpus is annotated following the Propbank guidelines. Unlike the Penn Treebank which contains only the WSJ articles, the texts of the Brown corpus were sampled from 15 different text categories. This makes the Brown corpus a good test bed for evaluating the adaptability power of an NLP system.

5.1 SRL Using Different Parsers

We build our SRL system according to the model designed by Xue and Palmer (2004) with an additional post-processing component. The post processing component is to further refine the resulting arguments. It filters out arguments according to the following constraints:

1. There are no overlapping arguments.

2. There are no repeating core arguments, unless the repeating one is a relative pronoun.[1]

[1]This constraint is an approximation of following a trace: in the PropBank corpus a verb would also have more than one core argument in a conjunction construction.

We use three syntactic parsers to perform our experiments: 1) The Collins parser (Collins, 1999), 2) Charniak's parser (Charniak, 2000) and 3) Ratnaparkhi's maximum-entropy parser (Ratnaparkhi, 1999). We use Bikel's implementation for the Collins model (Bikel, 2004) and the OpenNLP package[2] for the maximum-entropy parser.

Ratnaparkhi's parser has four maximum-entropy sub-models: TAG, CHUNK, BUILD, and CHECK. It generates a parse tree from the bottom up. Both Collins' and Charniak's parsers generate the parse trees respecting a top-down derivation process based on a lexicalized context-free grammar extracted from the training corpus. Collins also created models to use sub-categorization frames and wh-movement information.

The SRL system and the three parsers are trained on the WSJ corpus, Sec 02-21.

5.1.1 SRL Performance

Table 5.1 shows the SRL final performance on the WSJ Sec 23 test data with different parsers. WSJ Sec 23 has 2,416 sentences, and 14,541 arguments.

Table 5.2 demonstrates the SRL final performance on the Brown corpus (only the portion used by CoNLL-2005 shared task (Carreras and Márquez, 2005)). This portion of the Brown corpus consists of 426 sentences and 1,427 arguments.

Table 5.1: Final SRL System Performance with three different syntactic parsers. Test Data: Sec 23 on 16 semantic roles. # instances: 14,541

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy	78.01	68.98	73.22
Collins	78.87	70.42	74.41
Charniak	81.95	69.68	**75.32**

[2]http://sourceforge.net/projects/opennlp/

Table 5.2: Final SRL System Performance with three different syntactic parsers. Test Data: A small portion of the Brown corpus (used in the CoNLL-2005 shared task) on 16 semantic roles. # instances: 1,427

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy	77.08	63.63	**69.71**
Collins	66.72	55.78	60.76
Charniak	78.09	60.20	67.99

5.1.2 Discussion

Charniak's parser performs the best on the WSJ data. When we test on the Brown corpus, as expected, all three parsers do not perform as well as they did on the WSJ corpus. The Collins model even drops from 74.41 to 60.76. Among the three parsers, the maximum-entropy parser is affected the least while moving to another domain, therefore the maximum-entropy parser becomes the best performing parser on the Brown corpus.

The Collins parser uses heuristics to determine verb sub-categorization frames in order to distinguish verb complements and adjuncts. The design of the Collins parser is tailored to the WSJ corpus. Thus, the Collins parser faces a severe over-fitting problem.

Both the Collins parser and Charniak's parser construct parse trees according to a probabilistic context-free grammar extracted from the Penn TreeBank, while the maximum-entropy parser simply collects features from a context window, and, unlike other generative parsing models, does not implement domain-specific heuristics. We believe that it is why the maximum-entropy parser shows better robustness when moving to different genres.

5.2 Parsing Experiments

In this section, we explore the effectiveness of adding argument information into the training corpus for syntactic parsing. We generated the training corpus by simply

marking the constituents which are also semantic arguments.

5.2.1 Data Preparation

Following standard practice, we use Sec 02-21 of the Penn Treebank and the PropBank as our training corpus. The constituent labels defined in the Penn Treebank consist of a primary label and several secondary labels. A primary label represents the major syntactic function carried by the constituent, for instance, NP indicates a noun phrase and PP indicates a prepositional phrase. A secondary label, starting with "-", represents either a grammatical function of a constituent or a semantic function of an adjunct. For example, NP-SBJ means the noun phrase is a surface subject of the sentence; PP-LOC means the prepositional phrase is a location. Although the secondary labels give us useful information, because of data sparseness and training efficiency, we still stripped off all the secondary labels from the Penn Treebank.

After stripping off the secondary labels from the Penn Treebank, we augment the constituent labels with the semantic argument information from the PropBank. We adopted four different labels, -AN, -ANC, -AM, and -AMC. If the constituent in the Penn Treebank is a core argument, which means the constituent has one of the labels of ARG0-5 or ARGA in the PropBank, we attach -AN to the constituent label. The label -ANC means the constituent is a discontinuous core argument. Similarly, -AM indicates an adjunct-like argument, ARGM, and -AMC indicates a discontinuous ARGM.

For example, the sentence from Sec 02, *[ARG0 The luxury auto maker] [ARGM-TMP last year] sold [ARG1 1,214 cars] [ARGM-LOC in the U.S.]*, would appear in the following format in our training corpus: *(S (NP-AN (DT The) (NN luxury) (NN auto) (NN maker)) (NP-AM (JJ last) (NN year)) (VP (VBD sold) (NP-AN (CD 1,214) (NNS cars)) (PP -AM (IN in) (NP (DT the) (NNP U.S.)))))*

5.2.2 Maximum-Entropy Parser

Using the maximum-entropy parser implementation, we trained three additional parsers. An ANAM-parser was trained on the Penn Treebank corpus augmented with four semantic argument labels: - AN, -ANC, -AM, -AMC. Since the core arguments and the ARGMs in the PropBank loosely correspond to the complements and adjuncts in the linguistics literature, we are interested in investigating their individual effect on parsing performance. Thus, we also trained an AN-parser on the training corpus containing only -AN and -ANC; and another AM-parser on the training corpus containing only -AM and -AMC. Including the baseline maximum-entropy (ME) parser trained on only syntactic information, we now have four different parsers.

Results

Table 5.3 demonstrates the SRL final performance on the WSJ corpus.

We also evaluate our systems on core arguments and adjunct-like arguments respectively. Table 5.4 shows the results on core arguments, and Table 5.5 demonstrates the results on adjunct-like arguments.

In order to explore whether the new training corpus affects the quality of syntactic parsing, we also include the Parseval scores in Table 5.6.

Table 5.3: Final SRL System Performance. Test Data: Sec 23 on 16 semantic roles. # instances: 14,541

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	78.01	68.98	73.22
AN-parser	76.57	68.07	72.07
AM-parser	**78.55**	**69.59**	**73.80**
ANAM-parser	76.77	67.96	72.09

Table 5.4: Final SRL System Performance on Core Arguments. Test Data: Sec 23 on 14 core semantic roles. # instances: 10,726

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	79.37	67.86	73.17
AN-parser	77.65	66.64	71.72
AM-parser	**79.66**	**68.70**	**73.78**
ANAM-parser	77.81	66.68	71.81

Table 5.5: Final SRL System Performance on adjunct-like arguments (AM and C-AM). Test Data: Sec 23 on 2 adjunct-like semantic roles. # instances: 3,815

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	74.63	**72.11**	73.35
AN-parser	73.90	72.08	72.98
AM-parser	**75.72**	72.08	**73.86**
ANAM-parser	74.16	71.56	72.84

Table 5.6: Parseval scores

Parsers	**Label Precision**	**Label Recall**
Maximum-Entropy (ME)	86.99	85.79
AN-parser	86.46	85.29
AM-parser	86.50	85.88

Discussion

The insignificant difference of the Parseval scores between the AN-parser, AM-parser and the base maximum-entropy parser shows that putting additional argument information into the training corpus didn't affect the syntactic parsing performance. With the help of semantic argument information, the AM-parser does better than the base ME parser on semantic role labeling. However, both the AN-parser and the ANAM-parser didn't perform as well as we expected.

A possible explanation that the AM-parser does better than the AN-parser is that complements tend to show up close to the verb, while adjuncts attach to the parse tree more freely. The parsing algorithm is capable of observing local features but is weak at recovering long distance relations, therefore it would benefit more from adjunct-like argument information than from complement information which is more local to the verb in question.

As for the ANAM parser not performing as well as expected: In a sentence, there is often more than one verb, so a constituent can be a core argument of a verb and an adjunct-like argument for another verb in the same sentence. Since we did not include the verb anchor information when we attached the sub labels to the constituents, many constituents obtain both -AN and -AM as their sub labels. This causes confusion and exacerbates the data sparseness problem for the ANAM parser training. We believe the performance of this system can be improved if we can solve this problem.

5.2.3 Collins Parser

We also tried to train the Collins model on our new training corpus which merged the argument labels and the Treebank. In short, the original Collins model identifies a verb complement by its constituent label and its parent node unless it is ruled out by certain Penn Treebank function tags. With our new training data, we adopted two approaches to accommodate the complement and adjunct identification process

required by the Collins parser:

1. The complements are constituents which have an -AN sub-label.

2. The complements are constituents which have an -AN sub-label and those which fulfill Collins heuristics except for the ones ruled out by certain -AM labels.

Looking at the PARSEVAL score, the performance of the newly trained Collins model is similar to the original model. However, the best SRL system using the Collins parser trained on the new training corpus only has an F1 measure of 72.26[3].

The problem we encountered when we trained the ANAM parser (without verb anchor information, -AN and -AM tags might generate confusion) could contribute to the lower performance. But more importantly, we found that the Collins heuristics are not compatible with the PropBank guidelines. For example, Collins model generates subcategorization frames for every verb, including the modal verbs. However, the PropBank tags modal verbs as an ARGM-MOD of the main verbs.

We believe that the Collins model should be able to benefit from data combining the Penn Treebank and the PropBank. However, a careful and close examination of the new training corpus is required to generate a better set of heuristics. In contrast, the maximum-entropy model provides simplicity and flexibility which makes the training straightforward.

5.3 Robustness: the Brown Corpus

We use the Brown corpus, which is annotated following the Propbank guidelines, as our test data to evaluate parsing robustness. Unlike the Penn Treebank which contains only the WSJ articles, the texts of the Brown corpus were sampled from 15

[3]For the rest of the chapter we have continued to use the original Collins parser (Bikel's implementation) trained on the Treebank.

different text categories. This makes the Brown corpus a good test bed for evaluating the adaptability power of an NLP system.

In Sec 5.1, we have learned that the maximum-entropy parser demonstrates the best robustness. In this section, we will investigate how the new training data interact with the new domain.

5.3.1 Results

Table 5.7 shows the SRL final performance on the Brown corpus.

We also evaluate our systems on core arguments and adjunct-like arguments respectively. Table 5.8 shows the results on core arguments, and Table 5.9 demonstrates the results on adjunct-like arguments.

We also include the Parseval scores in Table 5.10.

Table 5.7: Final SRL System Performance. Test Data: the Brown corpus on 16 semantic roles. # instances: 19,801

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	73.17	**61.47**	**66.81**
AN-parser	73.44	60.84	66.55
AM-parser	**73.94**	60.88	66.78
ANAM-parser	72.77	60.54	66.10

Table 5.8: Final SRL System Performance on Core Arguments. Test Data: the Brown corpus on 14 core semantic roles. # instances: 13,743

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	75.63	62.69	68.56
AN-parser	75.94	62.26	68.43
AM-parser	**76.29**	**63.02**	**69.02**
ANAM-parser	74.98	62.02	67.69

Table 5.9: Final SRL System Performance on adjunct-like arguments (AM and C-AM). Test Data: Sec 23 on 2 adjunct-like semantic roles. # instances: 6,058

Parsers	**Precision**	**Recall**	**F1**
Maximum-Entropy (ME)	67.82	**58.68**	**62.92**
AN-parser	67.96	57.59	62.35
AM-parser	**68.56**	56.01	61.65
ANAM-parser	67.86	57.18	62.06

Table 5.10: Parseval scores

Parsers	**Label Precision**	**Label Recall**
Maximum-Entropy (ME)	80.25	78.60
AN-parser	79.64	78.17
AM-parser	79.66	79.58

5.3.2 Discussion

As shown from the above tables, the AM-parser is still the best performing parser among the three parsers trained on the merged corpus in terms of the final SRL performance. Also, the AM-parser has better Parseval scores than the AN-parser.

However, when we compare the AM-parser with the base ME parser, the final SRL performance is almost identical in terms of the F1-measure. After a closer look on the SRL performance on the core arguments and the adjunct-like arguments, we discover that the AM-parser performs better on the core arguments, but the recall number of the AM-parser drops significantly when we test on the adjunct-like arguments. Therefore, the AM-parser performs the worst on the adjunct-like arguments. This phenomenon does not appear when we run the same experiments on the WSJ corpus, and given that Parseval scores indicate that the AM-parser has the best recall number, it becomes a puzzle to us. We will do a thorough error analysis on this specific problem to determine what goes wrong when the AM-parser is used for classifying adjunct-like arguments.

5.4 Discussion

In order to obtain a parser suitable for the SRL task, we create a new training corpus by merging the argument information extracted from the PropBank into the Penn TreeBank. The AM-parser, which was trained with the mark-ups of the adjunct-like arguments, shows improvement on the SRL task over the base maximum-entropy parser on the WSJ corpus, and on the core arguments when testing on the Brown corpus. When evaluating the syntactic parsing accuracy with the Parseval evaluation metrics, the AM-parser is comparable with the base maximum-entropy parser on the WSJ corpus, and the AM-parser has better recall on the Brown corpus.

It is encouraging that the newly trained parser, the AM-parser, generally has better SRL performance than the best maximum-entropy parser without sacrificing the Parseval scores. However, some of the improvement is not significant. We believe that, in addition to the increased number of constituent labels making the data sparseness problem worse, the following problems are the major ones:

Noisy Training Data (explained in more detail in Sec 5.4.1): 1. We need only constituent-level arguments. 2. We do not need AM-MOD and AM-NEG. 3. We need to sort out the discrepancies between the PropBank and the Penn TreeBank.

Verb Anchor Problem: When we create the new training corpus, we indicate whether a constituent is an argument or not; however, when a constituent is an argument, we simply attach a sub-label of either -AN or -AM without stating its predicate. This piece of information is important both for learning the right dependency and correctly labeling a semantic role.

5.4.1 Noisy Training Data

Arguments whose label are attached to the POS tags

The purpose of putting argument information into a syntactic tree is for the parser to better learn the correct dependency structures within a sentence. Therefore, we should only put the information of constituent-level complements and adjuncts into the tree. However, there are some cases that the -AN, or -AM tags would attach to a POS tag, for example, the *NN-AN* tag in 1, and 2. The sub-labels attached to the POS tags reduce the accuracy of the POS tagger, which is an important component of a syntactic parser.

1. (NP(DT an)(ADJP(RB-AM ingeniously)(VBN chosen))(***NN-AN*** potpourri))
2. (NP(NP(DT the)(VBG ruling)(***NN-AN*** party)(POS 's))(NN fate))

Arguments that are either complements or adjuncts

AM-MOD and AM-NEG, which are annotated in the PropBank corpus, are neither complements nor adjuncts. AM-MOD indicates the modal verbs, and AM-NEG shows when there is a negation. These two tags should be removed from the new training corpus.

1. AM-NEG (VP(VBZ has)(RB-***AM[-NEG]*** n't)(VP(VBN been)(VP(VBN set))))

2. AM-MOD (VP(MD-***AM[-MOD]*** may) (VP fail to file the reports))

Arguments that are interpreted differently in the PropBank and the Penn TreeBank

The PropBank sometimes has to annotate an argument discontinuously; for example: *[ARG1 Absent other working capital], he said, [C-ARG1 the RTC would be forced to delay other...].* However, sometimes the PropBank annotates an argument discontinuously to accommodate conflicts between the PropBank and the Penn TreeBank

annotations, because the argument which the PropBank regards as a continuous argument is not a constituent in the Penn TreeBank. In the following example, *a group of workers* is the ARG1 of the predicate *expose*; however, in the Penn TreeBank, *a group*, *of*, and *workers* do not form a constituent, therefore, the PropBank has to tag them as ARG1, C-ARG1, and C-ARG1.

PropBank annotation: *A form of asbestos once used to make Kent cigarette filters has caused a high percentage of cancer deaths among [ARG1 a group] [C-ARG1 of] [C-ARG1 workers] exposed to it . . . .*

Penn TreeBank annotation: . . . (VP (VBZ has) (VP (VBN caused) (NP (NP (DT a) (JJ high) (NN percentage)) (PP (IN of) (NP (NN cancer) (NNS deaths))) (PP-LOC (IN among) (NP (NP (DT a) (NN group)) (PP (IN of) (NP (NP (NNS workers)) (RRC (VP (VBN exposed) (PP-CLR (TO to) (NP (PRP it))) (ADVP-TMP (NP (QP (RBR more) (IN than) (CD 30)) (NNS years)) (IN ago))))))))))) . . .

5.5 System Combination

We also implemented a simple voting algorithm for combining systems to further explore the differences among parsers. For every argument proposed, we count the votes it gathers from each participating system. The more votes an argument wins, the higher priority it gets. The way to break a tie is based on the confidence level of the argument given by the system. Whenever we pick an argument, we need to check whether this argument conflicts with previously selected arguments in order to assure that: 1) There are no overlapping arguments; 2) There are no repeating core arguments, unless the repeating one is a relative pronoun.

Table 5.11 presents the final SRL results of various system combinations on Sec 23 of the WSJ corpus.

Table 5.11: Final SRL System Performance on System Combination. Test Data: Sec 23 on 16 semantic roles. ME (maximum-entropy parser), CK (Charniak's parser), DB (the Collins parser). # instances: 14,541

Parsers	**Precision**	**Recall**	**F1**
AN+AM	77.05	71.35	74.09
ME+DB	77.71	73.59	75.59
DB+CK	79.12	73.94	76.44
AM+CK	79.74	75.33	77.47
ME+CK	79.75	75.42	**77.52**
ME+CK+DB	79.88	76.69	**78.25**
ME+AN+AM	77.15	71.74	74.35
AM+CK+DB	79.88	76.60	78.20

When we combine systems, we take advantage of their variety. However, there is a trade-off between the quality of the individual systems and the number of participating systems. The gain from the variety can be penalized by the loss from introducing noise.

In the WSJ corpus, the best performing SRL system uses the CK parser (Charniak's parser), and then the DB parser (the Collins parser); However, the combination of these two systems do not yield the best results. The best combination of two systems is ME+CK. The performance of AM+CK is similar to ME+CK. The results imply that the maximum-entropy parser and Charniak's parser generate more different parse trees than other combinations. The combinations of ME+CK+DB and AM+CK+DB have similar performance and both of them are comparable to the state-of-the-art performance (Punyakanok et al., 2005).

Because we don't have the complete Brown corpus parsed by Charniak's parser, we demonstrate the effects of system combination on a small portion of the Brown corpus used by the CoNLL-2005 shared task. Table 5.12 shows the results.

The best 2-system combination is still ME+CK. ME+CK is almost the best combination out of the 4 combinations shown in the table, since the Collins model didn't perform well on the Brown corpus, adding DB to the combination actually hurts the performance rather than improving it.

Table 5.12: Final SRL System Performance on System Combination. Test Data: A small portion of the Brown corpus (used in the CoNLL-2005 shared task) on 16 semantic roles. # instances: 1,427

Parsers	**Precision**	**Recall**	**F1**
ME+DB	70.48	64.75	67.49
DB+CK	71.11	64.68	64.74
ME+CK	77.40	70.08	**73.56**
ME+CK+DB	74.72	70.01	72.29

Our algorithm for combining the systems is simple. We plan to train a classifier to assign appropriate weights to participating systems, in order to try the joint reference approach introduced by Punyakanok et al. (Punyakanok et al., 2005).

5.6 Error Analysis

In order to enhance the performance and robustness of our SRL system, we need to pinpoint the causes of missing arguments and mis-labeled arguments. We performed some error analysis and the main objective of the error analysis is to find out whether a mistake is due to a parsing error or an SRL error. Furthermore, we compared the results generated by the AM-parser and the Collins parser. By investigating the strengths and weaknesses of these two different parsers, not only will we be able to improve the performance of a single parser but also we can figure out a better way to create an ensemble SRL system.

5.6.1 Error Classification

The error analysis is a multi-step process, and we look into missing arguments, mis-labeled arguments and false alarms. Because both of the Argument Identification and the Argument Classification components are trained and will make decisions on features extracted from a parse tree, it is difficult to draw a clear line between a parsing error and an SRL error. In general, we categorize errors by their appearance

in the whole pipeline.

For missing arguments, we first compare the final results with the gold answers and detect arguments that are missed because the predicate itself has been missed; that is, the predicate is not tagged as a verb in the parse tree and therefore this mistake is categorized as a parsing error. For other missing arguments, we try to locate their final appearance by back-tracking through the SRL system pipeline. If the argument is still present right after Argument Classification, Argument Identification, or the Pruning component, we consider the missed argument is due to an SRL error. If the argument is not even present after the pruning stage, we will determine whether it is because it has not been attached to the correct location or it is not a constituent at all in the parse tree. Either case is considered a parsing error.

We consider mis-labeled arguments to be SRL errors. False alarms are the most difficult to categorize but we basically consider them as parsing errors.

We analyzed the results generated by two SRL system pipelines using the Collins parser and the AM-parser. On the WSJ corpus, it seems that the AM-parser performs worse than the Collins parser with respect to every kind of parsing error; that is, the AM-parser has more verbs mis-tagged as other POS tags, it has slightly more mis-attached constituents, slightly more mis-defined constituents (i.e. the constituents do not have correct boundaries). However, on the Brown corpus, although the AM-parser still has more verbs mis-tagged, it performs significantly better than the Collins parser on constituent boundary detection and constituent attachment, so it has an advantage with respect to robustness.

5.6.2 Experiments on Parser Enhancement

We suspect that the Collins parser performs better on verb tagging because the most probable POS tag given by a POS tagger at an early stage can still be outweighed by another POS tag candidate at any moment before the parse is completed. However,

the POS tagger in the AM-parser is more straightforward and is hardly influenced at all by later stages.

We tried to enhance the POS tagger in the AM-parser by putting more verbs in its tag dictionary; however, it did not work as well as expected. In addition, since Bikel's implementation of the Collins parser takes partial parses as input, we fed the Collins parser partial parse trees generated by the AM-parser. We hoped that by doing this, the Collins parser could take advantage of the AM-parser's superiority on constituent boundary detection and constituent attachments. However, the results turned out to be much worse.

5.6.3 Discussion

In conclusion, if we want to get better parse results in order to improve SRL system performance without making any revolutionary changes to the implementation of current state-of-the-art parsers, then we have to be more creative. A better approach is to carefully examine the problem of noisy training data. It appears that the Treebank and the PropBank have many disagreements with respect to defining arguments. Inconsistent and contradictory training data is a significant barrier to the creation of a well-performed parser and a robust SRL system.

5.7 Conclusion

A syntactic parser is a very important upstream component for many NLP tasks such as word sense disambiguation, information extraction, and semantic role labeling. However, performance of a syntactic parser is often judged by label Precision and Recall on a single corpus, therefore, designers of syntactic parsers intentionally/unintentionally encode domain specific knowledge which leads to data overfitting.

Given that Semantic Role Labeling (SRL) requires a syntactic parser to generate

parse trees with both correct boundaries and dependencies, the SRL task can be an objective indication of parse tree quality. In this chapter, we investigate the impact of different syntactic parsing frameworks on the SRL task. Furthermore, we perform experiments on different genres (Brown corpus) to test parsing robustness.

We created a new training corpus by merging the argument information from the Propbank directly onto the Penn Treebank constituent labels. Our results demonstrate that the maximum-entropy style parser benefits from our new training corpus while the Collins style parser experiences a performance decrease. Furthermore, when moving to the Brown corpus, the maximum-entropy style parser shows better adaptability. We also demonstrate, as expected, that by combining different systems that exhibit enough variety with a simple voting algorithm, we can boost the SRL performance to an even higher level.

Chapter 6

The Discrepancies between Penn Treebank and PropBank

One of the serious issues current SRL systems face is noisy training data. Most of the current SRL systems are trained on the PropBank corpus. As described previously, the PropBank annotates the entire Penn Treebank with predicate argument structures by adding semantic role labels to the syntactic constituents. Theoretically, there should be a one-to-one mapping between syntactic constituents and semantic arguments; however, as we discussed in Section 5.4.1, in reality PropBank annotators sometimes could not find a constituent to assign an argument label and therefore had to assign an argument label to a POS node or to multiple constituents.

Shen (2006) also notices this problem and reports that the fact that one PropBank argument can map to several Penn Treebnk constituent nodes poses a problem for automatic extraction of LTAG trees out of an integrated Penn Treebank and PropBank. Shen tried to reconcile discrepancies when one semantic argument (PropBank argument) is mapped to multiple syntactic constituents (TreeBank nodes). For most mismatching cases, he integrated Penn Treebank and PropBank by tree transformations based on PropBank annotation, i.e. the tree transformation merges Treebank

nodes and create a new constituent which maps to the PropBank argument. After the integration, a LTAG elementary tree can be extracted. However, for other remaining mismatching cases when Treebank nodes cannot be merged because the segments are not continuous, arguments are either associated with LTAG auxiliary trees instead of elementary trees or left as is.

Furthermore, semantic arguments of a verb theoretically should only appear in a finite set of locations: sister nodes of the verb or sister nodes of the verb's ancestor nodes; therefore the Pruning component (described in Section 3.2.1) in an SRL system pipeline should be able to include all possible argument candidates. However, the Penn Treebank does not always distinguish syntactic arguments and adjuncts by demonstrating a structural difference (Babko-Malaya et al., 2006); therefore in reality we found semantic arguments were sometimes buried deeply in a sub-tree of a verb complement. In addition, the annotators of the Penn Treebank and the PropBank have disagreements between whether a constituent is a sentential argument or a predicate argument; therefore they had different opinions about the correct location of the constituent in question.

In this chapter, we first demonstrate the experiments we performed in an attempt to solve the problems stated in Sec 5.4.1. We basically removed the argument labels which are attached to a POS node and took off the AM-MOD and AM-NEG arguments. We then evaluated the results and loosened the evaluation metrics by counting it as correct if the SRL system found and assigned the correct argument label to the first chunk of a discontinuous argument.

There was no significant difference in SRL system performance from the above experiments. We realize that in order to make the training data more consistent and less contradictory, a thorough investigation of the discrepancies between the Penn Treebank and the PropBank is required. We therefore describe how we extracted the mismatch cases between the Penn Treebank and the PropBank and depict the discrepancies in later sections. We conclude this chapter by stating the plans to

reconcile the mismatches. In the next chapter we will list the modifications of both Penn Treebank and the PropBank and describe the implementation details of synchronizing the two corpora, and evaluate the impact of the new synchronized data on SRL.

The project of synchronizing Penn Treebank and PropBank is joint work accomplished with Olga Babko-Malaya, Ann Bies, Ann Taylor and many others. Taylor was in charge of all the Treebank changes, and Babko-Malaya was the main policy examiner of the PropBank corpus. My contribution to this project focuses on the extraction and categorization of mismatching cases, the implementation of automatic synchronization of Penn Treebank and PropBank, and the evaluation of the new data on the SRL task.

6.1 Removing Non-Semantic Arguments from the Training Data

In Chapter 5, we presented several parsers which were trained on the merged Penn Treebank and PropBank corpora. The merging of two corpora was done by adding -AN or -AM tags to the Treebank constituent labels if the constituent was a numbered argument or an ARGM argument in the PropBank. After carefully examining the training data, we found that there were some discrepancies between the two corpora, which made the merged training data inconsistent and contradictory. The most obvious discrepancy was demonstrated by the fact that a POS tag can be a semantic argument.

In order to clean up the noisy training data, we removed the -AN/-AM tags which were attached to a POS tag. We then examined the 13 ARGM labels carefully in order to remove the ARGM labels which are neither complements nor adjuncts. In the final training data, we didn't include ARGM-MOD, ARGM-NEG, ARGM-REC and ARGM-DIS because these four argument labels should be handled at the lexical

level: ARGM-MOD indicates the modal verbs; AM-NEG shows when there is a negation; ARGM-REC is a reciprocal argument which refers back in turn to one of the other arguments; ARGM-DIS is a discourse marker which reveals information in a larger context outside the scope of predicate argument structures (Babko-Malaya, 2005). Below are some example sentences of these four ARGM labels:

1. **ARGM-MOD** [may] fail to file the reports

2. **ARGM-NEG** has [n't] been set

3. **ARGM-REC** John and Bob killed [each other]

4. **ARGM-DIS** [But] you have to recognize that...

We trained several different parsers on the new training corpus, and the best F-1 score was achieved by the SRL system which used the parses generated by the AM-parser. The F-1 score was 78.93 which is slightly better than 73.80 (AM-parser trained on the old merged data); however, the difference was not significantly different. There are many discrepancies between the two corpora that were not captured by our simple adjustment. A thorough examination of the two corpora is required in order to obtain consistent and non-contradictory training data.

6.2 Extractions of mismatches between the Penn Treebank and PropBank

In order to reconcile the Penn Treebank and PropBank, we first have to extract instances of mismatches for analysis. We wrote a script which extracted possible mismatches by examining the following two cases: 1) argument locations; 2) discontinuous arguments. We examined argument locations because a PropBank argument should only appear in a finite set of locations relative to the verb in question. There might be a mismatch if an argument embeds deeply in a subtree of a sister node of

the predicate, or if an argument attaches to another verb spine. We examined discontinuous arguments because the PropBank annotators concatenate constituents to form a discontinuous argument if they cannot find a single constituent for the argument, and this might indicate a Treebank error. Discontinuous arguments also signal different treatments of the Penn Treebank and PropBank for cases involving traces and empty categories.

6.2.1 Examining Argument Locations

In a parse tree which expresses the syntactic structure of a sentence, a semantic argument occupies specific syntactic locations: it appears in a subject position, a verb complement location or an adjunct location. Relative to the predicate, its argument is either a sister node, or a sister node of one of the predicate's ancestors. We extracted cases of PropBank arguments which do not attach to the predicate spine, and filtered out VP coordination cases. For example, the following case is a problematic one because the argument PP node is embedded too deeply in an NP node hence it cannot find a connection with the main predicate verb *lifted*. This is an example of a PropBank annotation error:

[1] (VP (VBD[**rel**] *lifted*)
(NP us))
(NP-EXT
(NP a good 12-inches)
(PP-LOC[**ARGM-LOC**] above
(NP the water level))))

However, the following case is not problematic because we consider the ArgM PP to be a sister node of the predicate verb given the VP coordination structure:

[2] (VP (VP (VB[**rel**] *buy*)
(NP the basket of K)
(PP in whichever market K))
(CC and)
(VP (VBP sell)
(NP them)
(PP[**ARGM**] in the more
expensive market)))

6.2.2 Examining Discontinuous Arguments

Discontinuous arguments happen when PropBank annotators need to concatenate several Treebank constituents to form an argument. Discontinuous arguments often represent different opinions between PropBank and Treebank annotators regarding the interpretations of the sentence structure.

For example, in the following case, the PropBank concatenates the NP and the PP to be the Arg1. In this case, the disagreement on PP attachment is simply a Treebank annotation error.

[3] *The region lacks necessary mechanisms for handling the aid and accounting items.*
Treebank annotation:
(VP lacks

(NP necessary mechanisms)

(PP for

(NP handing the aid...)))

PropBank annotation:

REL: lacks

Arg1: [NP necessary mechanisms][PP for handling the aid and accounting items]

6.3 Type of Mismatches

We grouped and classified all the mismatches into several categories: 1) modifier attachment ambiguities, 2) sentential complements, 3) phrasal verbs and light verbs, 4) conjunctions, and 5) ICH traces and verbs of saying. Besides a small portion of mismatches that are due to annotation errors, the common causes of mismatches include different interpretations of modifiers and preferences with respect to syntactic considerations versus semantic considerations by the Penn Treebank and PropBank annotators. Also the Penn Treebank and the PropBank have different policies regarding different types of links including syntactic chains, semantic co-reference and semantic types; and these different policy decisions also generate some mismatching cases. In the following sections, we explain the different types of mismatches and give examples.

6.3.1 Modifier Attachment Ambiguities

Modifier Attachment Ambiguities include PP attachment ambiguity and Adverb attachment ambiguity. The former needs careful manual adjudication and the latter usually can be resolved automatically.

PP Attachment Ambiguities

A PP attachment ambiguity occurs when a prepositional phrase can either modify a verb or an NP; therefore it can attach either to a verb or an NP. From the mismatched cases we extracted, we discovered that in the Penn Treebank, the annotators tend to attach the PP to the NP; in the PropBank, the annotators usually consider the PP a modifier of the verb in question; therefore the PP is assigned an argument label.

Here is an example of the verb *write* (Babko-Malaya et al., 2006), which is genuinely ambiguous:

[4] *She wrote a letter for Mary.*

Treebank annotation:

```
(VP wrote
   (NP (NP a letter)
      (PP for
         (NP Mary))))
```

PropBank annotation:

REL: write

Arg1: a letter

Arg2: for Mary

Adverb Attachment Ambiguities

An adverb attachment ambiguity occurs when the Penn Treebank thinks a modifier (usually an adverb) attaches to a PP while the PropBank considers the modifier to be associated with the predicate; therefore it should have been attached to the verb. The most common cases in this category involve a directional adverbial which follows the verb. In Treebank, this adverbial is part of an ADVP which attaches to the verb; in PropBank, this adverbial is considered an argument and is labeled ARGM-DIR. Example [5] demonstrates this scenario. Example [6] gives an example of other possible cases in this category. To reconcile the mismatched cases in this category,

PropBank can follow the Treebank annotations and make the change automatically.

[5] *Mediobanca said during the weekend that it agreed to sell the shares back to Giovanni Agnelli for 333 billion lire.*

Treebank annotation:

```
( (S
    (NP-SBJ (NNP Mediobanca) )
    (VP (VBD said)
      (PP-TMP during the weekend)
      (SBAR (IN that)
       (S
        (NP-SBJ-2 (PRP it) )
        (VP (VBD agreed)
          (S
            (NP-SBJ (-NONE- *-2) )
            (VP (TO to)
             (VP (VB[rel] sell)
               (NP (DT the) (NNS shares) )
               (ADVP-CLR (RP[ARGM-DIR] back)
                (PP[ARG2] (TO to)
                  (NP (NNP Giovanni) (NNP Agnelli) )))
               (PP-CLR for 333 billion lire))))))) (. .) ))
```

Original PropBank annotation:

Rel: sell

ARGM-DIR: back

ARG2: to Giovanni Agnelli

Revised PropBank annotation:

Rel: sell

ARG2: back to Giovanni Agnelli

[6] *Our pilot simply laughed, fired up the burner and with another blast of flame lifted us, oh, a good 12-inches above the water level.*

Treebank annotation:

```
( (S
    (NP-SBJ (PRP Our) (NN pilot) )
    (VP
      (ADVP (RB simply) )
      (VP (VBD laughed) )
      (, ,)
      (VP (VBD fired)
       (PRT (RP up) )
       (NP (DT the) (NN burner) ))
      (CC and)
      (VP
       (PP with another blast of flame )
       (VBD[rel] lifted)
       (NP (PRP us) )
       (, ,) (INTJ (UH oh) ) (, ,)
       (NP-EXT
       (NP[ARG2-EXT] a good 12-inches )
          (PP-LOC[ARGM-LOC] above the water level )))) (. .) ))
```

Original PropBank annotation:

Rel: lifted

ARGM-LOC: above the water level

ARG2-EXT: a good 12 inches

Revised PropBank annotation:

Rel: lifted

ARG2-EXT: a good 12 inches above the water level

6.3.2 Sentential Complements

The category of sentential complements is the source of many mismatches between the Penn Treebank and the PropBank. In general, a mismatch involving a sentential complement occurs when the Treebank thinks that the verb takes an S node as its complement while the PropBank considers the sub-constituents of the S node to be in fact arguments of the verb. Babko-Malaya et al (2006) analyzes the background and the source of this type of mismatch in detail. Basically the PropBank determines whether a verb should take a sentential argument based on the semantic interpretation of the arguments of the predicate while the Treebank puts more weight on syntactic considerations. This type of mismatch can only be reconciled on a verb-by-verb basis. Example[7] and Example[8] demonstrate mismatching cases in this category.

[7] *Speculation about a takeover fight has sent Jaguar shares soaring in the past six weeks.*

Treebank annotation:

```
( (S
    (NP-SBJ
      (NP (NN Speculation) )
      (PP about a takeover fight )
    (VP (VBZ has)
      (VP (VBN[rel] sent)
       (S
          (NP-SBJ[ARG1] (NNP Jaguar) (NNS shares) )
          (VP[ARG2] (VBG soaring) ))
       (PP-TMP (IN in)
          (NP (DT the) (JJ past) (CD six) (NNS weeks) )))) (. .) ))
```

PropBank annotation:

Rel: sent ARG1: Jaguar shares

ARG2: soaring in the past six weeks

[8] *It unfortunately encourages others to engage in a highly dangerous and illegal activity that only a very few are doing now.*

Treebank annotation:

```
( (S
    (NP-SBJ (PRP It) )
    (ADVP (RB unfortunately) )
    (VP (VBZ[rel] encourages)
      (S
       (NP-SBJ[ARG1] (NNS others) )
       (VP[ARG2] (TO to)
        (VP (VB engage)
          (PP-CLR (IN in)
            (NP
             (NP (DT a)
                (ADJP highly dangerous and illegal)
                (NN activity) )
             (SBAR
                (WHNP-1 (IN that) )
                (S
                  (NP-SBJ (RB only) (DT a) (RB very) (JJ few) )
                  (VP VBP are doing now))))))))) (. .) ))
```

PropBank annotation:

Rel: encourages

ARG1: others

ARG2: to engage in a highly dangerous and illegal activity that only a very few are doing now

6.3.3 Phrasal Verbs and Light Verbs

Phrasal verbs (such as *turn on*) and light verbs (such as *make use of*) share a similar dilemma of whether the whole construction should be treated as a predicate or only the verb is considered the predicate. The Penn Treebank treats the second part of a phrasal verb as a part of a prepositional phrase while the PropBank considers an entire phrasal verb as a predicate having its own predicate arguments. The Penn Treebank analyzes each part of a light verb construction separately while the PropBank considers the part following the verb an argument. Example [9] is an example of phrasal verbs. Example [10] gives an example of light verbs.

[9] *But Japanese institutional investors are used to quarterly or semiannual payments on their investments, so...*

Treebank annotation:

(NP-SBJ-167 (JJ Japanese) (JJ institutional) (NNS investors))
(VP (VBP are)
(VP (VBN used)
(NP (-NONE- *-167)) (PP-CLR (TO to)
(NP[**ARG1**]
(NP
(ADJP (JJ quarterly) (CC or) (JJ semiannual))
(NNS payments))
(PP (IN on)
(NP (PRP their) (NNS investments))))))))

PropBank annotation:

Rel: used to

ARG1: quarterly or semiannual payments on their investments

[10] *The Chinese can not be seen to have made use of the Khmer Rouge and then discard them.*

Treebank annotation:

((S
(NP-SBJ-1 (DT The) (NNPS Chinese))
(VP (MD can) (RB not)
(VP (VB be)
(VP (VBN seen)
(S
(NP-SBJ (-NONE- *-1))
(VP (TO to)
(VP (VB have)
(VP
(VP (VBN[**rel**] made)
(NP
(NP[**ARG1**] (NN use))
(PP[**ARG2**] (IN of)
(NP (DT the) (NNP Khmer) (NNP Rouge)))))
(CC and)
(ADVP (RB then))
(VP (VB discard)
(NP (PRP them)))))))))) (. .) (" ")))

PropBank annotation:

Rel: made

ARG1: use

ARG2: of the Khmer Rouge

6.3.4 Conjunctions

A mismatch in the Conjunction category occurs when an adverbial modifies a conjunction. In Penn Treebank, the adverbial only modifies one member of the conjunction while in the PropBank the adverbial modifies the whole conjunction; therefore

this adverbial is an argument of the verb of each member of the conjunctions. Example [11] is an example of this type of mismatch.

[11] *In China, a great number of workers are engaged in pulling out the male organs of rice plants using tweezers, and one-third of rice produced in that country is grown from hybrid seeds.*

Treebank annotation:

```
( (S
    (S
      (PP-LOC[ARGM-LOC] In China ) (, ,)
      (NP-SBJ a great number of workers)
      (VP (VBP are)
        (VP (VBN[rel] engaged)
          (NP-2 (-NONE- *-18) )
          (PP-CLR in pulling out the male organs of rice plants using tweezers)
    (, ,)
    (CC and)
    (S
      (NP-SBJ one-third of rice produced in that country )
      (VP (VBZ is)
        (VP (VBN[rel] grown)
          (NP (-NONE- *-19) )
          (PP-CLR (IN from)
            (NP (JJ hybrid) (NNS seeds) ))))) (. .) ))
```

PropBank annotation:

Rel: engaged

ARGM-LOC: In China

Rel: grown

ARGM-LOC: In China

6.3.5 ICH Traces and Verbs of Saying

Most of the PropBank discontinuous arguments fall into this category. ICH as used in the Penn Treebank stands for Interpret Constituent Here. It indicates that two separate constituents that are separated by some text or other material should in fact be a single constituent (Bies et al., 1995). As shown in Example [12], the syntactic complement of the verb *begin* is *hearings on their proposal to...*; however, *hearings* and *on their proposal to...* are separated by *next week*; therefore an ICH trace is used to link to the PP *on their proposal to... hearings*. In the PropBank, when an annotator tags an argument which contains an ICH trace, they include the constituent linked by the ICH trace as well and therefore create a discontinuous argument.

As for verbs of saying, if the complement of a verb is separated by other material, in Penn Treebank a trace is placed at the location of syntactic complement of the verb and this trace is linked back to the maximal S node. In the PropBank, all the parts of the complement are tagged as parts of the argument of the verb and the trace is also included. Example [13] is an example of such case.

[12] *The Senate Banking Committee will begin hearings next week on their proposal to expand existing federal housing programs.*

Treebank annotation:

```
( (S
    (NP-SBJ The Senate Banking Committee)
    (VP (MD will)
      (VP (VB begin)
        (NP
          (NP (NNS hearings) )
          (PP (-NONE- *ICH*-1) ))
        (PP-TMP (IN next)
          (NP (NN week) ))
```

(PP-LOC-1 (IN on)
(NP (PRP their) (NN proposal)
(S
(NP-SBJ (-NONE- *))
(VP (TO to)
(VP expand existing federal housing programs)))))))) (. .)))

PropBank annotation:

Rel: begin

ARG1: hearings *ICH-1*, on their proposal to expand existing federal housing programs [13] *Nevertheless , said Brenda Malizia Negus , editor of Money Fund Report , yields " may blip up again before they blip down "*

Treebank annotation:

((S-1
(ADVP[**ARG1**] (RB Nevertheless))
(PRN
(, ,)
(SINV
(VP (VBD said)
(S[**C-ARG1**] (-NONE- *T*-1)))
(NP-SBJ Brenda Malizia Negus , editor of Money Fund Report) (, ,))
(NP-SBJ[**C-ARG1**] (NNS yields))
(“ “)
(VP[**C-ARG1**] (MD may)
(VP blip up again before they blip down)) (" ")
(PP-PRP because of recent rises in short-term interest rates))) (. .)))

PropBank annotation:

Rel: said ARG1: Nevertheless, *T*-1, yields " may blip up again before they blip down " because...

6.4 Summary

Noisy training data poses challenges to current SRL systems. In order to create more consistent and less contradictory training data, we investigated the discrepancies between the Penn Treebank and the PropBank. The extracted Treebank/PropBank mismatch cases fall into the following categories 1) modifier attachment ambiguities, 2) sentential complements, 3) phrasal verbs and light verbs, 4) conjunctions, and 5) ICH traces and verbs of saying. Some of the mismatch cases can be reconciled automatically, some require careful manual adjudication, and some demand policy changes in either the Penn Treebank or the PropBank guidelines. In the next chapter, we will report the current effort and progress on synchronizing the Treebank and PropBank and discuss the impact of reconciling the two corpora on SRL system performance.

Chapter 7

Synchronizing the Penn Treebank and PropBank

In chapter 6 we talked about the discrepancies between the Treebank and the PropBank. Efforts have been made to synchronize these two corpora for a selected 300K of data in order to facilitate the study of the syntax/semantic interface and the implementation of semantic role labeling.

Treebank changes include: 1) the introduction of a new empty category *PRO* to replace a subset of the cases represented by the empty category * in the original Treebank; 2) a clearer distinction between verbs which take small clauses and secondary predicates; and 3) the standardization of S-conjunctions. PropBank changes include: 1) the definition of three distinct types of chains which will be annotated as a three-stage process; 2) the modification of frame files of verbs involved in mismatches in the "sentential complements" category.

We assessed the quality of the synchronized data by comparing the performance of SRL taggers using different parsers on the old data and the new synchronized data. We evaluated the overall performance, the performance on arguments linked by traces, and the performance on arguments associated with different verb types. The observations of the various SRL results suggest that the noisy data problem

might still exist due to the fact that the synchronization is not yet complete.

7.1 Other Attempt on the Integration of Penn Treebank and PropBank

Other attempts at integrating the Penn Treebank and PropBank include the one described in Shen (2006). Shen uses PropBank to automatically construct an LTAG treebank; however, the fact that there is not necessarily a one-to-one mapping between PropBank arguments and Treebank constituents was a problem during the process. For most mismatching cases, when the constituents are continuous segments in the sentence, Shen used tree transformations which modified the structure of the syntactic trees to respect the PropBank annotations. Basically the tree transformation procedure merges constituents into one big constituent; therefore a new constituent node has to be created. For the remaining mismatching cases, usually when the constituents are discontinuous segments in the sentence, PropBank arguments are either mapped to LTAG auxiliary trees instead of elementary trees or left as is. The PropBank arguments which are handled with LTAG auxiliary trees include those associated with raising verbs, passive ECM verbs and many cases of the *say* verb. This approach allowed Shen to automatically produce LTAG trees with PropBank arguments, and also provided invaluable input into the detection of discrepancies between the Treebank and the PropBank, for which we are grateful.

7.2 Treebank Changes

Other than correcting trivial annotation errors and putting back missing traces, Treebank has made the following changes directly related to the merge of Treebank and PropBank: 1) a new empty category *PRO* is introduced to distinguish raising verbs and control verbs and to separate di-transitive and mono-transitive constructions; 2)

The boundaries are redrawn between the verbs which take small clauses and those which take objects and secondary predicates; 3) S-conjunction construction is standardized and the occasions of using Parentheticals tag (PRN) are redefined (Taylor, 2006).

7.2.1 New Empty Category *PRO*

The new Treebank guidelines introduce a new empty category *PRO* to replace a subset of cases which are represented by * in the original Treebank. In the new Treebank, * is only used for the empty categories caused by syntactic movements, i.e. passive and raising constructions, while *PRO* is used in control structures and for arbitrary PROs. The new Treebank changed some of the original Treebank ECM verbs (monotransitives) to di-transitive with object control, for example, the verb 'force' now has a di-transitive construction and uses the empty category *PRO*. In this way, the behavior of 'force' more closely matches that described in the PropBank frame file.

Below is an example: The original Treebank/PropBank annotation was a instance of a mismatch. Based on the old Treebank annotation, PropBank has to annotate two sub-constituents of the S node which is an S-complement of the predicate in question as ARG1 and ARG2. With the new Treebank annotation, PropBank therefore can annotate two sister nodes of the predicate as arguments.

[1] *This form forces a lawyer to become , in effect , a witness against his client...*

Original Treebank annotation:

```
( (SINV (“ “) (S-TPC-1 (NP-SBJ[ARG0] (DT This) (NN form) )
    (VP (VBZ[REL] forces)
      (S
       (NP-SBJ[ARG1] (DT a) (NN lawyer) )
       (VP[ARG2] (TO to)
        (VP (VB become)
```

(PRN

(, ,)

(PP (IN in)

(NP (NN effect)))

(, ,))

(NP-PRD

(NP (DT a) (NN witness))

(PP (IN against)

(NP (PRP his) (NN client)))))))))...

Original PropBank annotation:

Rel: forces

ARG1: a lawyer

ARG2: to become , in effect , a witness against his client

New Treebank annotation:

((SINV (“ “) (S-TPC-1 (NP-SBJ[**ARG0**] (DT This) (NN form))

(VP (VBZ[**REL**] forces)

(NP-2[**ARG1**] (DT a) (NN lawyer))

(S[**ARG2**] (NP-SBJ (-NONE- *PRO*-2))

(VP (TO to)

(VP (VB become)

(, ,)

(PP (IN in) (NP (NN effect)))

(, ,)

(NP-PRD (NP (DT a) (NN witness))

(PP (IN against)

(NP (PRP his) (NN client)))))))))...

New PropBank annotation:

Rel: forces

ARG1: a lawyer

ARG2: *PRO*-2 to become , in effect , a witness against his client

7.2.2 Small Clauses versus Secondary Predicates

Treebank allows more verbs to take sentential complements than PropBank does; therefore Treebank and PropBank have a significant amount of mismatches involving sentential complements, including infinitival clauses and small clauses (Babko-Malaya et al., 2006). After a collaborate effort made by both Treebank and PropBank annotators, the new Treebank redraws the boundary between verbs which take small clauses and those which take secondary predicates.

Basically, all the "label" verbs and some of the other verbs originally taking small clauses now take secondary predicates: 1) label verbs: including *appoint, call, code-name, designate, dub, elect, entitle, headline, label, mark, name, nickname, quote, rank, rate, rename, subtitle, tag, term, title, vote*; 2) other verbs: *drive, keep, leave, render, turn. Verbs which take small clauses are: believe, consider, declare, deem, fancy, fear, find, get, have, hold, judge, make, presume, proclaim, pronounce, prove, regard, render, report, rule, rumor, see, think, want, wish.* (Taylor, 2006)

Here is an example annotation affected by this change: The original PropBank had to label two sub-constituents of an S node (a sister node of the predicate, label) as ARG1 and ARG2, which generated a mismatch between Treebank and PropBank. With the new Treebank annotation, PropBank can now annotate two sister nodes of the predicate as ARG1 and ARG2.

[2] *Italian magistrates labeled his death a suicide .*

Original Treebank annotation:

```
( (S
    (NP-SBJ (JJ Italian) (NNS magistrates) )
    (VP (VBD[REL] labeled)
      (S
```

(NP-SBJ[**ARG1**] (PRP$ his) (NN death))

(NP-PRD[**ARG2**] (DT a) (NN suicide)))) (. .)))

Old PropBank annotation:

Rel: labeled

ARG1: NP-SBJ his death

ARG2: NP-PRD a suicide

New Treebank annotation:

((S (NP-SBJ (JJ Italian) (NNS magistrates))

(VP (VBD[REL] labeled)

(NP-1[ARG1] (PRP$ his) (NN death))

(S-CLR[ARG2] (NP-SBJ (-NONE- *PRO*-1))

(NP-PRD (DT a) (NN suicide)))) (. .)))

New PropBank annotation:

Rel: labeled

ARG1: NP-1 his death

ARG2: S-CLR a suicide

7.2.3 S-conjunction and PRN

Treebank standardized the position of modifiers of a conjoined S to the following (Taylor, 2006) so that the mismatches described in Section 6.4.4 can be eliminated:

(S (S-MODIFER)

(S ...)

(CC and)

(S ...))

The PRN (Parentheticals) tag in the original Treebank is used for various occasions, including marking sentential interpolations, anything enclosed in parentheses or set off by dashes, or phrases introduced by adverbs such as *namely, especially,*

particularly, notably, primarily, mostly, etc. (Bies et al., 1995) The new Treebank guidelines strictly constrain PRN usage to sentential interpolations (Taylor, 2006), and therefore all other PRNs are changed to relevant constituent labels. This change reduces some confusion PropBank annotators have encountered. Below is an example of a PRN tag that is changed to a PP-TMP tag which corresponds better with the PropBank ARGM-ADV label.

[3] *...finding reasons to say no , at least for now*

Original Treebank annotation:

```
(VP (VBG finding)
  (NP (NNS reasons)
    (S
      (NP-SBJ (-NONE- *) )
      (VP (TO to)
     (VP (VB say)
      (NP (RB no) )
      (, ,)
      (PRN[ARGM-ADV]
        (ADVP (IN at) (JJS least) )
        (PP-TMP (IN for) (RB now) ))))))))
```

New Treebank annotation:

```
(VP (VBG finding)
  (NP (NNS reasons)
    (S (NP-SBJ (-NONE- *PRO*))
      (VP (TO to)
        (VP (VB say)
          (FRAG (UH no))
          (, ,)
          (PP-TMP[ARGM-ADV] (ADVP (IN at) (JJS least))
```

(IN for)

(ADVP (RB now)))))))))

7.3 PropBank Changes

PropBank has made two major changes: 1) revision of frame files of certain verbs; and 2) policy changes regarding different types of chains. The verbs for which frame files are modified are those which are involved in the mismatches categorized as "sentential complements" (Section 6.4.2). Those verbs need to be examined on a verb-by-verb basis, and PropBank and Treebank have made concurrent efforts to make those verbs behave consistently. Relevant Treebank changes are described above and some verbs require revision of their frame files in PropBank.

7.3.1 PropBank Chains

PropBank annotates arguments of verbs, including all the entities linked to the arguments because of syntactic movement or semantic co-reference. However, in the original PropBank the annotation files do not specify different types of chains. This causes redundancy and inconsistency between PropBank and Treebank, because Treebank has already co-indexed all the syntactic chains and also empty categories are not co-indexed because the antecedents cannot be linked as a syntactic chain (Babko-Malaya et al., 2006).

In order to reconcile Treebank and PropBank, PropBank revised its annotation policy to annotate chains as a three stage process. First of all, PropBank annotates only empty categories as arguments and does not attempt to reconstruct syntactic chains. Syntactic chains are automatically reconstructed by the co-indexing provided in Treebank as a second stage. At the last stage, PropBank annotates semantic links manually which include co-reference, LINK-PCR, and indirect semantic chains, LINK-SLC. An indirect semantic chain means that the linked entities are not

semantically identical; however, the entities may have a certain kind of relationship such as "semantic type." For example, this type of link is commonly seen in cases of relative clauses and verbs of saying.

In Section 6.4.5, there is an example of verbs of saying. The original PropBank would concatenate several discontinuous constituents as a complete argument of the verb of saying; in the new PropBank, the argument is the maximal S node and an indirect semantic link will be specified. Below describes an example of a relative clause taken from (Babko-Malaya et al., 2006). At the first stage, PropBank only annotates the empty categories as arguments, and the syntactic chains are then recovered according to Treebank co-indexing, and at the third stage the NP answers is linked to the relative pronoun (which) as a semantic type link which is an indirect semantic link.

[4] Answers that we'd like to have

Treebank annotation:

(NP (NP answers) (SBAR (WHNP-6 which)
(S (NP-SBJ-3 we)
(VP 'd
(VP like
(S (NP-SBJ *-3)
(VP to
(VP have
(NP *T*-6)))))))))

Original PropBank annotation:

Rel: have

ARG1: [NP *T*-6] → which → answers

ARG0: [NP-SBJ *-3] → we

New PropBank annotation (first stage):

Rel: have

ARG1: [*T*-6]

ARG0: [NP-SBJ *-3]

New PropBank annotation (second stage, done automatically):

ARG1: [*T*-6] → which

ARG0: [NP-SBJ *-3] → we

7.4 Implementation of Synchronizing Penn Treebank and PropBank

As described below, the process of making the changes to PropBank to synchronize Treebank and PropBank is semi-automatic, and contains three phases: 1) Argument Location Realignment; 2) PropBank chain annotation; and 3) manual adjudication.

7.4.1 Argument Location Realignment

PropBank annotates arguments on top of Treebank, so an argument is labeled with its location in a Treebank tree. Since Treebank changes involve removing and adding traces, many argument locations are shifted by one index or more; therefore before making any change to PropBank annotation, PropBank annotation should be realigned to new locations.

The realignment is mostly done automatically. For every sentence that is changed in Treebank, a script examines all the PropBank propositions of that sentence. If an argument is shifted within reasonable range and still has the same span and constituent label, the script considers it a correct new argument. If the condition of same span and same constituent label cannot be satisfied, the script loosens the condition to be fulfilling one of the two and flags the new argument for further adjudication.

At the end the script generates four files: 1) a file contains PropBank lines which remain the same because there are no Treebank changes; 2) a file contains

new PropBank lines in which argument locations are changed because of Treebank changes; 3) a file contains arguments of which the new location cannot be determined because no realignment criteria can be met, for example, a trace is removed in the new Treebank; 4) a file contains lines of which verbs cannot be found in the new Treebank because the verb is changed to another part of speech.

7.4.2 PropBank Chains Annotation

The new PropBank annotates chains as a three-stage process, a script therefore is written to accommodate the new policy. Basically, the script first examines all the chains in the original PropBank; if the chain completely matches the chain specified in Treebank, then it's a syntactic chain, so only the trace is left as the new argument. If a PropBank chain is partially matched to a Treebank chain, then this chain can either be a direct semantic chain (co-reference) or an indirect semantic chain (semantic type).

If a semantic chain could be detected by straightforward criteria, the script changes the PropBank annotation correspondingly. For example, relative clauses and verbs of saying have regular patterns for the script to indicate an indirect semantic chain. However, other than very simple co-reference chains, most direct semantic chains have to be adjudicated manually.

7.4.3 Manual Adjudication

In addition to quality control of the automatic corrections made by the scripts on Treebank/PropBank realignment and PropBank chains annotation, manual adjudication has to be done on verbs for which frame files are modified. These verbs are usually the source of mismatches from the category of sentential complements. Please refer to 7.2.2 for a complete list.

7.5 Assessment on Synchronized Data

Comparing the performance of SRL taggers trained on the original data and the merged data can be a good way to evaluate the quality of the synchronized data. Since most of the Treebank and PropBank changes involve traces, we first evaluated the synchronized data by using an SRL tagger which would assign a semantic role label to traces and a syntactic parser which generates trace information. Focusing on the arguments linked by traces , we also compare the results with those of another SRL tagger using a MaxEnt parser which does not generate trace information. As far as SRL is concerned, giving a correct argument label to a trace-associated argument is more important than giving a correct label to a trace. A trace-associated argument is an argument which is linked by a trace, and the link can be a syntactic one or a semantic one. A trace-associated argument is one which has content, while a trace is usually an empty constituent.

In this section, we first report statistics on the distribution of PropBank changes due to Treebank changes. We also include the stats of PropBank arguments which are traces and trace-associated arguments. We then describe in detail the implementation of creating an SRL tagger which labels traces, and compare the results between training and testing on old data and merged data. We also report the results comparing two SRL taggers using different syntactic parsers on trace-associated arguments and the SRL performance on different type of mismatches.

7.5.1 Data Distribution

PropBank Changes due to Treebank Changes

There are 34,162 propositions in the selected 300k of PropBank annotation. 20,694 propositions are changed due to Treebank changes. Most of the changes are due to shifting the location of arguments because new traces are inserted, old traces are dropped, or some other empty categories are added. Among the 20,694 propositions,

13,150 propositions are changed due to new Treebank annotation on various types of verbs. Table 7.1 shows the stats. The numbers shown in Table 7.1 is a close approximation, because some verbs fall into two categories. For example, "*declare*" is an ECM verb and it is also a verb which takes small clauses.

Table 7.1: Stats of PropBank changes due to Treebank changes on different types of verbs.

Type of Verb	**Number of changes**
Raising Verbs	3,204
Subject Control Verbs	2,827
Ditransitive/Object Control Verbs	1,042
ECM Verbs	3,482
Verbs which take Secondary Predicates	686
Verbs which take Small Clauses	1,919

Stats of PropBank Arguments in Sec 23

Table 7.2 shows the stats of PropBank arguments in Sec 23 of the old 300k data and the merged 300k data. About 30% of the arguments are trace related arguments in both the old data and the merged data.

Table 7.2: Stats of PropBank arguments in Sec23. A trace-associated argument is one which has content, while trace is usually an empty constituent.

	All Arguments	Traces	Trace-Associated Arguments
Old 300k	10,373	1,542	1,523
New 300k	10,099	1,449	1,439

7.5.2 Implementation of a Trace-labeling SRL Tagger

To have an SRL tagger which can assign a semantic role label to traces, we had to modify the pruning component. The original pruning component does not consider any trace or empty node as an argument candidate. The modified pruning component includes traces and empty nodes which are sister nodes of the predicate or sister nodes

of the predicate's ancestor nodes. It also includes traces and empty constituents if they are the only child node of the sister nodes of the predicate or the predicate's ancestor nodes. If the trace is indexed, the pruning component will include all the co-indexed nodes no matter where the nodes are located in the tree.

We basically follow the original feature set to extract features for traces and empty nodes; however, we discard the head word feature or any combination feature which includes the head word.

For a syntactic parser which generates trace information, we used the one described in (Gabbard et al., 2006). The parser is based on Bikel's implementation with an empty category restorer which puts the traces back into the parse trees.

We trained two SRL systems and two syntactic parsers on Sec 02-21 of the selected original 300k data and synchronized 300k data. We then tested our systems on Sec 23 of the original and synchronized 300k data.[1]

7.5.3 Data Analysis and Parser Performance

The new PropBank adds two new argument labels, which are LINK-PCR and LINK-SLC. LINK-PCR indicates a direct semantic link and LINK-SLC indicates an indirect semantic link, i.e. a semantic type annotation. In order to more suitably compare the SRL results of the two systems, these two argument labels are linked back to a numbered argument or an ARGM and are represented in a "trace-chain" form according to the original PropBank guidelines.

In either the original or the synchronized 300k data, traces occupy slightly over 16% of the arguments. Table 7.3 describes the percentage of arguments and trace arguments preserved after the pruning component is run on the parse trees of Sec 23 generated by the syntactic parsers. This table is a good indication of the parser performance because the task of the pruning component is to generate argument

[1] Although the parser is trained on the selected 300k data, due to implementation complications, the empty category restorer could not be retrained and the one we used was trained on the Sec 02-21 of Penn Treebank 2 data.

candidates and the criterion to select a candidate is solely based on its relative location to the predicate in the parse tree.

Table 7.3: This table indicates the percentage of arguments and trace arguments preserved after the pruning component is run on the parse trees of Sec 23 generated by the syntactic parsers. About 16% of the arguments are traces.

Original 300k (Sec 23)		Merged 300k (Sec23)	
Argument Reserved	80.50%	Argument Reserved	80.60%
Trace Argument Reserved	73.40%	Trace Argument Reserved	71.16%

7.5.4 SRL Performance

Trace-labeling SRL system performance on old and merged data

Table 7.4 is the SRL performance with gold parses on Sec 23 of the original and synchronized 300k data, and Table 7.5 shows the SRL performance with automatic parses.

The general SRL performance trained and tested on the new synchronized data is significantly better than that trained and tested on the original data. Furthermore the recall number increases from 65.71 to 67.56. Both of these results are encouraging.

The motivation to synchronize Treebank and PropBank was because these two corpora had different opinions on which constituents should be arguments; therefore the arguments in PropBank which cause mismatches were not positioned at the correct locations in the Treebank trees. An increase of overall performance and especially an enhancement on recall, signal that synchronizing Treebank and PropBank reduces the noise in the training data and more arguments are attached at the right places.

The improvement of the SRL performance with automatic parses is not as obvious as that with gold parses. A possible reason is that the empty category restorer was trained on the 1M words of Penn Treebank2, and due to implementation complexity,

Table 7.4: SRL performance with gold parses on Sec 23 of the original and synchronized 300k data.

	Precision	**Recall**	**F1**
Original 300k	91.45	65.71	76.47
Merged 300k	90.86	67.56	77.50

Table 7.5: SRL performance with automatic parses on Sec 23 of the original and synchronized 300k data.

	Precision	**Recall**	**F1**
Original 300k	77.93	51.09	61.72
Merged 300k	77.35	52.12	62.28

we could not retrain it with only the Sec 02-21 of the 300k data. Therefore the empty category restorer might tend to generate traces according to the style of the old Treebank.

Comparison of two SRL taggers on trace-related arguments

In Chapter 5, we demonstrate that the MaxEnt parser we used is more robust than the Collins-style parser; therefore although the MaxEnt parser does not generate trace information, we still used the parser to combine with our base SRL tagger to evaluate the old 300k data and merged 300k data, focusing on the SRL performance on trace-associated arguments.

Table 7.6 and Table 7.7 show the stats of how many trace related arguments the SRL tagger retrieves when the SRL tagger uses gold parses (GOLD), parses generated by Bikel's parser with an empty category restorer (Bikel), and parses generated by the MaxEnt parser (ME) on the old 300k and the new merged 300k data.

Table 7.6: Number of trace related arguments retrieved. Data: old 300k.

	Traces (1,542 arguments)	**Trace-Associated Arguments** (1,523 arguments)
Gold	1,204	675
Bikel	920	490
ME		506

Table 7.7: Number of trace related arguments retrieved. Data: new 300k.

	Traces (1,449 arguments)	**Trace-Associated Arguments** (1,439 arguments)
Gold	1,067	719
Bikel	783	520
ME		558

The ME parser does not provide trace information, so the SRL tagger cannot assign any trace an argument label. However, on trace-associated arguments, somewhat to our surprise, the ME parser performs better than Bikel's parser. The objective of labeling a trace correctly is to get the trace-associated arguments by using the link. In order to determine whether the trace information provided by Bikel's parser will help us obtain more trace-associated arguments, we ran some post-processing code so that if a trace is labeled as an argument then all the co-indexed constituents are labeled with the same argument label. However, the overall results and the recall on the trace-associated arguments are not affected significantly. It seems that the SRL tagger managed to give trace-associated arguments correct labels solely by the lexical features. Given that the ME parser performs better on trace-associated arguments and the training of an empty category restorer is computationally expensive, for the moment the best approach to SRL is still using the ME parser.

SRL performance on arguments associated with different verb types

Table 7.8 and Table 7.9 show the SRL performance on arguments associated with different verb types. We report the SRL performance on gold parses and parses generated by the MaxEnt parser. Traces are not included in the evaluation, so the overall performance reported on gold parses is different than that reported in Table 7.4. Table 7.8 shows the SRL performance on the olf 300k data and Table 7.9 on new merged 300k data. Table 7.9 also reports the difference of F1 on the old data and the new data.

When using the gold parses, only the performance on the say verbs increases.

Table 7.8: SRL performance on arguments associated with different verb types. Data: Old 300k. P: Precision; R: Recall

	Gold Parses			**MaxEnt Parser**		
	P	**R**	**F1**	**P**	**R**	**F1**
All	90.74	82.56	86.46	71.69	63.26	67.21
Raising Verbs	92.32	89.38	90.82	71.65	68.43	70.00
Subject Control	89.82	85.31	87.51	75.66	70.81	73.15
Ditransitive	89.76	83.96	86.76	74.65	66.25	70.20
ECM	94.70	90.10	92.34	74.11	72.19	73.14
Secondary Predicate	86.07	72.92	78.95	66.67	52.78	58.91
Small Clauses	91.45	84.67	87.93	73.72	68.72	71.13
Say Verbs	98.18	95.50	96.82	73.98	74.08	74.03

Table 7.9: SRL performance on arguments associated with different verb types. Data: New 300k. P: Precision; R: Recall

	Gold Parses				**MaxEnt Parser**			
	P	**R**	**F1**	**Diff F1**	**P**	**R**	**F1**	**Diff F1**
All	90.26	81.62	85.72	-0.74	72.18	62.99	67.27	+0.06
Raising Verbs	91.88	88.82	90.32	-0.50	73.28	68.80	70.97	+0.97
Subject Control	89.34	83.88	86.53	-0.98	76.22	70.27	73.13	-0.02
Ditransitive	88.36	81.47	84.78	-1.98	72.79	64.21	68.23	-1.97
ECM	93.75	88.97	91.30	-1.04	75.07	71.96	73.48	+0.34
Secondary Predicate	84.47	70.08	76.60	-2.35	64.19	52.27	57.62	-1.29
Small Clauses	91.39	83.40	87.21	-0.72	76.54	69.07	72.62	+1.49
Say Verbs	98.45	95.62	97.01	+0.19	76.12	74.66	75.38	+1.35

When the SRL tagger takes input from the MaxEnt parser, the performance improves in general but degrades on ditransitive verbs and verbs which take secondary predicates. The verbs fall in these two categories are also the most problematic verbs when using gold parses. These results are all complicated by the amount of new training data being significantly less than normally used to train parsers.

7.5.5 Discussion

We evaluated the quality of the synchronized Treebank/PropBank data by comparing the SRL performance trained on the original and the synchronized dataset. The preliminary results using parses supplemented with trace information showed improvement on recall and overall performance. These results might indicate that synchronizing Treebank and PropBank reduces the noise in the training data because more arguments are attached to the right locations in the parse trees. However, when we compare the results with those generated by another SRL tagger using a MaxEnt parser which does not generate trace information, we find that the SRL performance on trace-associated arguments is better with the MaxEnt parser. With a further analysis on SRL performance on arguments associated with different verb types, we realized the synchronized data do not necessarily improve the performance on every verb category. Furthermore, when we do not include traces in the evaluation, the overall performance is not improved either.

The observations indicate that the noisy data problem might still exist and there are three possible reasons: 1) There are still cases of gapping and phrasal verbs (including light verbs) which have not been addressed and may be having the greatest negative impact on SRL performance; 2) The lexical features have been doing a better job than we anticipated of providing the correct labels in the face of noisy data, and do not benefit from cleaner data; 3) These results are still quite preliminary based on a small amount of data and more careful investigation is required, in particular of the interactions between detecting traces and SRL.

7.6 Conclusion

In Chapter 5, we discovered that one serious problem current SRL system are facing is noisy training data. This problem is demonstrated by many mismatches found in the Penn Treebank and the PropBank. In order to attack this problem and facilitate the study of the syntactic/semantic interface, a project was launched to synchronize Treebank and PropBank.

In Chapter 6, we described different types of mismatches. In this chapter, we demonstrated the changes Treebank and PropBank have to make in order to accommodate the mismatches. Selected files (300k) are synchronized, and the preliminary assessment on the merged data by comparing SRL performance on the old 300k and new 300k data indicate that the noisy data problem might still exist because the synchronization is not yet complete.

Chapter 8

The Impact of Verb Senses on SRL

The task of SRL is to locate and classify the arguments of the predicate in question. Because a lemma with different senses often has different predicate argument structures, one would expect word sense information to have a significant impact on the performance of SRL systems. For example, according to the PropBank frame files, excluding the phrasal verb cases the verb call has 6 framesets, and each of them demonstrates different predicate argument structures, as illustrated in the following table.

Frameset	Argument Roles and Example Sentences
call.01	Arg0: caller
	Arg1: item being labelled
	Arg2: attribute of arg1
	Mary called John an idiot
call.02	Arg0:caller
	Arg1:thing being summoned
	Arg2:benefactive
	The bells called the faithful to evensong
call.03	Arg0:demander
	Arg1:thing being demanded
	Gore called for a recount
etc.	

Although the relevance of verb sense information should be obvious , most of

the current SRL systems use only verb lemmas and not verb senses in their feature sets (Hacioglu et al., 2003; Moschitti, 2004; Pradhan et al., 2004; Punyakanok et al., 2004; Yi and Palmer, 2004; Pradhan et al., 2005b; Punyakanok et al., 2005; Toutanova et al., 2005). Difficulty of implementation is thought to be a major factor. Given that current SRL systems often use a uni-directional pipelined framework, it is challenging to suitably incorporate a WSD component in the current framework so that the SRL system can utilize word sense information. Furthermore, without a perfect WSD component, the error propagation effect is likely to consume the benefit an SRL system can obtain from the word sense information.

In this chapter, we evaluate the contribution of word sense information (for only verbs) to SRL system performance and conduct experiments to investigate a possible solution to the implementation issue. We start with small scale experiments which focus on a small subset of verbs selected based on their degree of polysemy and the number of instances in the dataset. We also make sure that the verbs we select have a relatively balanced distribution among different senses, i.e., relatively high entropy. We then train a syntactic parser which uses verb sense information as a feature and extend our experiments to cover polysemous verbs that have at least 50 instances in the training data. Finally we perform data and error analysis to assess the correspondences between verb sense information and SRL performance.

This chapter is structured as follows: Section 8.1 describes the pioneer experiments which focus on a small subset of verbs in the training data. We will explain the criteria we use to select the polysemous verbs and demonstrate the challenge of incorporating a WSD component in an SRL system. We then propose an approach using a parser that is trained with verb sense information in order for an SRL system to benefit from verb sense information. Encouraged by the performance improvement shown by the pioneer experiments, we extend the experiments to a larger scale. We perform two sets of experiments using numbers of instances as the cut-off value: One set of experiments uses verbs that have at least 100 instances in the training

data and the other set uses verbs that have at least 50 instances. In Section 8.3, we present different analyses in order to assess the correspondences between SRL performance and characteristics of verb senses. We conclude our discovery and suggest future direction to utilize verb sense information in Section 8.4.

8.1 Pioneer Experiments

For our pioneer experiments in investigating the impact of verb sense information on SRL performance, we manually selected ten verbs which are very polysemous and have enough instances for training. The results of the early experiments show that SRL performance improves when we put verb sense into the feature set; however, when we switch our experiments to use automatic parses, the SRL system trained with verb sense information performed significantly worse than the original system. The results inspired us to have a parser be trained on verb sense information. The combination of the new parser and the new SRL system significantly outperforms the old system on the 10 selected verbs, which encourages us to extend our experiments to a larger scale.

8.1.1 Ten Selected Verbs

In order to select verbs which are representative for experiments which investigate the impact of verb sense, we started by selecting verbs which have the most framesets in the PropBank frame files and filter them out by the number of instances in the training data and the data distribution among different senses. We therefore selected ten verbs which have more than 7 PropBank Framesets (senses) and have at least 300 instances in the training data. In the process, we discarded some verbs which do not have balanced distribution among verb senses. For example, we discarded the verb *fall*, because its data distribution is too skewed. The verb fall has 794 instances in the training data but out of the 794 instances, 787 instances are tagged *fall.01*.

The ten verbs we selected for the pioneer experiments are: *call, hold, put, get, make, set, close, follow, lose, look.*

8.1.2 SRL System Modifications

In order to have an SRL system which utilizes verb sense information, we modify the feature set by replacing the verb lemma with PropBank frameset id. For example, for the verb *call*, in the original SRL system we will only use its lemma, *call*, in the feature set; in the new SRL system, we will use as features *call.01*, *call.02*, etc. The features which are modified are summarized in Table 8.1.

Table 8.1: modified features in order to accommodate frameset id information

SRL System Component	**Features Modified (lemma is replaced with frameset id)**
Argument Identification	1. lemma + phrasal type of the constituent
	2. lemma + head word of the constituent
	3. lemma + distance between the predicate and the constituent
	4. lemma + phrasal type of the parent node of the constituent if the parent node is a PP
Argument Classification	1. lemma & All of the above

8.1.3 Single Verb Experiment: the Verb '*Call*'

We first did a single verb experiment, and selected the verb *call* as the target verb. We discarded the phrasal verb senses of *call* such as *call up, call in, call on* etc to make the experiment more focused. In the final dataset, we have 6 senses of *call*: *call.01, call.02, call.03, call.07, call.10*, and *call.11*. We randomly picked 90% of instances from each sense as training data and 10% as test data.

Two SRL systems are trained: one uses the original feature set; the other uses the feature set in which the lemma is replaced with the frameset id. The models are trained on gold parses and gold verb senses. When testing, we ran the systems on

gold parses and automatic parses but we assumed that we could always obtain gold verb senses.

Table 8.2: SRL Performance of the Single Verb Experiment: the Verb *call*

Results on gold parses and gold verb senses			
System	**Precision**	**Recall**	**F1**
SRL - lemma	96.19	66.01	78.29
SRL - frameset id	95.45	68.63	79.85
Results on automatic parses and gold verb senses			
SRL - lemma	74.44	43.79	55.14
SRL - frameset	68.04	43.14	52.8

When gold parses are used, the verb sense information improves the overall performance; however, when we switched to automatic parses, the new SRL system performs significantly worse than the original system even though it had the gold verb sense information. The results suggest that putting a WSD component in the SRL pipeline will not improve performance and it might be even less helpful to overall SRL performance since most current WSD systems also rely on automatic syntactic parsers for lexical features. We need a parser which generates appropriate parses for the same verb with different senses. This inspired us to train a syntactic parser which takes verb sense information into consideration.

8.1.4 Putting Verb Sense in Parses

We trained an automatic parser which uses verb sense information as an additional feature based on the hypothesis that this parser will generate parses which better accord with the predicate argument structures associated with verb senses; therefore it will improve SRL system performance.

After further examining the frame files for the ten selected verbs, we decided to take out *call* and *make* because they have somewhat conflicting frameset definitions. We added *turn* and *keep* in the verb pool. The parser was trained on WSJ corpus Sec 02-21 supplied with the verb sense information of the ten selected verbs. When

parsing the test sentences, we assumed that the gold standard verb sense tags are given to the parser.

Table 8.1.4 shows the overall SRL performance of the original system and the new system on the argument labels of the 10 selected verbs.

System	**Precision**	**Recall**	**F1**
SRL - lemma	87.24	78.78	82.79
SRL - frameset	88.88	80.72	84.60

The results are promising - both precision and recall improved which resulted in a significantly better F1 score. This suggests that a parser which learns verb sense information might generate parses which demonstrate appropriate predicate argument structure for the verb in question. We therefore continued to perform our experiments on more polysemous verbs.

8.2 Polysemous Verbs and SRL

We wanted broader scale coverage for our experiments ; therefore we analyzed the data distribution of all the annotated verbs in the PropBank. Out of the 3,324 verbs that have a frame file in the PropBank, 548 verbs are polysemous, which means they have at least two senses and also have at least 50 instances. The stats, based on the number of instances and the number of senses, are summarized in the Table 8.3.

We used the number of instances as a cut-off index and performed two sets of experiments on verbs which have more than 50 instances and on verbs which have more than 100 instances. Table 8.4 shows the overall SRL performance on verbs which have more than 50 instances; Table 8.5 demonstrates the overall SRL performance on verbs which have more than 100 instances. SRL-original means that the SRL system is trained on the verb lemma only and uses parses generated from the original syntactic parser which was not supplied with verb sense information. SRL-frameset means that the SRL system is trained on the frameset id rather than

Table 8.3: The Stats of the polysemous verbs in PropBank.

# instances	# verbs	# senses	# verbs
less than or equal to 50	359	2	336
51 100	74	3	101
101 150	31	4	49
151 200	20	5	25
more than 200	64	6	7
		7	10
		8	3
		9	4
		10	3
		more than 10	10

the verb lemma and uses parses generated by the parser which is trained with verb sense information.

Table 8.4: SRL Performance on Verbs which have more than 50 instances.

System	**Precision**	**Recall**	**F1-Score**
SRL-original	75.14	66.89	70.77
SRL-frameset	79.49	63.26	70.45

Table 8.5: SRL Performance on Verbs which have more than 100 instances.

System	**Precision**	**Recall**	**F1-Score**
SRL-original	74.72	67.67	71.02
SRL-frameset	80.08	64.39	71.38

The overall SRL performance didn't improve over verbs which have more than 50 instances and it is slightly but not significantly better over arguments of verbs that have more than 100 instances in the training data. What is noteworthy is that the SRL-frameset systems have significantly better precision numbers; with the degradation on recall numbers bringing down the overall results. In the next section, we investigate the correspondence between SRL performance and the verbs along with their distribution over their verb senses on a verb-by-verb basis.

8.3 Correspondences between Verb Sense and SRL Performance

We evaluated the SRL performance on a verb-by-verb basis for all the 115 verbs which have more than 100 instances in the training data. Except for one verb, note, for which the SRL-frameset system didn't generate any arguments, 61 verbs have a better F-1 score while the remaining 53 verbs have a worse F-1 score. Among the 61 verbs which have a better F-1 score, 32 verbs have both better precision and recall; 27 verbs have better precision but worse recall; 2 verbs have worse precision but better recall. Among the 53 verbs which have a worse F-1 score, 20 verbs have both worse precision and recall; 33 verbs have better precision but worse recall. 92 out of 115 verbs have better precision; only 34 of 115 verbs have better recall. We summarized the observations in Table 8.3.

F-1 score improves: 61 verbs		
# verbs	**Precision**	**Recall**
32	+ (improves)	+
27	+	- (degrades)
2	-	+
F-1 score degrades: 53 verbs		
20	-	-
33	+	-

We first eye-balled the characteristics of verbs according to their performance in an attempt to find a correspondence between the two; we especially focused on the verbs for which precision and recall move in the same directions, i.e. both of them improve or both of them degrade. However, as shown in Table 8.6 and Table 8.7, there is no obvious relationship to be seen. We therefore calculated the entropy for each verb but did not find a correspondence between the entropy and the performance either. Finally, we trained a simple classifier to predict whether the performance will improve using a feature set which includes the number of instances, the number of senses, and the entropy. Although the training accuracy is high, the

classifier has poor prediction power. It seems that in order to predict whether verb sense information will help boost SRL performance, a thorough lexical analysis is required for every verb in question.

8.4 Conclusion

A verb with different senses often has different predicate argument structures; therefore verb sense information should play a significant role on semantic role labeling task. However, due to implementation difficulties, current SRL systems often only use a verb lemma instead of verb senses of a verb in their feature sets. In this chapter, we investigated the impact of verb senses on SRL performance. We discovered that in order to benefit from verb sense information, a word sense disambiguation component should be closely integrated into the SRL system pipeline rather than being a standalone component.

Our early experiments showed that the SRL system performance improved significantly given that the SRL system and the syntactic parser are both trained on verb sense information on selected verbs which have more than 7 senses and more than 300 instances in the training data. However, when we expanded our experiments to cover more polysemous verbs, the improvement of overall SRL performance became less significant. The analysis of the characteristics of a verb (including number of verb senses, number of instances in the training data, and the entropy) did not indicate any strong correspondence between these characterisitics and SRL performance.

Although it seems that a thorough lexical analysis on a verb-by-verb basis is required to determine whether it is helpful to use verb sense information to enhance the overall SRL system performance, a noteworthy fact is that about 80% of the verbs experienced improvement with respect to the precision metric. This means we could find a benefit from this approach by giving more weight to the argument labels it predicts when combining it with other SRL systems.

Table 8.6: The distribution and characteristics of verbs of which the SRL-frameset system has both higher precision and recall.

Verb	**# instances**	**# senses**	**Entropy**
Add	719	4	1.3355186661752934
Argue	161	2	0.133697960635451
Back	135	5	0.9890517936173526
Buy	946	3	0.11079373814121543
Climb	119	2	0.3227569588973983
cut	298	10	1.557668634882871
decline	452	2	0.9925159706595619
do	792	2	0.025429870053185876
drive	107	2	0.8683587934693187
drop	347	7	1.0965881214790107
end	553	3	0.79630549320698
enter	115	2	0.7553754125614288
extend	110	2	0.5499109046334101
gain	232	3	1.036549250305852
get	1087	15	2.468676676850146
have	3033	4	0.77878962103729
head	149	3	1.3208188086907655
jump	162	5	0.9737052047125692
lose	345	7	1.3042485567255513
pay	757	7	0.4809829309315353
play	186	7	1.5881221701356159
reach	306	3	0.9927766478497978
relate	138	2	0.06192738891155925
return	153	5	1.2505294427112563
sell	1278	4	0.10125261769614297
spend	190	5	0.9375459625938841
succeed	161	3	0.9062911315715206
talk	133	2	0.19476878174480028
total	159	2	0.05503793508320906
vote	123	3	0.5246328621381392
watch	105	2	0.2334897104477296
win	232	2	0.1500665680177682

Table 8.7: The distribution and characteristics of verbs of which the SRL-frameset system has both lower precision and recall.

Verb	**# instances**	**# senses**	**Entropy**
aim	107	3	0.7485071088846285
allow	316	3	0.3024350550109494
bring	276	5	0.6153860682508624
change	270	2	0.3943002379274748
claim	136	2	0.1528492278398859
comment	159	2	0.20159510785384202
cover	175	4	0.8565954081806364
hurt	145	2	0.21639693245126468
like	111	2	0.9929042710341394
live	129	4	0.7851121435117333
name	253	2	0.0928712084188745
prepare	115	2	0.9598636448150406
put	431	13	1.8502596579103958
see	661	4	0.111444172472922
send	190	5	1.0787006247521855
serve	155	3	0.2615066631884383
settle	144	4	1.2804584077234917
sign	133	5	1.409485174669811
wait	120	2	0.06952964699480783
yield	288	3	0.48838369395599446

Chapter 9

Can Semantic Role Generalize Across Genres?

PropBank has been widely used as training data for Semantic Role Labeling (SRL) and the performance of the SRL systems trained on the PropBank corpus seems to be reaching a ceiling. However, those systems do not port well to other genres. For instance, in Chapter 5, we demonstrated that the SRL system performance dropped significantly when tested on the Brown corpus. The 2005 CoNLL shared task has also addressed this issue by evaluating participating systems on a test set extracted from the Brown corpus, which is very different from the WSJ corpus that was used for training. The results suggest that there is much work to be done in order to improve system robustness.

One possible reason for this lack of robustness is that the machine learning models tend to overfit on idiosyncrasies of the style of the training text, especially since the training corpus, WSJ corpus, is highly specialized, and tends to use genre-specific word senses for many verbs.

Another possible reason is that current SRL systems have difficulty deciding which role label to assign to a given argument because PropBank role labels are defined on a verb-by-verb basis. This is less problematic for ARG0 and ARG1,

where a conscious effort was made to be consistent across verbs; but is a significant problem for ARG[2-5], which tend to have very verb-specific meanings. For example, the verb "make" uses ARG2 for the "Material" argument; but the verb "multiply" uses ARG2 for the "Extent" argument. This problem is exacerbated even further on novel genres, where SRL systems are more likely to encounter uses of arguments that were unseen in the training data.

Current efforts at mapping different lexical resources have shed light on this problem. As the mapping between PropBank and VerbNet has been made available, we have been able to transform the SRL data to make it more consistent, and to expand the size and variety of the training data. In particular, we can use the mapping to transform the verb-specific PropBank role labels into the more general thematic role labels that are used by VerbNet. Unlike the PropBank labels, the VerbNet labels are defined consistently across verbs and verb classes; and therefore it should be easier for statistical SRL systems to model them. Furthermore, since the VerbNet role labels are significantly less verb-specific than the PropBank roles, the SRL models should generalize better to novel verbs, and to novel usages of known verbs.

In order to test these ideas, we re-trained our Maximum Entropy SRL system on a transformed version of the PropBank data, where PropBank role labels were mapped to the corresponding VerbNet thematic role labels. Our hypothesis is that the mapping between VerbNet and PropBank creates more consistent training data; therefore it should improve the SRL system performance. In addition, because VerbNet thematic roles behave more consistently than PropBank argument labels across verbs, an SRL system trained on VerbNet thematic roles should be better able to generalize to new instances, making it more robust on genres other than WSJ-style articles.

In the following text, we elaborate the issues caused by the verb-specific PropBank roles, describe the mapping between PropBank and VerbNet, and report and

discuss the experiments using new set of semantic roles across genres and demonstrate that the new labels are easier to learn. The work described in this chapter is joint work accomplished with Edward Loper. Loper's contribution is the creation of the mapping between PropBank and VerbNet, which is described in Sec 9.2.

9.1 Overloaded Argument Labels in PropBank

The PropBank annotates the entire Penn Treebank with predicate argument structures, using semantic role labels for each verb argument. Because there is little consensus in the linguistic and NLP communities about what set of role labels are most appropriate, PropBank avoids the debate by using numbered arguments (ARG0, ARG1,...,ARG5) defined on a verb-by-verb basis. Therefore there is no guarantee that an argument label will be used consistently across different verbs.

In fact, except for ARG0 and ARG1, where an explicit effort was made when PropBank was created to use ARG0 for arguments that fulfill Dowty's criteria for "prototypical agent" and ARG1 for arguments that fulfill criteria for "prototypical agent," (Dowty, 1991), all other numbered arguments are highly overloaded. And even though ARG0 and ARG1 are more consistent across verbs than other numbered arguments there are still some inter-verb inconsistencies for ARG0 and ARG1.

Table 9.1 gives an example of the overloading of ARG2. The argument label ARG2 is used as a "Material" argument for the verb *make*, an "Extent" argument for the verb multiply, and a "Destination" argument for the verb **bring**.

Table 9.1: An overloaded ARG2 label

Verb	**Corresponding thematic role**	**Example sentence (ARG2 marked in brackets)**
make	Material	Those chips are made [of gallium arsenide].
multiply	Extent	...investors shouldn't just multiply the third quarter [by four]...
bring	Destination	She brought [them] shame.

The overloaded PropBank numbered arguments pose a challenge to SRL system robustness. Because the numbered arguments behave inconsistently across verbs, an SRL system trained on the PropBank labels will encounter difficulties in generalizing to novel text and novel verb behaviors.

9.2 Mapping between PropBank and VerbNet

A bi-directional mapping between PropBank and VerbNet has been created through a two-step process: 1) lexical mapping and 2) instance classification. The lexical mapping provides possible mapping options for a given verb including the verb class mapping and argument label mapping; the mapping candidates are then passed to an instance classifier. The instance classifier determines the most probable mapping for a given verb depending on lexical features extracted from the context. Currently about 75% of the word instances in PropBank are mapped to VerbNet. Work is underway to cover the 25% gap. More information about the mapping, and the details of its creation can be found in (Loper et al., 2007).

A preliminary analysis of the mapping demonstrated in (Yi et al., 2007) has confirmed our belief that PropBank roles ARG0 and ARG1 are relatively coherent, while ARG2-5 are much more overloaded. We include the analysis figure here as Figure 9.1. From the figure, we can see that ARG0 maps to agent-like roles over 94% of the time, and ARG1 maps to patient-like roles over 82% of the time, while ARG2-5 are mapped to various types of roles. From this figure, we can also see that ARG0 and ARG1 occupy about 90% of the sample size, while ARG[3-5] occupy only 0.5% of the sample size. This indicates that classifiers trained for ARG3-5 can suffer from a severe data sparseness problem.

Arg0 (45,579)	
Agent	85.4%
Experiencer	7.2%
Theme	2.1%
Cause	1.9%
Actor1	1.8%
Theme1	0.8%
Patient1	0.2%
Location	0.2%
Theme2	0.2%
Product	0.1%
Patient	0.0%
Attribute	0.0%

Arg1 (59,884)	
Theme	47.0%
Topic	23.0%
Patient	10.8%
Product	2.9%
Predicate	2.5%
Patient1	2.4%
Stimulus	2.0%
Experiencer	1.9%
Cause	1.8%
Destination	0.9%
Theme2	0.7%
Location	0.7%
Source	0.7%
Theme1	0.6%
Actor2	0.6%
Recipient	0.5%
Agent	0.4%
Attribute	0.2%
Asset	0.2%
Patient2	0.2%
Material	0.2%
Beneficiary	0.0%

Arg2 (11,077)	
Recipient	22.3%
Extent	14.7%
Predicate	13.4%
Destination	8.6%
Attribute	7.6%
Location	6.5%
Theme	5.5%
Patient2	5.3%
Source	5.2%
Topic	3.1%
Theme2	2.5%
Product	1.5%
Cause	1.2%
Material	0.8%
Instrument	0.6%
Beneficiary	0.5%
Experiencer	0.3%
Actor2	0.2%
Asset	0.0%
Theme1	0.0%

Arg3 (609)	
Asset	38.6%
Source	25.1%
Beneficiary	10.7%
Cause	9.7%
Predicate	9.0%
Location	2.0%
Material	1.8%
Theme1	1.6%
Theme	0.8%
Destination	0.3%
Instrument	0.3%

Arg4 (18)	
Beneficiary	61.1%
Product	33.3%
Location	5.6%

Arg5 (17)	
Location	100.0%

Figure 9.1: The frequency with which each PropBank numbered argument is mapped to each VerbNet thematic role in the mapped corpus. The numbers next to each PropBank argument reflects the number of occurrences of that numbered argument in the mapped corpus.

9.3 SRL Experiments on Linked Lexical Resources

In order to verify the feasibility of performing semantic role labeling with VerbNet thematic roles, we re-trained our existing SRL system, which originally used PropBank role labels, with a new label set that makes use of VerbNet thematic role information.

9.3.1 SRL Experiments on Mapped VerbNet Thematic Roles

Since PropBank arguments Arg0 and Arg1 are already quite coherent, we left them as-is in the new label set. We replaced Arg2-Arg5 by mapping them to their corresponding VerbNet thematic role. We found that mapping directly to individual role labels exacerbated the already significant sparse data problem, since the number of output tags was increased from 6 to 23. We therefore grouped the VerbNet thematic roles into five coherent groups of similar thematic roles, shown in Figure 9.2.[1] Our new tag set therefore included the following tags: **Arg0** (*agent*); **Arg1** (*patient*); **Group1** (*goal*); **Group2** (*extent*); **Group3** (*predicate/attrib*); **Group4** (*product*); and **Group5** (*instrument/cause*).

In order to evaluate the two different sets of argument labels, we trained two SRL systems: The Original system labels Arg0-5, ArgA and ArgM, and the Mapped system labels Arg0, Arg1, ArgA, ArgM and Group1-5. In particular, the role labels generated by the original system are verb-specific, while the role labels generated by the new system are less verb-dependent.

9.3.2 Results

For our training and testing, we used the portion of Penn Treebank II that is covered by the mapping, and where at least one of Arg2-5 is used. Training was performed using sections 02-21 of the Treebank (10,783 instances of argument); and testing was

[1] Karin Kipper assisted in creating the groupings.

Group 1	Group 2	Group 3	Group 4	Group 5
Recipient	Extent	Predicate	Patient2	Instrument
Destination	Asset	Attribute	Product	Cause
Location		Theme		Experiencer
Source		Theme1		Actor2
Material		Theme2		
Beneficiary		Topic		

Figure 9.2: Thematic Role Groupings for the experiments on linked lexical resources; and for Arg2 in the experiments on arguments with different verb independency.

System	**Precision**	**Recall**	**F1**
Original	90.65	85.43	87.97
Mapped	88.85	84.56	86.65

Table 9.2: Overall SRL System performance using the PropBank tag set ("Original") and the augmented tag set ("Mapped"). Test Set: WSJ corpus.

performed on section 23 (859 instances) and on the Brown corpus (1,107 instances) which is also PropBank-ed and VerbNet-mapped. Table 9.2 displays the SRL system performance on all the arguments of two SRL systems (Original and Mapped) for the WSJ corpus and Table 9.3 for Brown corpus. We also evaluated the two SRL systems only on the argument labels which are involved in the mapping, i.e. Arg2-5 for the Original system and Group1-5 for the Mapped system. The performance on the WSJ corpus and on the Brown corpus is shown respectively in Table 9.4 and Table 9.5.

9.3.3 Discussion

The results on WSJ corpus indicate that performance drops when we train on the new argument labels, especially on precision when we evaluate the systems on only

System	**Precision**	**Recall**	**F1**
Original	81.56	72.65	76.85
Mapped	82.72	70.91	76.36

Table 9.3: Overall SRL System performance using the PropBank tag set ("Original") and the augmented tag set ("Mapped"). Test Set: Brown corpus

System	Precision	Recall	F1
Original	97.60	83.67	90.10
Mapped	91.70	82.86	87.06

Table 9.4: SRL System performance evaluated on only Arg2-5 (Original) or Group1-5 (Mapped). Test Set: WSJ corpus.

System	Precision	Recall	F1
Original	71.21	63.17	66.95
Mapped	82.38	58.58	68.47

Table 9.5: SRL System performance evaluated on only Arg2-5 (Original) or Group1-5 (Mapped). Test Set: Brown corpus.

Arg2-5/Group1-5 (see Table 9.3). However, the results on Brown corpus show a very different trend. In general, the recall drops. This is probably due to the sparse data problem caused by an increase in the number of argument labels. But what is noteworthy is that the precision improves, especially when we evaluate the systems on only Arg2-5/Group1-5 (see Table 9.5). The improvement of precision not only compensates the loss of recall but brings out a significant better F1 score.

Our hypothesis is that the new set of semantic role labels are less verb-specific and therefore should be easier to learn and should improve SRL system robustness. The improvement on the Brown corpus echoes this hypothesis. However, this might be a premature conclusion because 1) we have very few mapped Arg3-5 instances (less than 1,000 instances); 2) The Original system and the Mapped System are performing slightly different tasks since they generate different argument labels, and therefore although their results suggest a trend, the results are not directly comparable.

In order to further verify our hypothesis, we redesigned our experiments by limiting the scope to mapped instances of Arg1 and Arg2. By doing this, we should be able to accomplish the following: 1) we can map new argument labels back to the original PropBank labels; therefore we can directly compare results; 2) We can validate our original hypothesis that the behavior of Arg1 is primarily verb-independent

Group 1	Group 2	Group 3	Group 4	Group 5
Theme	Source	Patient	Agent	Topic
Theme1	Location	Product	Actor2	
Theme2	Destination	Patient1	Experiencer	Group 6
Predicate	Recipient	Patient2	Cause	Asset
Stimulus	Beneficiary			
Attribute	Material			

Figure 9.3: Thematic Role Groupings for Arg1 in the experiments on arguments with different verb independency.

while Arg2 is more verb-specific.

9.4 SRL Experiments on Arguments with Different Verb Independency

We conducted two further sets of experiments: one to test the effect of the mapping on learning Arg2; and one to test the effect on learning Arg1. Since Arg2 is used in very verb-dependent ways, we expect that mapping it to VerbNet role labels will increase our performance. However, since a conscious effort was made to keep the meaning of Arg1 consistent across verbs, we expect that mapping it to VerbNet labels will provide less of an improvement.

Each experiment compares two SRL systems: one trained using the original PropBank role labels; the other trained with the argument role under consideration (Arg1 or Arg2) subdivided based on which VerbNet role label it maps to. In order to prevent the training data from these subdivided labels from becoming too sparse (which would impair system performance) we grouped similar thematic roles together. For Arg2, we used the same groupings as the previous experiment, shown in Figure 9.2. The argument role groupings we used for Arg1 are shown in Figure 9.3.

The training data for both experiments is the portion of Penn Treebank II (sections 02-21) that is covered by the mapping. We evaluated each experimental system using two test sets: section 23 of the Penn Treebank II, which represents the same

System	Precision	Recall	F1
Arg1-Original	89.24	77.32	82.85
Arg1-Mapped	90.00	76.35	82.61
Arg2-Original	73.04	57.44	64.31
Arg2-Mapped	84.11	60.55	70.41

Table 9.6: SRL System Performance on Arg1 Mapping and Arg2 Mapping, tested using the *WSJ corpus (section 23)*. This represents performance on the same genre as the training corpus.

System	Precision	Recall	F1
Arg1-Original	86.01	71.46	78.07
Arg1-Mapped	88.24	71.15	78.78
Arg2-Original	66.74	52.22	58.59
Arg2-Mapped	81.45	58.45	68.06

Table 9.7: SRL System Performance on Arg1 Mapping and Arg2 Mapping, tested using the *PropBank-ed Brown corpus*. This represents performance on a different genre from the training corpus.

genre as the training data; and the PropBank-ed portion of the Brown corpus, which represents a very different genre.

9.4.1 Results and Discussion

Table 9.6 describes the results of SRL overall performance tested on the WSJ corpus Section 23; Table 9.7 demonstrates the SRL overall system performance tested on the Brown corpus. Systems Arg1-Original and Arg2-Original are trained using the original PropBank labels, and show the baseline performance of our SRL system. Systems Arg1-Mapped and Arg2-Mapped are trained using PropBank labels augmented with VerbNet thematic role groups. In order to allow comparison between the system using the original PropBank labels and the systems that augmented those labels with VerbNet thematic role groups, system performance was evaluated based solely on the PropBank role label that was assigned.

We had hypothesized that with the use of thematic roles, we would be able to

create a more consistent training data set which would result in an improvement in system performance. In addition, the thematic roles would behave more consistently than the overloaded Args[2-5] across verbs, which should enhance robustness. However, since in practice we are also increasing the number of argument labels an SRL system needs to tag, the system might suffer from data sparseness. Our hope is that the enhancement gained from the mapping will outweigh the loss due to data sparseness.

From Table 9.6 and Table 9.7 we see the F1 scores of Arg1-Original and Arg1-Mapped are statistically indifferent both on the WSJ corpus and the Brown corpus. These results confirm the observation that Arg1 in the PropBank behaves fairly verb-independently so that the VerbNet mapping does not provide much benefit. The increase of precision due to a more coherent training data set is compensated for by the loss of recall due to data sparseness.

The results of the Arg2 experiments tell a different story. Both precision and recall are improved significantly, which demonstrates that the Arg2 label in the PropBank is quite overloaded. The Arg2 mapping improves the overall results (F1) on the WSJ by 6% and on the Brown corpus by almost 10%. As a more diverse corpus, the Brown corpus provides many more opportunities for generalizing to new usages. Our new SRL system handles these cases more robustly, demonstrating the consistency and usefulness of the thematic role categories.

9.5 Improved Argument Distinction via Mapping

The ARG2-Mapped system generalizes well both on the WSJ corpus and the Brown corpus. In order to explore the improved robustness brought by the mapping, we extracted and observed the 1,539 instances to which the system ARG2-Mapped assigned the correct semantic role label, but which the system ARG2-Original failed to predict. From the confusion matrix depicted in Table 9.8, we discover the following:

Confusion Matrix		ARG2-Original		
		ARG1	ARG2	ARGM
ARG2-Mapped	ARG0	53	50	-
	ARG1	-	716	-
	ARG2	1	-	2
	ARG3	-	1	-
	ARGM	1	482	-
233 ARG2-Mapped arguments are not labeled by ARG2-Original				

Table 9.8: Confusion matrix on the 1,539 instances which ARG2-Mapped tags correctly and ARG2-Original fails to predict.

The mapping makes ARG2 more clearly defined, and as a result there is a better distinction between ARG2 and other argument labels: Among the 1,539 instances that ARG2-Original didn't tag correctly, 233 instances are not assigned an argument label, and 1,252 instances ARG2-Original confuse the ARG2 label with another argument label: the system ARG2-Original assigned the ARG2 label to 50 ARG0's, 716 ARG1's, 1 ARG3 and 482 ARGM's, and assigned other argument labels to 3 ARG2's.

9.6 Distinction of Core Arguments and Adjunct Arguments

The PropBank Args2-5 are overloaded and many of them in the PropBank behave more like adjuncts and should in fact be tagged as ArgM. With the mapping between PropBank and VerbNet, we therefore are able to re-tag those arguments as ArgM according to their mapped VerbNet thematic roles. Our hypothesis is that by doing this, we can better define the distinction between core arguments and adjunct-like arguments, as a result, the training data will be less noisy and the resulting SRL system can perform more robustly.

We design our experiments as follows: We used the portion of Penn Treebank II

System	Precision	Recall	F1
Old	93.60	84.89	89.03
New	93.29	85.37	89.15

Table 9.9: SRL System Performance on the original role set and on the new role set which contains ArgMs transformed from some Arg2-5. Test Set: WSJ corpus

System	Precision	Recall	F1
Old	81.28	72.11	76.42
New	81.26	73.56	77.21

Table 9.10: SRL System Performance on the original role set and on the new role set which contains ArgMs transformed from some Arg2-5. Test Set: Brown corpus

that is covered by the mapping and where at least one of Arg2-5 is used. For every Arg2-5, we re-tag it as ArgM if it is mapped to one of the following VerbNet thematic roles: Asset, Source, Location, Cause, Material, Destination, and Instrument. About 10% of the numbered argument, Arg2-5, are re-tagged as ArgM. We then trained two systems: one on the original data set and the other on the new data set, and tested them on both the WSJ corpus and the Brown corpus.

Table 9.9 shows the overall SRL performance of these two systems (Old and New) on the WSJ corpus and Table 9.10 on the Brown corpus.

The overall results suggest a slight improvement but it is not significant. We suspect that this is due to an insufficient amount of training data and therefore not many arguments were re-tagged. However, the improved recall on both WSJ and Brown corpora suggests that a clearer distinction between core arguments and adjunct-like arguments might help the SRL system capture arguments better.

9.7 Conclusion

We have described a mapping from the annotated PropBank corpus to VerbNet verb classes with associated thematic role labels. We hypothesized that these labels would be more verb-independent and less overloaded than the PropBank Arg2-5, and would

therefore provide more consistent training instances which would generalize better to new genres. Our experiments confirm this hypothesis, with a 6% performance improvement on the WSJ and a 10% performance improvement on the Brown corpus for Arg2.

In future research we will work on expanding the coverage of VerbNet and therefore enlarge the overlap between the two lexical resources, PropBank and VerbNet.

Chapter 10

Conclusion

In the thesis, we explored the problem of semantic role labeling (SRL). SRL is an important and necessary intermediate step for successful natural language processing applications, such as text summarization, question answering, and machine translation. Many researchers have investigated applying machine learning to a corpus specifically annotated with this task in mind, PropBank, since 2000. For two years, the CoNLL workshop has made this problem the shared task. Current SRL system performance on the Wall Street Journal corpus seems to be reaching a ceiling. However, in order to make an SRL system applicable, many unsolved issues remain.

The task of SRL faces two major challenges: 1) finding a suitable syntactic parser; 2) achieving adequate system robustness. A syntactic parser is the upstream component of an SRL system pipeline; therefore it naturally acts as a bottleneck for the task. While current SRL systems seem to have reached a performance ceiling on the WSJ corpus, they do not port well to other genres. Achieving system robustness is an important goal for the task of semantic role labeling. An SRL system without robustness cannot serve as a useful semantic processing tool.

In this thesis, we investigate ways to train a better syntactic parser and increase SRL system robustness. We demonstrate that parse trees augmented by semantic role markups can serve as suitable training data for training a parser for an SRL

system. Furthermore, we show that by resolving the discrepancies between the Penn Treebank and PropBank, we can create a cleaner training corpus both for training the parsers and the SRL systems. For system robustness, we propose that it is easier to learn a new set of semantic roles transformed from the original argument roles based on the mapping between VerbNet and PropBank. The new roles are less verb-dependent compared to the original PropBank roles. As a result, the SRL system trained on the new roles achieves significantly better robustness than the original system.

Based on the observations we obtained from the various experiments described in the previous chapters, we recommend taking the following approach to training a robust SRL system: 1) Use a MaxEnt parser if the target test text will be different from the training text; 2) Train with verb senses in addition to verb lemmas for highly frequent and highly polysemous verbs; and 3) Use the more verb-independent thematic roles to replace the highly verb-dependent PropBank argument roles Arg2-5, to gain system robustness and more meaningful semantic roles.

Future improvements in SRL can be gained by continued resource development. Currently only 300k words of the Treebank 2 and PropBank are synchronized and the overlap between VerbNet and PropBank has not yet reached 100%. Finishing the synchronization of the TreeBank and PropBank and completing the verb overlap between VerbNet and PropBank will provide a cleaner, and larger training corpus. This will facilitate the learning of parsing and SRL as a joint reference problem. Furthermore, we will be able to provide empirical evidence as to the question of, given certain circumstances, which set of semantic roles is more appropriate?

Bibliography

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM.

James Allen. 1995. *Natural Language Understanding*. Benjamin/Cummings, CA, second edition.

Olga Babko-Malaya, Ann Bies, Ann Taylor, Szu-ting Yi, Martha Palmer, Mitch Marcus, Seth Kulick, and Libin Shen. 2006. Issues in synchronizing the english treebank and propbank. In *Frontiers in Linguistically Annotated Corpora, A Merged Workshop with 7th International Workshop on Linguistically Interpreted Corpora (LINC-2006) and Frontiers in Corpus Annotation III, Coling/ACL*.

Olga Babko-Malaya. 2005. Propbank annotation guidelines. http://www.cis.upenn.edu/ mpalmer/project_pages/PBguidelines.pdf.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING/ACL*, pages 86–90, Montreal, Canada.

Mark Baker. 1985. *Incorporation: A Theory of Grammatical Function Changing*. Ph.D. thesis, MIT, Cambridge.

A. Belletti and L. Rizzi. 1988. Psych-verbs and theta-theory. *Natural Language and Linguistic Theory*, 6:291–352.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for treebank ii style. Technical report, Penn Treebank Project, University of Pennsylvania, Department of Computer and Information Science.

Daniel Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

C.J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Xavier Carreras and Lluís Márquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

Xavier Carreras and Lluís Márquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

Chih-Chung Chang and Chih-Jen Lin. 2001. Libsvm: a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

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

John Chen and K. Vijay-Shanker. 2000. Automated extractions of the tags from the penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 65–76.

Nancy A. Chinchor. 1998. Overview of muc-7/met-2. In *Proceedings of the Message Understanding Conference MUC-7*.

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

A. Copestake, B Jones, A. Sanfilippo, H. Rodriguez, P. Vossen, S. Montemagni, and E. Marinai. 1992. Multilingual lexical representation. Technical report, University of Cambridge Computer Laboratory.

A. Copestake. 1992. The acquilex lkb: representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, pages 88–96, Trento, Italy.

Hoa Trang Dang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI.

H. T. Dang, K. Kipper, M. Palmer, and J. Rosenzweig. 1998. Investigating regular sense extensions based on intersective levin classes. In *Proceedings of Coling-ACL 98*, Montreal.

H. T. Dang, K. Kipper, and M. Palmer. 2000. Integrating compositional semantics into a verb lexicon. In *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING-2000)*, Saarbrucken, Germany.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum-likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc. Ser. B (methodological)*, 39:1–38.

A. M. Di Sciullo and E. Williams. 1987. On the definition of word. *Linguistic Inquiry Monograph*, 14.

Bonnie J. Dorr, M. Antonia Martí, and Irene Castellón. 1997. Spanish eurowordnet and lcs-based interlingual mt. In *Proceedings of the Workshop on Interlinguas in MT, MT Summit*, pages 19–32, San Diego, CA.

Bonnie J. Dorr, David Farwell, Rebecca Green, Nizar Habash, Stephen Helmreich, Eduard Hovy, Lori Levin, Keith J. Miller, Teruko Mitamura, Owen Rambow, Florence Reeder, and Advaith Siddharthan. 2004. Interlingual annotation of parallel text corpora: A new framework for annotation and evaluation. *Journal of Natural Language Engineering*. submission.

Bonnie J. Dorr. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation*, 12(4):271–322.

David R. Dowty. 1989. On the semantic content of the notion thematic role. In *Properties, Types, and Meanings*, volume 2, pages 69–130, Kluwer.

David R. Dowty. 1991. Thematic proto-roles and argument selection. In *Languages*, volume 67, pages 574–619.

Myroslava O. Dzikovska, Mary D. Swift, and James F. Allen. 2003. Constructing custom semantic representations from a generic lexicon. In *Proceedings of 5th International Workshop on Computational Semantics*, Tilburg, The Netherlands.

David Farwell, Stephen Helmreich, Bonnie J. Dorr, Nizar Habash, Florence Reeder, Keith Miller, Lori Levin, Teruko Mitamura, Eduard Hovy, Owen Rambow, and Advaith Siddharthan. 2004. Interlingual annotation of multilingual text corpora. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Workshop on Frontiers in Corpus Annotation*, pages 55–62, Boston, MA.

C. Fillmore. 1968. The case for case. In *Universals in Linguistic Theory*.

Michael Fleischman and Eduard H. Hovy. 2003. A maximum entropy approach to framenet tagging. In *HLT-NAACL*.

Atsushi Fujita, Kentaro Furihata, Kentaro Inui, Yuji Matsumoto, and Koichi Takeuchi. 2004. Paraphrasing of japanese light-verb constructions based on lexical conceptual structure. In *Proceedings of the 2nd ACL Workshop on Multiword Expressions: Integrating Processing (MWE-2004)*, pages 9–16, Barcelona.

Ryan Gabbard, Seth Kulick, and Mitch Marcus. 2006. Fully parsing the penn treebank. In *HLT-NAACL*.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 57–64, Sapporo, Japan.

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

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

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, pages 239–246, Philadelphia, PA.

Daniel Gildea. 2002. Probabilistic models of verb-argument structure. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 308–314, Taipei.

J. Grimshaw. 1990. *Argument Structure*. MIT Press, Cambridge.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

J. Gruber. 1967. *Studies in Lexical Relations*. Ph.D. thesis, MIT, North Holland.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2003. Shallow semantic parsing using support vector machines. Technical report, The Center for Spoken Language Research at the University of Colorado (CSLR).

Edward Hovy, Andrew Philpot, Judith L. Klavans, Ulrich Germann, and Peter T. Davis. 2003. Extending metadata definitions by automatically extracting and organizing glossary definitions. In *Proceedings of the National Conference on Digital Government Research*, Boston, MA.

Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425.

R. Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press, Cambridge.

R. Jackendoff. 1983. *Semantics and Cognition*. MIT Press, Cambridge, Mass.

R. Jackendoff. 1987. The status of thematic relations in linguistic theory. *Linguistic Inquiry*, 18:369–411.

R. Jackendoff. 1990. *Semantic Structures*. MIT Press, Cambridge, Mass.

R. Jackendoff. 1996. *The Architecture of the Language Faculty*. MIT Press, Cambridge, Mass.

Christopher Johnson and Charles J Fillmore. 2000. The framenet tagset for frame-semantic and syntactic coding of predicate-argument structure. In *Proceedings of the ANLP-NAACL 2000*, Seattle, WA.

Karin Kipper, Hoa Trang Dang, William Schuler, and Martha Palmer. 2000. Building a class based verb lexicon using tags. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms*.

Ulrich H.-G Krebel. 1999. Pairwise classification and support vector machines. *Advances in kernel methods: support vector learning*.

Alex Lascarides and Ann Copestake. 1998. Pragmatics and word meaning. *Journal of Linguistics*, 34(2):387–414.

B. Levin and M. Rappaport. 1986. The formation of adjective passives. *Linguistic Inquiry*, 17:623–662.

B. Levin. 1993. *English Verb Classes and Alternations, a Preliminary Investigation*. The University of Chicago Press.

Edward Loper, Szu-ting Yi, and Martha Palmer. 2007. Empirical evidence for useful semantic role categories. In *Proceedings of the International Workshop on Computational Linguistics*.

Alec P. Marantz. 1984. *On the Nature of Grammatical Relations*. MIT Press.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of english : The penn treebank. *Computational Linguistics*, 19(2):313–330.

Lluís Márquez, Pere Comas, Jesús Giménez, and Neus Catalá. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL-2005*.

Diana McCarthy. 2000. Using semantic preferences to identify verbal participation in role switching alternations. In *Proceedings of the 1st NAACL*, pages 256–263, Seattle, Washington.

Paola Merlo and Suzanne Stevenson. 2001. Automatic verb classification based on statistical distribution of argument structure. *Computational Linguistics*, 27(3), September.

G. Miller. 1990. Wordnet: an online lexical database. *International Journal of Lexicography*, 3(4).

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *Proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004)*, Barcelona, Spain.

Kamal Nigam, Andrew Kachites McCallurn, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134.

Taisuke Nishigauchi. 1984. Control and the thematic domain. In *Lg*, pages 397–414.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005a. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Martha Palmer, Nianwen Xue, Olga Babko-Malaya, Jinying Chen, and Benjamin Snyder. 2005b. A parallel proposition bank ii for chinese and english. In *Proceedings of the 2005 Workshop on Frontiers in Corpus Annotation II*.

Martha Palmer. 1990. *Semantic Processing for Limited Domains*. Cambridge University Press.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st ACL*, pages 183–190, Columbus, Ohio.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*, Boston, MA.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. Martin, and D Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*.

J. Pustejovsky. 1991. The generative lexicon. *Computational Linguistics*, 17(4):409–441.

Malka Rappaport and Beth Levin. 1988. What to do with theta-roles. In *Wilkins*, pages 7–37.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

P. Resnik and M. Diab. 2000. Measuring verb similarity. In *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society (CogSci 2000)*, pages 399–404.

Henk Van Riemsdijk and Edwin Williams. 1986. *Introduction to the Theory of Grammar*. MIT Press, Cambridge.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th Annual Meeting of the ACL*, College Park, MD.

Bozena Rozwadowska. 1988. Thematic restricitons on derived nominals. In *Wilkins*, pages 147–66.

A. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, and A. Oh. 1999. Creating natural dialogs in the carnegie mellon communicator system. In *Proceedings of Eurospeech*, volume 4, pages 1531–1534.

A. Sanfilippo and V. Poznanski. 1992. The acquisition of lexical knowledge from combined machine-readable dictionary sources. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, pages 80–88, Trento, Italy.

Antonio Sanfilippo, Nicoletta Calzolari, Sophia Ananiadou, Rob Gaiazauskas, Patrick Saint-Dizier, Piek Vossen, Antonietta Alonge, Nuria Bel, Kalina Bontcheva, and Pierrette Bouillon. 1999. Eagles le3-4244: Preliminary recommendations on lexical semantic encoding final report.

Eric Tjong Kim Sang, Sander Canisius, Antal van den Bosch, and Toine Bogers. 2005. Applying spelling error correction techniques for improving semantic role labeling. In *Proceedings of CoNLL-2005*.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Sabine Schulte im Walde. 2000. Clustering verbs semantically according to their alternation behavior. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 747–753, Saarbrucken, Germany.

S. Seneff and J. Polifroni. 2000. Dialogue management in the mercury flight reservation system. In *ANLP Conversational Systems Workshop*.

Libin Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

David Stallard. 2000. Talk'n'travel: A conversational system for air travel planning. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP'00)*, pages 68–75.

Mihai Surdeanu and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL-2005*.

Mihai Surdeanu, Sanda Harabagiu, JohnWilliams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, Japan.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

Mary Swift. 2005. Towards automatic verb acquisition from verbnet for spoken dialog processing. In Katrin Erk, Alissa Melinger, and Sabine Schulte im Walde, editors, *Proceedings of Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, Saarbruecken, Germany.

Leonard Talmy. 1985. Figure and ground as thematic roles. In *Proceedings of Annual Meeting of the Linguistic Society of America*, Seattle.

Ann Taylor. 2006. Treebank 2a guidelines. http://www-users.york.ac.uk/ lang22/TB2a_Guidelines.htm.

Kristina Toutanova, Aria Haghighi, and Christopher D. 2005. Joint learning improves semantic role labeling. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of me and svm via integer linear programming. In *Proceedings of CoNLL-2005*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Piek Vossen. 1997. Eurowordnet: a multilingual database for information retrieval. In *Proceedings of the Delos workshop on Cross-language Information Retrieval*.

W. Ward and S. Issar. 1994. Recent improvements in the cmu spoken language understanding system. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 213–216.

W. Ward and B. Pellom. 1999. The cu communicator system. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone, Colorado.

E. Williams. 1981. Argument structure and morphology. *Linguistic Review*, 1:81–114.

Nianwen Xue and Martha Palmer. 2003. Annotating propositions in the penn chinese treebank. In *Proceedings of the Second Sighan Workshop*, Sapporo, Japan.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Szu-ting Yi and Martha Palmer. 2004. Pushing the boundaries of semantic role labeling with svm. In *Proceedings of the International Conference on Natural Language Processing*.

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

Szu-ting Yi, Edward Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of NAACL-HLT*.

M. L. Zubizarreta. 1987. *Levels of Representation in the Lexicon and in the Syntax*. Foris, Dordrecht.