2008-07-14

Semantic Role Labeling with Analogical Modeling

Warren C. Casbeer
Brigham Young University - Provo

SEMANTIC ROLE LABELING WITH ANALOGICAL MODELING

by

Warren Christopher Casbeer

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Arts

Department of Linguistics and English Language

Brigham Young University

July 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Warren Christopher Casbeer

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Deryle Lonsdale, Chair
Date	David Eddington
Date	Royal Skousen

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Warren Christopher Casbeer in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Deryle Lonsdale
Chair, Graduate Committee

Accepted for the Department

William Eggington
Department Chair

Accepted for the College

Joseph Parry
Associate Dean

ABSTRACT

SEMANTIC ROLE LABELING WITH ANALOGICAL MODELING

Warren Christopher Casbeer

Department of Linguistics and English Language

Master of Arts

Semantic role labeling has become a popular natural language processing task in recent years. A number of conferences have addressed this task for the English language and many different approaches have been applied to the task. In particular, some have used a memory-based learning approach. This thesis further develops the memory-based learning approach to semantic role labeling through the use of analogical modeling of language. Data for this task were taken from a previous conference (CoNLL-2005) so that a direct comparison could be made with other algorithms that attempted to solve this task. It will be shown here that the current approach is able to closely compare to other memory-based learning systems on the same task. Future work is also addressed.

ACKNOWLEDGMENTS

I am grateful to my thesis committee chair, Dr. Deryle Lonsdale, for his knowledge and willingness to direct me through this process.

David Eddington and Royal Skousen are also appreciated greatly for their work on my committee.

The Department of Linguistics and English Language provided useful materials for being able to complete the publication of this thesis.

Finally, I would like to thank my family, especially my wife Marta T. Casbeer, for their support during this process.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Semantic information describes the relationships that exist between lexical and syntactic constituents and their predicates. The identification of these relationships is important for answering questions such as 'Who?', 'What?', and 'Where?' Improvement of current natural language processing tasks such as text summarization, information extraction and question answering requires deeper semantic information than was previously needed.

For this reason, semantic role labeling (SRL) has become a popular task for conferences and workshops in recent years. The basic SRL task consists of identifying arguments (or semantic roles) of given target verbs in sentences.

In recent years the Conference on Computational Natural Language Learning (CoNLL) has focused on SRL for the English language. Participants used sentences from the Penn Treebank that had certain target verbs identified. Information such as lexical words, part of speech tags and clausal information were provided to participants. Along with the actual sentences, this information was used in the task of identifying arguments of the given target verbs.

Most of the approaches towards SRL at these conferences have used machine learning methods. In this thesis I will describe a system that has been developed for the SRL task and applied to the actual data from the 2005 CoNLL task. The system is based on analogical modeling (AM) (Skousen, 1989).

Chapter 2 discusses natural language tasks as well as machine learning in general before providing background information on SRL. Conferences based on this task and approaches toward the task are described. Analogical modeling, the paradigm chosen for the current system's approach to the task, is also introduced. Chapter 3 details the methodology of the current system, including a description of computational tools used. Chapter 4 describes the overall architecture of the current system. Chapter 5 gives results, discusses them, and compares them with those from systems

that participated in the previous CoNLL SRL tasks. Conclusions and future work, including the possibility of extending this type of system for SRL in other languages, are addressed in chapter 6.

Chapter 2

Background and Literature Review

The SRL task has become popular recently due to improvements in natural language processing (NLP). This chapter briefly describes NLP as well as how machine learning has helped to establish the ability to complete more complex NLP tasks such as SRL. Included in this will be a brief discussion of memory-based learning (specifically using TiMBL[1] as this machine learning technique is directly comparable to the present approach to SRL. Additionally, the SRL task as accomplished at CoNLL in 2004 and 2005 will be described. This will be compared briefly with the task as given at another competition, Senseval-3. Approaches that have been used for the task will be explained, followed by a closer look at approaches most similar to the present one. Finally, a description of the computational paradigm (analogical modeling, or AM) behind the present approach will be provided.

2.1 Natural Language Processing and Machine Learning

NLP is concerned with trying to develop automatic generation and understanding of natural human languages. It belongs to the fields of artificial intelligence and computational linguistics. Systems may either try to take information and generate a snippet of human language or they may seek to take portions of human language and understand their content. Example tasks from the NLP field include information extraction, word sense disambiguation, automatic summarization, machine translation, text to speech, and natural language generation.

In order for any of these tasks to be completed successfully, computers need access to significant amounts of knowledge as well as the ability to change the form of knowledge for understanding. Syntactic, semantic, and lexical information is required, for example, for segmentation of input (in either text or speech format) and disambiguation of syntactic ambiguity (where more than one parse is available). Additionally, other information is sometimes required that can solve pragmatic differences

[1] available at http://ilk.uvt.nl/timbl

such as speech acts. Computers must also be learn to be able to resolve imperfect input (i.e. grammatical errors, speech impediments).

NLP tasks can become intricately difficult, as will be demonstrated. The ability to carry out such tasks has been aided by the recent development of machine learning techniques. Machine learning is a part of artificial intelligence, and it consists of the development of computational algorithms and techniques that help computers to learn. In this process, computers take large input data sets and inductively learn rules or patterns from them.

Memory-based learning (MBL) is a popular approach to machine learning (Lin and Vitter, 1994). The whole premise behind MBL consists in directly using examples (for analogical purposes) instead of rule application, where rules could have been extracted from those examples. In this way, new instances (of unseen data) can be classified according to their degree of similarity with a stored set of known instances (seen data).

A recent book provides a comprehensive review of MBL for purposes of problem solving in language technology (Daelemans and van den Bosch, 2005). It compares this and other machine learning techniques for modelling language, and the TiMBL system is discussed.

MBL processing is a viable alternative to other machine learning algorithms in a variety of linguistic NLP tasks including clause identification (Tjong Kim Sang, 2001), named entity recognition (Hendrickx and van den Bosch, 2003), and grammatical relation finding (Buchholz, 2002). The last of these is important to the present discussion as it relates to SRL tasks. The MBL algorithms for language modelling, including TiMBL, are somewhat comparable to the algorithm (AM) used for the present system.

2.2 Semantic Role Labeling Task

Carreras and Màrquez (2004, 2005) describe the basic SRL task at CoNLL[2]. The task focused on identifying propositions (constituents of the verb that play a role) including the semantic arguments of target verbs. Arguments involve both standard (e.g. agent, goal, patient) as well as adjunctive (e.g. manner, temporal, locative) constituents. The CoNLL (both 2004 and 2005) organizers provided all participants with data sets that were used to develop computational systems to recognize and label arguments of the given target verbs in a test corpus of sentences from PropBank. Data will be described more in depth at a later point.

[2]http://www.lsi.upc.edu/∼srlconll/spec.html

Senseval-3[3] also included an SRL task (Litkowski, 2004) for English using FrameNet[4] (Fillmore et al., 2001), which is a database similar to PropBank. FrameNet is a set of sentences that give target verbs along with a frame that identifies arguments of these target verbs. The version of FrameNet (1.1) used at Senseval-3 included 132,968 sentences. Most of these were taken from the British National Corpus. Only 8000 sentences (chosen randomly) were used for the Senseval-3 SRL task.

The task was based on the work done by Gildea and Jurafsky (2002). It was similar to the CoNLL tasks in that it provided sentences as well as target verbs, and asked participants to recognize arguments (or frame elements in FrameNet terminology). However, additionally a frame type was provided in the Senseval-3 task. A frame example[5] is shown below:

```
<frame name="Cause_fluidic_motion">
  <instance lexunit="pump.v" luID="9973" sentID="256263">
    <sentence>However, its task is made much more difficult by the fact that derogations granted to
    the Welsh water authority allow it to pump raw sewage into both those rivers.</sentence>
    <target start="125" end="128">pump</target>
    <frame_elements>
      <frame_element name="Agent" start="119" end="120">it</frame_element>
      <frame_element name="Fluid" start="130" end="139">raw sewage</frame_element>
      <frame_element name="Goal" start="141" end="162">into both those rivers</frame_element>
    </frame_elements>
  </instance>
</frame>
```

SRL tasks have recently been extended to other languages. An example is the Arabic SRL task at SemEval-2007[6].

2.3 Approaches to SRL

Much work has been accomplished in SRL for the English language. A number of systems, based on various theories, have used for a basis the features provided by Gildea and Jurafsky (2002). These included phrase type, parse tree path, position, voice, head word, governing category, and subcategorization.

The rest of this section surveys the basic features and different approaches used in SRL. This will include a discussion about features used by systems similar to the system preapared in this thesis.

[3]http://www.senseval.org/senseval3
[4]http://framenet.icsi.berkeley.edu/
[5]http://www.clres.com/SensSemRoles.html
[6]http://nlp.cs.swarthmore.edu/semeval/tasks/task18/description.shtml

2.3.1 Basis for SRL Features

Feature development for SRL began with Gildea and Jurafsky (2002) and Gildea and Palmer (2002), with the first system being tested on FrameNet and the second on PropBank. Both these systems used the same base set of features. First, the *phrase type* is the syntactic type of the consituent (phrase) labeled as a predicate argument. The *path* attempts to identify the syntactic relationship that the argument under question has with the rest of the sentence. It contains the part-of-speech tag of the target verb, and successively links (upward or downward within a parse) syntactic categories moving toward the marked argument. The *position* indicates the argument's location in relation to the predicate (before or after). *Voice* marks the target verb as active or passive, based on identification patterns for passive structure. These patterns generally require a passive auxiliary as well as a past participle VBN.

The *head word* of the evaluated phrase is the head word (i.e. the word whose part-of-speech tag determines phrasal type) based on rules in Collins (1999). For example, in prepositional phrases (PP), the preposition would be the head word. The *governing category* applies only to identified NPs. A value of S is given to NPs that are subjects of verbs (dominated by a sentence phrase), while a value of VP is given for verbal objects (dominated by a verbal phrase). The *subcategorization* defines the arguments (in phrasal form) that a target verb expects, essentially expanding the target word's parent node within the parse tree. This information was provided in a schema such as VP $\rightarrow$ VBD PP NP. Here, the verbal phrase VP requires a verb (VBD is a past tense verb), a prepositional phrase PP, and a noun phrase NP.

2.3.2 Extending the Basic Feature Set

The basic set of features has provided a base for many other researchers working on the SRL task using a variety of system types. Gildea and Hockenmaier (2003) used Combinatory Categorial Grammar (CCG, Steedman (2000)) to test the basic feature set. Features were obtained from a CCG grammar instead of the Treebank. This type of grammar seeks to have direct mappings between syntactic structure and underlying semantic meaning. In this work, the authors mapped PropBank role labels to a CCGBank of arguments. They noted that this system performed better on core (numbered) arguments than the system by Gildea and Palmer (2002).

Chen and Rambow (2003) likewise built a system based on Tree Adjoining Grammar (TAG), and used the base set of features for testing. The TAG had to be extracted from PropBank in this process. TAG allowed them to extend the features

into deeper syntactic levels. They used a set of features called supertag path, supertag, Srol, Ssubcat, Drole, Dsubcat, and semsubcat.

Surdeanu et al. (2003) used a decision tree (DT) algorithm in an information extraction technique, taking advantage of basic SRL features for predicate argument identification. An additional predicate word feature had two forms. The first was the target verb in sentential context, with case and morphological information preserved, while the second was the lemma (i.e. verb in lower case, infinitive form).

A second set of features was also developed based on their own observations. First, most of the predicate arguments are prepositional attachments (PP) or relative clauses (SBAR). Due to this, the head word feature is not often the most informative word of a phrase. For example, they provide the phrase 'in last June'. The preposition 'in' is the head word of this PP, but the word 'June' seems to be more informative. A new set of heuristics (similar to head word rules) was made to identify these more informative words (called *content words*). For an SBAR phrase such as 'that occurred yesterday', the left-most sentence (S*) clause is selected as the content word. Here, the content word 'occurred' differs from the head word 'that'. Secondly, they noticed that neither the head word nor the content word features were used often. Part-of-speech (POS) tags for the content and head words were then given as features. The third observation relates to named entities (NE); these tags categorize elements (such as persons, locations, organizations, etc.) in text. For example, the phrase 'Big Board' could be given an ORGANIZATION NE tag. A new feature provides the NE class of the content word, defined to be the class of the NE that includes the content word. They additionally provided Boolean NE features (e.g. neOrg, neLoc, nePers) that were helpful in identifying adjunct arguments such as locations. Finally, for phrasal verbs (e.g. 'put up') the verb particles are part of the predicate and not the argument. Two features were added to measure frequencies in which verbs were followed by certain prepositions or particles.

Pradhan et al. (2003) similarly implemented some new features for SRL, while using support vector machines (SVM) in their approach to the task. New features included verb clusters, named entity, partial path, and the POS tag for the head word. Some of these coincide with features used by Surdeanu et al. (2003).

Thompson et al. (2003) created a generative model for the SRL task using FrameNet. This type differs from the discriminative model of Gildea and Jurafsky (2002).

For an overview of all features, Carreras and Màrquez (2004, 2005) categorized those used by systems participating in the CoNLL SRL tasks. First, local information

(e.g. lexical items, POS tags) provides context for constituents considered. Secondly, other features (e.g. phrase type, content words) provide a look at the internal structure of an argument. Next, characteristics of the target verb are considered; features include the actual predicate as well as voice of the target verb. Another feature type considered relationships existing between the target verb and the potential arguments (e.g. path, position).

2.3.3 Memory-based algorithms for SRL

Other recent approaches to the SRL task include memory-based learning algorithms as described by Kouchnir (2004), van den Bosch et al. (2004), Ahn et al. (2004) and Tjong Kim Sang et al. (2005). All of these machine learning systems were based on the Tilburg Memory-Based Learner (TiMBL) (Daelemans et al., 2003). With the exception of Ahn et al. (2004), all of these systems were participants in the CoNLL SRL tasks. The system described by Ahn et al. (2004) took part in the Automatic Labeling of Semantic Roles task at Senseval-3.

These MBL systems performed the SRL task by means of classification, determining whether individual constituents could be considered arguments of the target verb. Since these are the SRL systems that most closely parallel the approach used in this thesis, each one will be reviewed in detail below. A close look at features used by these systems should be important in determining what features should be used in our approach here, since they are similar.

Kouchnir's (2004) system was divided into two modules, the first of which worked to recognize possible arguments while the second labeled the identified arguments. Many of the same features were used for both modules, but each will be treated individually here.

The recognition module included features for the head word and POS tag of the focus element, clausal information (beginning/intermediate/end of a clause), chunk type, directionality (position feature in Gildea and Jurafsky (2002)), distance, voice (passive if the predicate was tagged as a past participle VBN and was preceded by a form of 'to be'), context, and adjacency. Chunk type and adjacency were novel additions to the feature possibilities, and are explained further here. Kouchnir used a set of twelve identified chunk types such as NP, VP, or PP. The feature of adjacency related to the placement of the focus element in relation to the verb chunk (adjacent to the verb chunk or not); it was even possible for the focus element to be within the verb chunk. Additionally, context features were used in which the head word, POS tag, chunk type, and adjacency of preceding and following chunks were taken into

account. Adjacency was found to be quite a useful feature in the SRL task, and is now regularly considered.

The labeling module used by Kouchnir (2004) included the word, POS tag, and chunk sequence, as well as clausal information, length of argument, directionality, adjacency, and the voice of the target verb. These were all used commonly between the two modules. An additional feature used for the labeling module was the PropBank roleset of the target verb. In this feature Kouchnir decided to consider only the first sense roleset, as 86% of target verbs used the first sense and many times the first two senses contained the same arguments.

Kouchnir (2004) also offers an interesting suggestion that could be taken into consideration in the present case.

> "As all argument boundaries, except for those within the target verb chunks, coincide with base chunk boundaries, the data is processed by words only within the target verb chunk, and by chunks otherwise" (p. 118).

The system developed by van den Bosch et al. (2004) used a number of features in common with other systems. They used distance, passive main verb, the role pattern, and word form features. Distance was measured in several ways including by words, chunks, NP chunks, and VP chunks.

Other new features called preceding preposition, attenuated words (Eisner, 1996) and current clause were used. The preceding preposition feature contained the head word of the previous chunk if it was labeled as a preposition. Attenuated words are wordforms occurring below a certain frequency that have been converted to a string that captures specific features (e.g. capitalization, suffixes, etc.) of the original word. Current clause was a binary feature that was on if the current word was in the same clause as the main verb. These three features were novel additions to the field of SRL.

Ahn et al. (2004) based much of their work on Gildea and Jurafsky (2002) since both systems aimed at the SRL task using FrameNet. Features used included the path, the frame name, lexical items in the path, POS tags, semantic classes, and subcategorization frames.

The system of Tjong Kim Sang et al. (2005) divided features into various categories, which included lexical, syntactic, semantic, and positional. Additionally, combinations of these types of features were used. All features used were from previous work (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Xue and Palmer, 2004).

Lexical features included the predicate, the first and last words of phrase, as well as the words immediately preceding and following the phrase. Syntactic features included POS tags (for all the words from the lexical features), path, subcategorization of the verb, and parent. Several paths were used including the standard as well as only those words before or after the verb. Semantic features included named entity tags for words from the lexical section of features. Positional features included position and distance, which were common in SRL systems (based on Gildea and Jurafsky (2002)). Distance was measured in parent nodes. Combinations of types of features were also used for purposes of this task.

One important consideration that came up with these systems was the choice of features, as noted by van den Bosch et al. (2004):

> "in previous research, we have found that memory-based learning is rather sensitive to the chosen features and the particular setting of its algorithmic parameters" (p. 103).

In particular, having too many (possibly redundant) features can decrease performance of the system. Bi-directional hill-climbing (Caruana and Freitag, 1994) was found to be useful in determining the most relevant features for the specific task at hand (van den Bosch et al., 2004; Tjong Kim Sang et al., 2005). Though these findings were from systems based on TiMBL, issues potentially are the same with AM. In fact, these issues have been noted in AM as well. This technique (bi-directional hill-climbing), or something akin to it, could be useful for ensuring that redundancy of features is not an issue in overall efficiency of the present system.

These MBL systems performed well within the SRL tasks at CoNLL 2004 and 2005. The MBL system with the best results, participating in 2005, obtained accuracies of around 70%. A detailed discussion is found in 3.4.1. As TiMBL systems were successful in SRL tasks, the present system was developed to determine if another MBL algorithm, analogical modeling, could provide similar results.

2.4 Analogical Modeling

The present system uses analogical modeling (AM) (Skousen, 1989), which is similar to other MBL algorithms such as TiMBL.

AM is an exemplar- (or instance-) based learning algorithm that uses data instances to predict outcomes for novel test instances. Both data and test instances contain a feature vector which defines a context for the instance. The data instances additionally have an outcome (or behavior) associated with the particular context.

The data instances are likened to 'stored memory tokens' by Eddington and Lonsdale (2007). This is because the algorithm first directly accesses each of these data contexts to determine which are included within the new test context. Data contexts displaying more similarity to the given test context more directly predict test context behavior, and this is determined only through the exemplars themselves.

The system creates all possible subcontexts of the given context, and possible data contexts are assigned to these subcontexts. Disagreements between the context and each subcontext are calculated. Further division of subcontexts and disagreement calculations (in a continuing process) provide the basis for discarding most 'inconsistent' (or heterogeneous) data contexts (those not likely to analogically influence the outcome of the test context). This process creates the analogical set, which in AM contains all the data instances that possibly have influence on the test context. From this set, the system either chooses the most common behavior or randomly chooses a data context that determines the behavior of the test context. Skousen (2003) notes that the system is thus procedural, and that "the significance of any combination of variables is always determined in terms of the given context for which we seek a predicted outcome." (p. 1).

Three factors are considered in determining which data contexts display the greatest similarity to the given context being tested. First, proximity compares the given context to data contexts; contexts that are more similar to the given context are chosen. Secondly, the gang effect multiplies the chances of an example being selected when a group of examples in a space behave similarly. Lastly, heterogeneity prohibits an example from being chosen if an intervening example closer to the context given behaves differently. Skousen (2003) provides an example that demonstrates how each of these principles work.

Analogical modeling has an important advantage in that it is robust. It can be used even when some data features are missing. It has been found to predict gang effects when exceptions are found near specific contexts given. Additionally, it is possible to essentially ignore certain features on certain instances.

Skousen et al. (2002) is a comprehensive source for AM. It provides examples of work done with AM for a variety of applications, compares AM with other exemplar-based algorithms, and addresses use of the computational program implementing the algorithm. Recent developments in the computational program, described by Eddington and Lonsdale (2007), provide the ability to use larger data sets and more features, a crucial point for the present work.

Chapter 4 further details preparation and running of the computational algorithm based on AM.

Chapter 3

Methodology

In order to develop the AM SRL system, it was necessary to create features that were amenable for processing. Features, some mentioned above, are the key for successful recognition of argument structure. This chapter explains the process required to create these features, starting from the basic data used. Tools used for converting the raw linguistic data into features are discussed. Additionally, tools provided by CoNLL that were proved useful in feature analsyis are described. Feature selection is then addressed. Finally, evaluation metrics are presented.

3.1 Data

CoNLL 2004[1] and 2005[2] SRL task data are freely available, but the former was only used briefly for preliminary work in this thesis. The data are divided up into three distinct sets. Training and development data are used to build the system. Carreras and Màrquez (2004) state that "the training set is used for training systems, whereas the development set is used to tune parameters of the learning systems and select the best model." A test set, for purposes of evaluating the final system, was also provided.

Data were taken from sections of the Wall Street Journal (WSJ) portion of the Penn TreeBank (Marcus et al., 1993), a corpus of parsed sentences in English which displays syntactic relations between items of sentences. A typical sentence is as follows:

He wouldn't accept anything of value from those he was writing about.

Additionally, corresponding sections of PropBank provided semantic annotations for sentences of the TreeBank. These annotations includes typical semantic roles such as agent and patient, as well as adjunctive roles. Sections used for each set of

[1] http://www.lsi.upc.edu/ srlconll/st04/st04.html
[2] http://www.lsi.upc.edu/∼srlconll/soft.html

Set	CoNLL-2004	CoNLL-2005
Training	15-18	2-21
Development	20	24
Test	21	23

Table 3.1: Sections of WSJ Used in CoNLL Datasets

CoNLL data are listed in Table 3.1. Note that CoNLL 2005 had a larger training set available.

Training and development data contain both features and outcomes. The features consist of lexical items, named entities, syntax information, etc. (as mentioned above). The outcomes describe the propositions for each target verb used, in PropBank format. See 3.1.3 for details.

3.1.1 Feature Data

Input WSJ data were available for all three sets. Information provided by CoNLL included lexical items, named entities, part-of-speech (POS) tags, chunks, clauses, and full syntactic informaton (only for CoNLL-2005). Chunks are bare phrases identified by shallow parsing techniques, and thus do not give the full syntactic relationship in the sentence. Verb sense information (PropBank verbs have an associated sense) was also provided in CoNLL-2005, but only for the training and development sets; it was not used for the present work.

Feature information was provided from preprocessor systems. POS tags were provided by Giménez and Màrquez (2003), using a system based on support vector machines trained on sections 0-18 of the Penn TreeBank. Carreras and Màrquez (2003) provided a chunker and clause recognizer built on voted perceptrons. This was used for the partial syntax features of CoNLL-2004. Named entities were obtained by maximum-entropy classifiers in a system by Chieu and Ng (2003).

These processors were all run in a pipeline, from POS tagging to chunking, then to clausal parsing, and finally to named entity recognition. Carreras and Màrquez (2004, 2005) provide information regarding performance of these individual processors.

3.1.2 PropBank

PropBank (Kingsbury and Palmer, 2002; Kingsbury et al., 2002; Palmer et al., 2005) is a set of predicate-argument annotations of sentences taken from the Penn TreeBank (Marcus et al., 1993). It served as the source of semantic annotations for CoNLL tasks, providing output data for the training and development sets as well as the standard for the test set. Propositions for annotated verbs from PropBank are freely available[3]. Since tagging was done by human annotators, guidelines (Babko-Malaya, 2005a,b) were prepared to improve consistency of annotations.

Each sentence can have a number of target verbs, and each target verb governs exactly one proposition. Propositions contain the set of arguments associated with semantic roles of the target verb. First, potential arguments are identified for each target verb of a sentence. The following identifies these constituents for the target verb 'accept'.

```
[He] [would] [n't] [_V accept] [anything of value] from
[those he was writing about].
```

Afterwards, argument labels are mapped to these constituents, as shown here:

```
[_{A0} He] [_{AM-MOD} would] [_{AM-NEG} n't] [_V accept] [_{A1} anything of value]
from [_{A2} those he was writing about].
```

The roles for the target verb (or predicate) 'accept' follow:

```
V: verb
A0: acceptor
A1: thing accepted
A2: accepted-from
AM-MOD: modal
AM-NEG: negation
```

Arguments can be either general or adjunct. The general (or numbered) arguments fill roles that are verb-specific. These arguments are labeled A0, A1, A2, A3, A4, or A5. Typically, A0 refers to the *agent* ('acceptor' in above example) role while A1 stands for either the *patient* or *theme* ('thing accepted' in above example) role. However, this completely depends on the verb along with the sense of the verb utilized in the sentence. A2-A5 arguments typically are verb-specific arguments. For this reason, no specific generalization of roles can be made between verbs or verb

[3]http://www.cs.rochester.edu/∼gildea/PropBank/Sort

senses. A0 and A1 arguments are the most frequently found roles in PropBank, while the last ones (A3, A4 and A5) are less common. The adjunct arguments (e.g. 'modal' verb and 'negation' marker in above example) can be any of the following, and these can occur for any verb:

```
AM-ADV: general purpose    AM-MOD: modal verb
AM-CAU: cause              AM-NEG: negation marker
AM-DIR: direction          AM-PNC: purpose
AM-DIS: discourse marker   AM-PRD: predication
AM-EXT: extent             AM-REC: reciprocal
AM-LOC: location           AM-TMP: temporal
AM-MNR: manner
```

These adjunctive roles indicate other important information regarding any verb present in PropBank. Discourse markers include items such as 'for example', 'but', and 'since'. Location roles refer to places where an action might take place, and manner refers to how an action is carried out.

Each verb in PropBank has a number of verb senses, taken from VerbNet. A set of possible roles is provided for each sense of each verb. This set of possible roles is called the *roleset*. PropBank *Frames files* defines the rolesets. These rolesets were also used in development of features, described in a following section.

Additionally, a C- tag could be added to the front of an argument to show continuation from a previous argument. A non-contiguous argument, shown below, was given by Carreras and Màrquez (2004).

$[_{A1}$ The apple], said John,$[_{C-A1}$ is on the table].

Another tag was also given (only for purposes of the CoNLL data). When co-referenced arguments were found in PropBank, the referent was given an R- tag to refer it back to the referenced argument. Consider the following sentence:

```
The deregulation of railroads and trucking companies that began in
1980 enabled shippers to bargain for transportation.
```

Here, 'that' (an R-A1) refers back to the phrase 'the deregulation' (an A1) for the target verb 'began'. Rules for matching pronominal expressions were used for tagging these arguments.

As a final note, a number of propositions were filtered out from the data sets provided by CoNLL. This was done to ensure that the arguments met certain requirements. Arguments could not overlap and had to be ordered sequentially. Additionally, arguments with incorrect role labels were not included in the data set for CoNLL.

To obtain these data, a verb by verb index of proposition frames is available online[4] and for download[5]. This offers a rich source for development of features describing common rolesets for particular verbs, and was used for this purpose.

Each verb of the index can have a number of different senses, and for each sense of each verb an annotated example is provided. These examples give a representative idea of what roles are available for each verb and corresponding sense. They may also provide information about the order of arguments for the target verb, but this can be affected by elements such as voice of the verb in a specific sentence.

Verb sense information was only given for the training and development sets. For this reason, verb sense information available in PropBank was not used for feature development. This was done in accordance with what had been in previous work, as only 20% of verbs within PropBank have more than one frameset (i.e. more than one sense) (see Kouchnir (2004)). Roleset features thus were all developed based on the first sense of the verb, no matter how many senses were present.

Let us refer to a concrete example, using the predicate 'carry', which has five senses. The corresponding file separates information for each sense (i.e. each roleset) of the predicate. We will incrementally look at the contents of the file for the first sense of this verb. The first area identifies the roleset as follows:

```
Roleset carry.01 Verbnet Class: 1 "bring with, have":
```

It shows the verb with its corresponding sense (in this case the first sense) as well as the verbnet class and a definition of the verb (in relation to the specific sense. The next portion of the file contains all the semantic roles (i.e. no adjunct role information is listed, as adjuncts can apply for any type of verb) possible for the current roleset. Note that here the A of the argument names is replaced with ARG. The verb with this roleset can take five different arguments, as demonstrated below.

```
Roles:  Arg0: carrier
        Arg1: thing carried
        Arg2: instrument
        Arg3: 'with' reflecting back on arg0
        Arg4: benefactive (predicted but not seen)
```

This roleset can contain up to five different arguments. It is noted that a higher number role is not possible without its corresponding lower roles (i.e. ARG4 does not

[4]http://verbs.colorado.edu/framesets/

[5]http://verbs.colorado.edu/verb-index/index.php

exist if ARG3 doesn't). A majority of verbs can only take ARG0, ARG1, and perhaps ARG2, so this is an interesting case. The final part (referred to as 'Examples' later on) of the roleset contains actual examples of sentences with verbs of that roleset. The current roleset contains three examples, as seen here:

```
Examples:  basic transitive (-)
           Citibank carries $150 million in earthquake insurance.
           Arg0:  Citibank
           REL:   carries
           Arg1:  $150 million in earthquake insurance

           with instrument object (-)
           T-shirts carried the school logo on the front.
           Arg0:     T-shirts
           REL:      carried
           Arg1:     the school logo
           Arg2-on:  the front

           with with (-)
           The transportation bill carries with it a permanent smoking
           ban.
           Arg0:       The transportation bill
           REL:        carries
           Arg3-with:  it
           Arg1:       a permanent smoking ban
```

This clearly points out that though a sense of a verb may be able to take certain arguments (A0-A4 for the present case), it does not necessarily mean that each case of that verb sense will have all of them. At the same time, though, it is certain that roles not present within the roleset will not appear in example sentences. With this information, precision (evaluation metric to be described later) could be improved if we could use it (roleset information) to restrict allowed arguments (doesn't permit extraneous arguments to be present).

The last example demonstrates the fact that arguments don't necessarily appear in the order that they are written in the roleset. Many linguistic phenomena (e.g. verbal voice change, phrasal or clausal movement, etc.) may occur that cause role elimination or movement for a specific sentence.

The verb 'carry' provides an interesting case because the other four senses are particle verbs ('carry on', 'carry out', 'carry over', 'carry off'). Only in these cases

```
1           2   3       4    5          6         7         8       9
J.C.    (PER*   NNP   (NP*   (S*   (S1(S(NP*         -      (A0*    (A0*
Penney       *)  NNP      *)   *           *)        -         *)      *)
will         *   MD    (VP*    *         (VP*        -   (AM-MOD*)      *
continue     *   VB       *    *         (VP*  continue       (V*)      *
to           *   TO       *    *       (S(VP*        -      (A1*       *
service      *   VB       *)   *         (VP*   service          *     (V*)
the          *   DT    (NP*    *         (NP*        -         *    (A1*
receivables  *   NN       *)   *            *))))))   -         *)      *)
.            *    .       *    *S)          *))       -         *       *
```

Figure 3.1: Example Set of CoNLL-2005 Data

does the predicate extend beyond one word. Though there are relatively few rolesets containing particle verbs overall, they are an important point of discussion that will be considered elsewhere.

3.1.3 Data Exemplified

An example set of CoNLL-2005 data for a single sentence is provided in Figure 3.1.3. Sets for CoNLL-2004 slightly differed (as described below) from this.

Each data type is represented in exactly one column. Each column provides a new piece of information regarding the current word of a sentence. Columns contain each of the following types of information.

1. Words (or another type of lexical item)
2. Named Entities
3. POS tags
4. Chunks
5. Clauses
6. Full syntactic information

Chunks, clauses, named entities and full syntactic information are provided in start-end format for the CoNLL-2005 data. Chunks and named entities in CoNLL-2004 data were represented in IOB2 format, with tags that indicate whether the current item was outside (O), begin (B), or are inside (I) the chunk or named entity.

For example, an I-NP tag would indicate that the current item is within a noun phrase. Note that with this type of tag there is no way to distinguish whether the item was in the middle or at the end of the phrase. However, tools are available for converting between tag types.

The CoNLL data were separated by type so that data columns could be ordered according to preference.

The columns following the input data described (columns 1-6 are input data) are output data (or propositions) (columns 7-9). The first of these (7) shows the two target verbs ('continue' and 'service') while the other two represent the propositions for each target verb (8 for the first and 9 for the second verb). The number of columns after column 7 depends on the number of target verbs in column 7.

3.2 System Scripts and Tools

A number of tools were used in the development of the current system. Some of these were used to create additional features from raw linguistic data for use within the system. Others (a few provided by CoNLL) were used for discovering utility of features. The following discusses these two types of tools.

3.2.1 Feature Creation Using Perl

The main tool used in creation of features was the Practical Extraction and Report Language (or Perl for short). This freely and widely available scripting language[6] allows for manipulation of linguistic data (see Hammond (2003), for example). With this capability, a number of different features were easily extracted for use within the current system.

Although not used extensively for the current project, it is worth noting the existence of a Perl module for manipulating TreeBank structures. The Lingua Treebank[7] extension by Jeremy Kahn is freely available. It enables extraction of natural objects from TreeBank structures. Methods within the module allow for direct access to syntactic constituents. These could be used for possible argument boundary identification, since most arguments in propositions were found to coincide with constituent boundaries (Kouchnir, 2004). In the present work, some Perl code was written to identify boundaries for these potential arguments. This will be described elsewhere.

[6] http://www.perl.com/download.csp

[7] http://search.cpan.org/search?query=Lingua+Treebank

A Perl script (successive versions of 'mk-AMvecs.pl') was incrementally built as new features were developed. The script outputs features in the format necessary for processing in AM, which was the base of the current system. Any combination of features desired, from only one feature to the whole set of features (30+) can be printed from this script, depending on what is desired. Further description of the process will be given in the following chapter, which provides the architecture of the complete system.

3.2.2 PropBank Feature Extraction

The PropBank index provides useful information for features. In order to take advantage of this, framesets were obtained and Perl code ('rolegetter.pl') was developed to extract information about each roleset (and sense) for each verb. The code outputs a line for each roleset that contains the verb, the sense, and the specific roles possible for that roleset. For the verb 'carry', the corresponding output appears as follows:

```
carry 01 0 1 2 3 4
carry 02 0 1 2 3
carry 03 0 1
carry 04 0 1 2 3
carry 05 0 1
```

Note the difference in possible arguments for each sense of the verb. This point could be useful if the sense of the target verb could be identified for each sentence. However, this information is only provided in CoNLL training and development sets.

It is also possible to extract the potential order (including where the predicate is placed within the order) of roles for a particular sentence with a given roleset. This information could be useful in feature creation, or in other necessary procedures (described in the next chapter) that prepare output from AM processing for evaluation based on CoNLL metrics.

Order information was extracted via Perl from blocks of example sentences (such as the ones shown above). This also allowed for order in relation to the predicate as the 'REL' argument within the block identified the target verb of each example sentence.The Perl script reads each example for each sense of each verb and prints out a summary of arguments (only numbered arguments, not adjuncts) in relation to verb for each of the examples. This provides another file similar to the one above, with slight differences as to content. No sense information are available, and the

arguments listed after the verb are the numbered arguments in order of appearance within the sentence.

To understand the usefulness of this information, consider a sentence with a few arguments for a certain target verb. AM processing could correctly recognize all potential arguments, but not discover the correct type of argument for each. A feature describing the typical order of arguments could be useful for identifying the role types for each argument recognized.

A0 generally precedes the verb while other arguments (at least numbered arguments) follow it. However, orders do switch as different linguistic phenomena play a part in a sentence. This could be controlled if the information regarding voice and other issues was available, and certainly this seems obtainable from PropBank or TreeBank.

In order to evaluate this, I searched the created file of examples (containing all verbs from PropBank accompanied by all their senses as well as examples). I found three verbs that had an example with both arguments A0 and A1 before the verb. In all three cases, there was a mistake in the order of the arguments given in relation to the order they were in the sentence. A1 always seems to follow the verb (at least in the general case). I found one other example of A0 and A2 together before the verb. This seemed to be legitimate because the A2 was an ARG2-MANNER role that immediately preceded the verb. It might be best to say that in general A0 will precede the verb, while A1 always follows it. It doesn't seem correct to say that the other argument types necessarily follow the verb.

If generalizations can be made, it might not be necessary to explicitly compare the typical roleset with the guessed roles for a sentence for the purposes described above. However, in order to eliminate guessed arguments that do not have a place within the roleset of the verb, something of this nature must be accomplished.

The above work served as a foundation from which features could be extracted about the roleset to use in the system. More information about features, including features based on this work, will be provided later.

In some cases, the PropBank example gives information (e.g. person, tense, aspect, voice) regarding the form of the verb within the sentence. This information could be useful for determining how argument order changes in response to a certain verbal conjugation. It certainly is expected that voice differences could affect the order (and even presence) of certain roles.

3.2.3 Output from AM

Currently, output from the AM program is captured in a file called 'amcpresults'. Information from this file can be (and has been in the past for other projects) very helpful. The following will describe the contents of these files, through looking at portions of an actual file created in the current work.

The file first contains a block for each instance tested in AM. This block first shows the 'Context' (or set of features) that were tested in AM computations, as well as the total number of data items. Other information given describes AM processing. Afterwards, a statistical summary is provided. Each output type (for SRL labeling, these types are equivalent to role types) guessed by the system for this instance is shown, along with numbers that determine which output is the most likely according to the analogical algorithms of AM. The output with the highest associated percentage wins. Additionally, a statement is provided that indicates whether the correct (if output data are provided) outcome was guessed by the system. In the example block below, the winning argument matches the expected outcome '(A1*' and the correct outcome is predicted.

```
Given Context:  nervous (NP* 2 buck = AVO = RP JJ NNS IN,
Include context even if it is in the data file
Number of data items: 262630
All data items considered
Total Excluded: 0
Nulls: exclude
Gang: squared
Number of active variables: 9

Statistical Summary

(A1*         2553   99.805%
*               3    0.117%
*)              2    0.078%
             ----
             2558

Expected outcome: (A1*
Correct outcome predicted.
```

After all instance blocks have been output, the ending portion of 'amcpresults' gives a summary for each output type (for SRL outputs these are role types) in the whole AM test set. An example for the argument '(A0*' follows:

```
Test items with outcome (A0*)       were predicted as follows:
 80.645% (A0*)        (25/31)
  3.226% (A1*)        (1/31)
```

```
12.903% *           (4/31)
 3.226% *)          (1/31)
```

For each output (or argument) type, a summary is provided showing how often this type was guessed correctly. It also shows the percentage of times that it was guessed as other output types. There are a total of 31 '(A0*)' outputs in the test set, with 25 of them being guessed correctly. Other outputs guessed for these outcomes were '(A1*)', '*', and '*)'.

The last line (shown below) provides the total number of correct predictions. In this case there were 4870 instances in the test set; this run guessed ∼91.9% of instance outcomes correctly. It is noted that this number did not necessarily correlate with overall evaluation, as some runs with more correctly guessed instances resulted in lower F1 scores.

```
Number of correct predictions: 4475
```

It is important to note that this last information (summary of outputs and overall number of correct predictions) is provided in 'amcpresults' only if the specific outcomes for the AM test set are given. When running the actual CoNLL test set, these outcomes are not given. However, when developing and tuning the system, training data could be used for an AM test set. Doing this allowed for the actual output (annotated argument roles) to be present, and so all described information was present.

This information can be useful when trying to distinguish between argument types, and has been used in the present work. As an example, at one point I discovered that closures of arguments were not being guessed correctly a high percentage of the time (only ∼54%). Because of this, I focused on creating features to identify the closures of arguments specifically. With this help, percentage of closures guessed correctly increased to well over 70% on most runs, and a couple runs were able to achieve percentages around 85%.

This file is thus very helpful in being able to determine whether specific argument types (with an emphasis on boundaries of arguments) are being guessed at a high percentage overall within the set tested. Some of this information, when combined with other information from CoNLL tools to be described next, can be of great utility.

3.2.4 CoNLL Tools

When data sets from CoNLL are downloaded, a set of tools (Perl programs) that have been developed is also provided. An accompanying README file explains the purposes for each of these and describes how each is used.

There are a number of scripts (in Perl code) available from CoNLL. These include 'col-format.pl', 'prop-discr.pl', 'prop-filter.pl', and 'srl-eval.pl'.

The first of these is a script that will take the start-end format for NE, clauses, and chunks (from 2005 data) and turn that into the IOB2 format. These formats were described above when discussing the CoNLL data in detail. This other format might be useful for feature development.

The 'prop-discr.pl' and 'prop-filter.pl' scripts are very useful, especially when combined together in a pipeline. The first of these takes two proposition files and creates new files that contain propositions that are just in one and not the other (and vice versa) as well as props that are common to both (disjoints and union of input proposition files). This enables comparison of propositions output from a system with the actual propositions.

The 'prop-filter' code takes a props file and prints out only specified filtered arguments. This was useful when trying to determine whether a certain argument type was being guessed correctly, as you could ignore other argument types in a file.

The final script was used for evaluation of the system. This will be described in detail in a subsequent section, as well as in relation to the architecture in the following chapter. We will note here that the outcome from the evaluation set was useful as well, since it displays the actual scores for each individual argument type. An option on the 'srl-eval.pl' file allowed for the generation of a confusion matrix, that gave great insight into the recognition of arguments in separation from actual labeling of these arguments. This was helpful in being able to determine which arguments were correctly delineated (borders identified) but had problems when labeling occurred.

The following section will look at how feature selection took place.

3.3 Feature Selection

Features were mainly selected from the review of previous literature. All were tried out to some extent, with some not being as useful as others. Determining the utility of individual features was done by following certain guidelines that will be described here.

Based on suggestions by the TiMBL work done in SRL, feature sets were developed incrementally. One important consideration that came up with these systems was the choice of features. Recall that van den Bosch et al. (2004) noted that previous work had indicated that MBL work was sensitive to chosen features. A particular problem was the possibility of redundant features decreasing system performance. Tjong Kim Sang et al. (2005) also used the same technique (Caruana and Freitag, 1994) for feature selection.

A method was developed for the current work to incrementally build feature sets, similar to the process described in the above references. In this way, the best individual feature set could be developed without worry of redundancy. It is noted that this worry did seem reasonable, in that many smaller sets of features performed much better than much larger sets (noted also by Baayen and Moscoso del Prado Martín (2005)). This seems to be attributed to redundancy or harmful features. A particular example was a case when a set of 27 features was run. At this point in time, this was all the features that were currently available in the present work. It did not help to have such large feature sets unless they are progressively shown to improve as feature set size increases.

We will next look at the metrics used for evaluation of the system.

3.4 Evaluation

In order to evaluate a system, certain metrics must be developed. CoNLL organizers provided a Perl script[8] for automatic evaluation of guessed propositions, based on three different metrics.

CoNLL data were separated into three sections. The third was a test set that only included input data (in contrast to the other two sets, where output data were also provided).

Evaluation took place based on the guessed outputs for this test set by a system participating in the task. Arguments had to be completely recognized, which meant that the argument needed to be labeled correctly and that the span of the argument had to be correct. Note that the verb argument was not included for purposes of evaluation.

An evaluation output from a preliminary run in this study is shown in Figure 3.2.

The automatically generated report is quite useful. The first part provides the number of sentences tested, the number of propositions within those sentences,

[8] called srl-eval.pl

```
Number of Sentences    :          73
Number of Propositions :         163
Percentage of perfect props :   6.13

              corr.  excess  missed    prec.    rec.       F1
------------------------------------------------------------
   Overall      117     156     329    42.86   26.23   32.55
----------
        A0       40      36      80    52.63   33.33   40.82
        A1       44      58     111    43.14   28.39   34.24
        A2        4      16      26    20.00   13.33   16.00
        A3        0       2      12     0.00    0.00    0.00
        A4        0       1       4     0.00    0.00    0.00
    AM-ADV        0       3      13     0.00    0.00    0.00
    AM-CAU        0       0       2     0.00    0.00    0.00
    AM-DIR        0       2       1     0.00    0.00    0.00
    AM-DIS        6       7       7    46.15   46.15   46.15
    AM-EXT        0       0       2     0.00    0.00    0.00
    AM-LOC        3       6       8    33.33   27.27   30.00
    AM-MNR        0      10      23     0.00    0.00    0.00
    AM-MOD       13       1       1    92.86   92.86   92.86
    AM-NEG        3       0       0   100.00  100.00  100.00
    AM-PNC        0       0       5     0.00    0.00    0.00
    AM-TMP        1      13      26     7.14    3.70    4.88
      R-A0        3       0       5   100.00   37.50   54.55
      R-A1        0       0       2     0.00    0.00    0.00
  R-AM-LOC        0       0       1     0.00    0.00    0.00
  R-AM-TMP        0       1       0     0.00    0.00    0.00
------------------------------------------------------------
         V      155       5       8    96.88   95.09   95.98
------------------------------------------------------------
```

Figure 3.2: Evaluation Example Output

as well as the percentage of all propositions that were guessed completely correctly (perfect propositions, where all arguments were identified and labeled without error). It then provides scores based on three metrics for each individual argument type, as well as an overall score. These metrics are often used in evaluating machine learning approaches; more details are available elsewhere (van Rijsbergen, 1979).

Notice that the three columns following the argument type column (headed by 'Overall') represent the number of correct (A_{corr}) (guessed and correct) arguments, the number of excess (A_{excess}) (guessed but not correct) arguments, and the number of missed (A_{miss}) (not guessed but should have been) arguments. The sum of correct and missed arguments equals the total number of arguments found (A_{real}) within the set of data being tested. The sum of correct and excess arguments equals the total number of guessed arguments A_{guess}.

The first metric, precision p gives the ratio of all arguments predicted by the system which are correct:

$$p = \frac{A_{corr}}{A_{corr} + A_{excess}} = \frac{A_{corr}}{A_{guess}} = \frac{117}{117 + 156} \tag{3.1}$$

The p for the overall set is equivalent to .4286, which is listed in Figure 3.2 as 42.86.

Next, the recall r gives the ratio of all the real arguments (as found in the test set) which are predicted:

$$r = \frac{A_{corr}}{A_{corr} + A_{miss}} = \frac{A_{corr}}{A_{real}} = \frac{117}{117 + 329} \tag{3.2}$$

The recall calculated from the above situation is .2623, listed as 26.23 above.

The last standard of measurement was the F_1 score, which is the harmonic mean of the first two standards. It is the final standard upon which systems are assessed. It can be calculated as follows:

$$F_1 = \frac{2pr}{p + r} = \frac{2 \cdot .4286 \cdot .2623}{.4286 + .2623} \tag{3.3}$$

The calculated score is equal to .3255, listed as 32.55. Notice that each calculated score here is equivalent to the score given for the 'Overall' situation in the evaluation output above.

Additionally, the 'srl-eval.pl' evaluation program provides an optional setting that could print out a confusion matrix. This gave the ability to get scores based not only on complete labeling, but also on recognition.

	-1	0	1	2	3	4	5
-1: -NONE-	0	25	42	5	4	0	1
0: A0	62	57	1	0	0	0	0
1: A1	70	11	71	2	0	0	0
2: A2	16	0	5	8	0	0	0
3: A3	9	0	0	0	3	0	0
4: A4	2	0	0	1	0	1	0

Table 3.2: Condensed Confusion Matrix

This extra information about recognition helps us see what types of arguments are getting mixed up. Analyzing this information could potentially help in the creation of features to distinguish these roles to a greater extent. The first part of a confusion matrix displays overall information for recognition, just as is given in the first line of the evaluation output:

```
---------------------------------------------------------------------
              corr.  excess  missed    prec.    rec.      F1    lAcc
 Unlabeled      206      89     240    69.83   46.19   55.60   84.95
---------------------------------------------------------------------
```

Afterwards, the matrix is given. Table 3.2 is a condensed version of the matrix, displaying results only for the core arguments (no adjunct arguments).

The rows of the table represent the arguments actually present within the tested propositions, while the columns represent the guessed arguments. The row labeled '-1: -NONE' represents excess arguments (guessed but not correct). The column labeled '-1' indicates the number of missed arguments.

To read the matrix, consider the row starting with '1: A1'. The sum of the numbers in the row should equal the total number of A1 arguments in the test set. There are 70 missing, 11 that are guessed as A0, 71 that are guessed correctly (in column '1'), and 2 guessed as A2. A total of 13 arguments were recognized (boundaries identified) correctly, but labeled as either A0 or A2 instead of the correct A1. This is a potential area of improvement, as there might be methods for improving the labeling of arguments.

Metric	Individual	Individual (w/ post-processing)	Combined
Precision	70.70	73.84	76.79
Recall	69.85	69.88	70.01
F1	70.27	71.80	73.24

Table 3.3: Results of System by Tjong Kim Sang et al. (2005)

Metric	Ahn (2004)	Kouchnir (2004)	van den Bosch (2004)	Tjong Kim Sang (2005)
Precision	73.50	56.86	67.12	70.70
Recall	63.60	49.95	54.46	69.85
F1	-	53.18	60.13	70.27

Table 3.4: Previous Results of MBL Systems on SRL Tasks

3.4.1 Evaluation Exemplified

Here we will provide results for the systems that most closely parallel the present approach. These systems performed well at SRL tasks in CoNLL-2004, CoNLL-2005, and Senseval-3. Table 3.3 shows results for the only MBL system (Tjong Kim Sang et al., 2005) represented at CoNLL-2005.

Though using the same data set and a similar MBL algorithm, the present model differs in important ways from the system of Tjong Kim Sang et al. (2005). Their system combined maximum entropy models, support vector machines, and memory-based learning. Table 3.3 includes results for both the combined system and the individual MBL system. Additionally, results for the individual system after a post-processing technique was applied are given. For a comparison case, we will use the individual system without post-processing.

Table 3.4 provides the F1 (overall) scores for previously mentioned MBL systems designed for SRL.

In most systems above, recall scores were much lower than precision scores. The system under consideration here provided similar results in most cases.

The systems by Ahn et al. (2004); Tjong Kim Sang et al. (2005) do not directly compare with the other two or with each other. The first of these participated in Senseval-3 (based on FrameNet), while the second was involved in CoNLL-2005. The

other two participated in CoNLL-2004, where full syntactic information (available in 2005) was not provided. Additionally, the evaluation metrics were slightly different in Senseval-3. The numbers above represent the best match. Additionally, no F1 score was reported by Ahn et al. (2004), but this could be calculated relatively easily.

Chapter 4

System Architecture

The current system uses AM to determine SRL task outcomes, but a number of pre- and post-processing steps are required. All files and programs used in the current system are available at Brigham Young University. These include the AM program itself, the Penn Treebank, the actual data (training, development, and test sections) from CoNLL-2005, and PropBank. The system goes through five basic processes to complete a cycle:

1. Data preparation
2. Processing of data using AM
3. Results from AM obtained and corrected
4. Preparation for evaluation
5. Evaluation of data for SRL task

4.1 Data Preparation

Data and test files are first prepared in a format for AM processing, detailed in Lonsdale (2002) and Parkinson (2002). Perl code is used for formatting data. File preparation also involves temporarily ignoring sentences containing no arguments. Running these in AM is not reasonable because it would waste time in addition to giving data that do not conform to observed arguments from other sentences. The sentences without given arguments were typically short, as seen here:

```
Is      *  VBZ    *  (S*   (S1(SQ*     -
this    *  DT  (NP*)   *      (NP*)    -
the     *  DT  (NP*    *   (NP(NP*     -
future  *  NN     *)   *         *)    -
of      *  IN  (PP*)   *      (PP*     -
chamber *  NN  (NP*    *      (NP*     -
music   *  NN     *)   *         *)))  -
?       *  .      *    *S)       *))   -
```

Longer sentences without target verbs were noted as well. For example:

> "That's because the male part, the tassel, and the female, the ear, are some distance apart on the corn plant."

It appeared that all sentences containing no propositions (i.e. target verbs) only contained some form of the verb 'to be', though it is possible that other sentences with other verbs may contain no proposition.

Eliminating these sentences for this step created the need to resynchronize the data before final evaluation, which is explained in a following section.

The vector creation program 'mk-AMvecs.pl' (written in Perl) for AM can produce all features at once or can just produce new features prepared (or any desired combination of features). Thus we can avoid reproducing the whole set of features every time a new one is ready to test.

To create a vector set for AM processing, the program is set to print out first the outcome followed by a comma. Afterwards, each desired feature is separated by a space on the same line. The Perl program is then run, taking as input the training data. Some typical vector sets with the combination of features LCHSOTVBBSBTPS are shown below for a complete sentence:

```
     (A0*,   J.C.         (NP*  (S1(S(NP*       -3   continue  active  =  ?+?+NNP
        *),  Penney          *)         *)      -2   continue  active  =  ?+NNP+NNP
(AM-MOD*),   will         (VP*       (VP*       -1   continue  active  =  NNP+NNP+MD
      (V*),  continue        *        (VP*        0   continue  active  =  NNP+MD+VB
     (A1*,   to              *     (S(VP*         1   continue  active  =  MD+VB+TO
        *,   service         *)      (VP*        2   continue  active  =  VB+TO+VB
        *,   the          (NP*       (NP*        3   continue  active  =  TO+VB+DT
        *),  receivables     *)         *))))))  4   continue  active  =  VB+DT+NN
        *,   *PERIOD*        *          *))      5   continue  active  =  DT+NN+*PERIOD*
```

Each individual lexical item of each sentence has a generated vector of features which defines its AM instance.

4.2 Running AM

After all files are prepared, the AM program is run. Details on running the program are available in Parkinson (2002). The AM program analogically determines an outcome (here, the presence of arguments) based on the features provided for each individual word (or wordform) of a sentence. In the example sentence above, the outcome (placed before the comma) results from the corresponding feature set that follows. The AM run generates an 'amcpresults' output file that contains information about the run and guessed outcomes.

4.3 Obtaining and Correcting AM Results

Results from AM must be converted into another format for SRL evaluation. A Perl program was developed to parse the results for this purpose. Post-processing steps are then run to correct, resynchronize, and prepare this file for evaluation. Resynchronization involves replacing the sentences ignored in AM processing.

The parsing process provides results in correct format for evaluation, but improper argument structures still cause evaluation to fail. AM instances for SRL are word-based and therefore AM is not capable in this task of independently defining phrasal/clausal boundaries.

Arguments in PropBank do not overlap and each argument has exactly one opening and one ending; improper structures result as these rules are violated in guessed propositions. First, the system might guess a closure of an argument when no accompanying opening has been identified. Secondly, an argument might be identified where a previous argument has not closed. Additionally, open arguments sometimes extend to the end of the sentence without closure. In rare cases, an argument might be opened (and not closed simultaneously) on the last word of the sentence. This is an impossible scenario because of punctuation closing the sentence. The output set on the left below displays some of these inconsistencies, while that on the right is in a correct format for evaluation.

```
*              *
*              *
*              *
*              *
*              *
*)             *
(AM-MOD*)      (AM-MOD*)
(V*)           (V*)
(A1*           (A1*
*              *
*              *)
*              *
```

The modal argument is obviously complete, as it has both an opening and a closing. Two errors appear in the left column (AM results that have been parsed but not corrected). A closing argument item appears without an opening, and an A1 argument is started without being closed.

To solve these (and other) problems, a Perl program identifies argument boundary discrepancies and corrects incomplete arguments. The AM results after correction are displayed in the right column. Currently, corrections are completed in the simplest way possible. If a closure is identified without any previous opening, then an

opening is created at either the beginning of the sentence or at the end of the previous argument. Likewise, if an argument is opened without the previous argument's closure, a closure is created at the immediately preceding line.

These procedures could be improved to discover the argument structure in a more sophisticated manner, but this provided a first attempt at correcting this problem. Some other possibilities were briefly tested, and will be commented on in 6.2.4. The evaluation program (explained next) expects arguments to be complete (with both an opening and a closing) and to not overlap. With these corrections a complete evaluation is now possible.

4.4 Evaluation

The evaluation program was provided for the CoNLL-2005 SRL task, so that direct comparisons could be made between participating systems. The Perl program reads in two files, one containing the actual propositions and the other containing the guessed propositions. These files are then compared by the program and a summary of the results is prepared by the system. Precision, recall, and the F1 score are computed for each argument type individually as well as for all arguments of the set. For an argument to be correctly identified, it needs to span all the words of the argument (recognition) and the argument role has to be correct (labeling).

Chapter 5

Results and Discussion

5.1 Preliminary Work

A basic set of features (directly obtained or derivable from given information from CoNLL) were used in a preliminary AM system. The first group were 'direct' features, and correspond with the columns of data given by CoNLL. These included lexical words, POS tags, base chunk information, NE tags, and clausal information. The target verb column was also used as a feature. All these features were directly available from CoNLL data, only requiring transfer into a format for AM processing.

Additionally, some other features (called 'derived' to differentiate them from the 'direct' features) provided by further processing of direct data were used. These included two path features (indicating syntactic path from a target verb to a potential candidate), previous preposition, distance to target verb, and the position of the potential argument in relation to the target verb. Table 5.1 describes each of these features.

No full syntactic information (available in CoNLL-2005) was used in these features. These preliminary runs would therefore be more comparable to results from CoNLL-2004, where this syntactic information was not available.

Preliminary runs with the AM system were completed for three different data sets. A test set of unseen data was evaluated for each case. These data were taken from the training data available from CoNLL, which allowed comparison to actual PropBank propositions. Information regarding the size of these sets is found in Table 5.2. The number of instances is equivalent to the number of lexical items found in the set. The number of propositions is the number of target verbs (each of which governs a proposition) for identification within the set.

As a side note, the data sets used here seem to be the largest ever used in AM. This will be commented on later.

A number of data sets were used to evaluate the assumption that more data would provide better results. Table 5.3 summarizes results from these three runs,

Code	Feature	Description
L	lexical item	the word or other lexical item
Ps	part-of-speech	POS tag for the word or lexical item
Ch	base chunk info	phrase type, location w/in phrase (begin, middle, end)
N	named entity	tag showing named entities (NE)
Cl	clausal info	information on syntactic clauses
Vb	target verbs	target verb if present on current item, otherwise a '-'
U	up path	syntactic path from item to common parent w/ verb
D	down path	syntactic path from common parent to verb
V	previous prep	previous preposition occurring in sentence
O	offset/distance	distance from the target verb
P	position	position of current item with respect to target verb

Table 5.1: Preliminary Features Used in Present System

Measure	DS1	DS2	DS3	Test
Instances	50000	135389	262630	4870
Propositions	1693	4592	9134	163
Sentences	753	1989	3931	73

Table 5.2: Preliminary Data and Testing Set Sizes

Metric	DS1	DS2	DS3
Precision	42.86	45.92	47.84
Recall	26.23	30.27	32.29
F1	32.55	36.49	38.55

Table 5.3: Preliminary Results

# of Features	DS1	DS2	DS3
Base Case (11)	32.55	36.49	38.55
10	35.00	38.07	40.57
9	33.60	38.12	40.99
7	34.32	38.22	37.73

Table 5.4: Effects of Feature Set Chosen (F1 Score)

showing that, as assumed, larger data sets led to improvement. Much more training data are available, and it was expected that results would continue improving with increased amounts of data. However, a practical limit was expected as well. At a later point, a larger data set (approximately double of DS3, 538299 instances) was briefly tested, but improvements were only noted for recognition, and these were small.

Obviously, these preliminary results (high F1 of 38.55) do not even approach those from previous work. The system of Kouchnir (2004), with the lowest scores obtained on TiMBL systems, provided an F1 score of 53.18. However, these basic runs demonstrated that the present system is feasible in this type of setting (the SRL task), so system development continued. Throughout this process, the largest data set used (herein called 'DS3') and the test set (called 'thesis test set' from this point on) described in Table 5.2 were used.

5.2 Development of System

5.2.1 Testing Feature Set Size

Further work took into account the fact that some of the features present in the above list could have been redundant, as MBL results are sensitive to the features chosen (van den Bosch et al., 2004; Tjong Kim Sang et al., 2005). Table 5.4 compares results (based only on F1 score) for three smaller feature sets with the preliminary set. In each case, a different number of features was taken away from the base set (of 11 features). At this point, these features won't be named as we are only interested in looking at the redundancy factor. The F1 score was chosen as an overall comparison since it takes into account both other metrics. Larger combinations do not necessarily give better results. For this reason, choice of features seems to be important for the present system (in accordance with related previous literature).

For further development of the system, feature combinations were built up successively in a manner similar to Caruana and Freitag (1994). In practice this meant

	Combination	F1
Individual Feature	Tps	3.42
	L	3.52
	O	**13.49**
2 Featuress	PsO	28.70
	LO	30.46
	SO	**32.24**
3 Featuress	PsSO	36.77
	SOTps	40.59
	LSO	**41.21**

Table 5.5: Sequential Combination Development Based on F1 Scores

that the best individual feature was chosen first, and then the best combination of two features was chosen (by determining which feature addition to the best individual feature) resulted in the greatest gain. The process continued in this way to ensure that the best possible combination was found for this specific task.

This sequential method is demonstrated briefly in Table 5.5, showing combination growth selection from 1 to 2 features and subsequently to 3 features. Best selections for each combination size are shown in bold.

The best individual feature (O, or distance/offset) achieved an F1 score of 13.49. To determine the best set on the next feature size (of 2 features), runs were evaluated by adding other features to this one individually. Two-feature combinations with the best results are listed in the table. All three (PsO, LO, and SO) result in more than double the score of the individual feature. The highest score was provided by the combination of distance (O) and the full syntax (S) feature. This was in turn used as a basis for determining the best combination of three features. Just as O was used as a basis for determining the best combination of two features, SO was used to determine the next best set.

Features from the other close combinations (for example, Ps and L) hinted at what could be added to get the next best combination (in this case, three features). Adding these to the best two feature combination gave some of the best results for the three feature combination, as shown in Table 5.5. LSO was the best three-feature combination. The feature Tps shown is a trigram of POS tags; this additionally gave good results when combined with the two feature combination. Redundancy is

Metric	Labeling	Recognition
Precision	59.00	70.00
Recall	39.69	47.09
F1	47.45	56.30

Table 5.6: Results after First Round of Testing

further demonstrated here, as the three-feature combinations provided comparable scores to the larger feature sets above.

It was straightforward to determine when a certain feature was not useful at all and hence could be discarded for practical purposes. A good example of this was the feature VB, which when added to the previous base case caused no change in results. This was observed from up through the combination of five features, and thus this feature was not implemented in further runs.

5.2.2 Improvements through Boundary Recognition

After testing several features and feature combinations, results did not reach those of previous systems. The highest F1 score was 47.45 (using a set of 10 features, LCHCLSOPTVBBSB). The new features here (TVB, B and SB) are the target verb, the voice, and another feature created that relates to position of SBAR chunks within the sentence. The target verb provides the verb that defines the proposition as a feature. The SB feature was created because of 'to' clauses, which seemed to indicate argument boundaries in many cases. Table 5.6 summarizes scores for this combination.

This table introduces a distinction between labeling and recognition scores. Results reported above were labeling scores, meaning that the correct roles had to be provided in addition to recognizing that a certain constituent was an argument. Recognition only concerns the identification of the boundaries of an argument, even if it was labeled as another type of argument. Note in Table 5.6 that recognition is higher, reflecting the fact that many other arguments are correctly identified (or located) even though not appropriately labeled.

Obviously, with the high score of a little over 47 we are still lower than the scores of previous systems. With help from the tools discussed in Chapter 3, it became apparent that there were several arguments being partially identified (i.e. either the opening or closing of the argument was identified correctly while the other boundary

Metric	Labeling	Recognition
Precision	49.51	73.40
Recall	45.07	66.82
F1	47.18	69.95

Table 5.7: Results after Boundary Identification

wasn't). Thus, recognition of argument boundaries prior to labeling is critical. This aligns well with findings by Pradhan et al. (2005). "A detailed error analysis of our baseline system indicates that the identification problem poses a significant bottleneck to improving overall system performance." For this reason, many of the previously mentioned systems had separate modules for recognition and labeling. The present system does not attempt this directly, instead containing both features for recognition (described here) and others for labeling.

System output data were thus examined to determine how to improve recognition of argument boundaries, and new features were developed for this purpose. Many of these features rely on the syntax of the sentence, relating the target verb's position to other chunks of the sentence. Features to identify clausal boundaries have been implemented as well.

Other new features developed attempt to identify exactly and only the boundaries of potential arguments. These focus mostly on complete phrases (or clauses) that are syntactically linked to the target verb, and thus are possible constituents. The feature has a null value '=' for any element of the sentence that isn't identified as a potential boundary while potential boundary points are marked with items such as BVO, BVC, AVO, or AVO-AVC. These respectively mean 'before verb open', 'before verb close', 'after verb open', and 'after verb open - after verb close'. The final one identifies potential arguments that are only one word long, which happens frequently.

Experimenting with various versions (and combinations) of these new features provided good improvement. For example, a five feature set (LSOFvn5Pv2) using these features obtained nearly the same score as the 10 feature set described above (47.18 v. 47.45 F1 scores). Scores for this combination are shown in Table 5.7. The two new features (Fvn5 and Pv2) identify potential boundaries before and after the verb.

On this smaller set (of 5 features), the gap between precision and recall has lessened as recall improved. Improvements are greatest in terms of recognition; compare the F1 score of ∼70 on this smaller set to the score obtained on the larger set of 56.3. Targeting recognition of boundaries with these features did indeed help out.

5.3 Final Improvements

As a final step, a number of features were added that were only possible after argument boundaries were identified. These features were believed to be useful for distinction of recognized arguments.

The best feature set at this stage was chosen for purposes of testing on the CoNLL test set. This combination contained fifteen features, and was LCHSOTVB-SBFVN5PV2PAPEAPFAPPVAPSSPTHPT. New features are explained in Table 5.8. All of these were created only after argument boundaries had been identified.

Consider briefly the utility of the path (Pth) feature. Though the full syntactic information (S) provided by CoNLL is quite useful, a path feature is more informative. The S feature is a static description of the Treebank structure for a certain lexical item, while the path enables us to envision how the target verb and the current item are related hierarchically in the TreeBank representation.

Consider the following to appreciate the difference between these two features. For the same lexical item, the syntax and path features are as follows:

```
Syntax: (PP*
  Path: PP^S^S_S_VP
```

The syntactic item only indicates that a prepositional phrase begins at this point of the sentence. The path feature indicates that the PP is embedded within two successive clauses (indicated by the carets); the PP also includes (indicated by underscores) a clause which dominates the target VP. This provides a much more detailed picture of how this potential argument boundary should be seen in terms of the complete sentence.

As a slightly different example, consider the following:

```
Syntax: (S1(S(NP*
  Path: NP^S_VP
```

In this case, much of the information contained in the path is also present within the syntax feature; for example we know that the NP is dominated by an S.

Code	Description
Pap	POS tag for element previous to argument
Eap	POS tag for last element of recognized argument
Fap	POS tag for element following argument
Pva	arguments immediately prior to target verb
Pss	POS tag for first element of certain (1-line or PP) recognized arguments
Pth	complete path only given for start of argument elements
Pt	phrase type as described by Gildea and Jurafsky (2002)

Table 5.8: Introduction of New Features on Final Combination

Metric	Labeling	Recognition
Precision	61.27	77.45
Recall	51.79	65.47
F1	56.14	70.96

Table 5.9: Results of Final Combination

However, the path features displays the relationship of this NP to the target verb's VP (i.e. the two are contained in the same sentence). This is a particularly common path that seems to indicate a typical A0 (agent) argument.

This best combination obtained the overall scores shown in Table 5.9 for the thesis test set used in development.

Table 5.10 shows the complete development of the combination from one feature (O) to the final set of fifteen features. Figure 5.1 shows how labeling results improved as features were added incrementally up to the chosen combination.

Figure 5.2 demonstrates recognition scores for the same incremental combination growth, showing that improvement of recognition doesn't necessarily follow labeling improvement.

Though not related to the CoNLL SRL task, performance data were obtained for many of the runs accomplished. Figure 5.3 illustrates performance statistics with increasing feature combination size. All runs were performed with the DS3 data set (262630 exemplars) and the thesis test set (4870 instances).

These performance data indicate that even with this expanded data set, processing times do not grow exponentially (at least to this feature set size). This was a problem in previous versions of AM with much smaller data sets.

Size	Addition	Combination
0		
1	O	O
2	S	SO
3	L	LSO
4	Fvn5	LSOFvn5
5	Tvb	LSOTvbFvn5
6	Pap	LSOTvbFvn5Pap
7	Eap	LSOTvbFvn5PapEap
8	Fap	LSOTvbFvn5PapEapFap
9	Pv2	LSOTvbFvn5Pv2PapEapFap
10	Pva	LSOTvbFvn5Pv2PapEapFapPva
11	Pss	LSOTvbFvn5Pv2PapEapFapPvaPss
12	Pth	LSOTvbFvn5Pv2PapEapFapPvaPssPth
13	Pt	LSOTvbFvn5Pv2PapEapFapPvaPssPthPt
14	Ch	LChSOTvbFvn5Pv2PapEapFapPvaPssPthPt
15	Sb	LChSOTvbSbFvn5Pv2PapEapFapPvaPssPthPt

Table 5.10: Incremental Combination Development

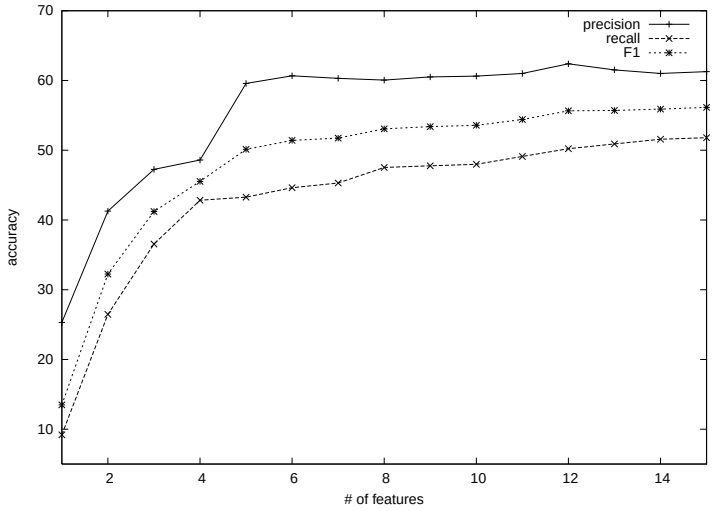

Figure 5.1: Labeling Improvement with Increased # of Features

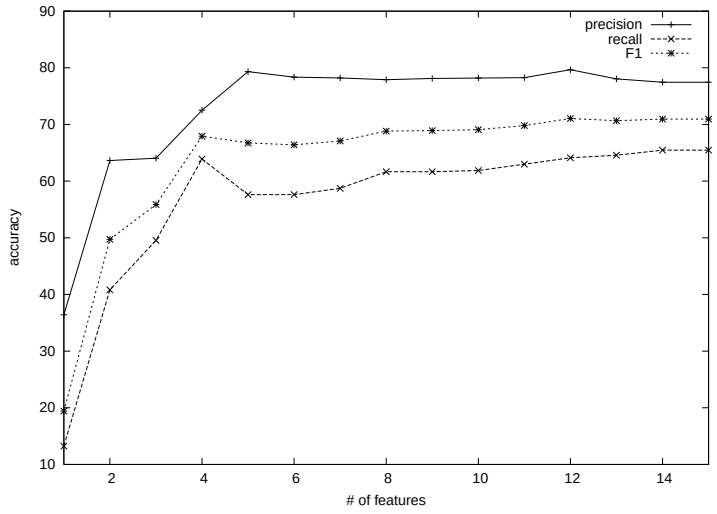

Figure 5.2: Recognition Scores with Increased # of Features

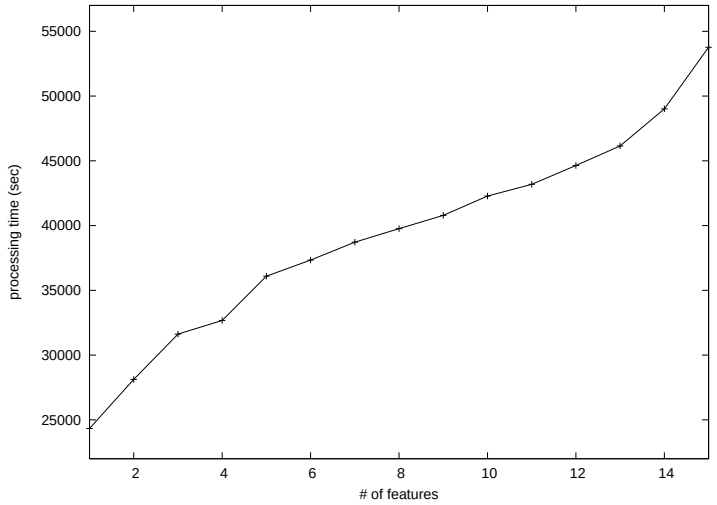

Figure 5.3: Performance Statistics with Increased # of Features

Measure	Thesis Data Test Set	CoNLL Test Set
Instances	4870	148647
Propositions	163	5267
Sentences	73	2416

Table 5.11: Size Comparison of Thesis Data and CoNLL Test Sets

Metric	Thesis Data Test Set	CoNLL Test Set
Precision	61.27	56.39
Recall	51.79	49.16
F1	56.14	52.53

Table 5.12: Labeling Results for Two Test Sets

5.4 Results from CoNLL Test Set

The combination described above was finally used on the CoNLL-2005 test set, in order to have a direct comparison with systems participating in that SRL task. The DS3 data set used throughout system development was used as the data set on this final run. Table 5.4 provides sizes for the thesis data and the final CoNLL test sets.

Table 5.12 compares overall labeling results obtained from the thesis and CoNLL test sets. Figure 5.4 shows the complete set of results using the chosen set of features for the thesis test set, while Figure 5.5 shows the complete set of results (overall and individual arguments), based on the CoNLL evaluation output, for the final test set.

Figure 5.4 shows that 20.25% of propositions in the test set (33 of 163 total) were identified completely correctly in the chosen combination. This represents a great improvement from the preliminary run, where only 6.13% were identified completely correctly.

Though overall, results tended downward with the CoNLL test set, an improvement in AO precision, recall, and F1 measure was found with the larger data set.

```
Number of Sentences    :          73
Number of Propositions :         163
Percentage of perfect props :  20.25

              corr.  excess  missed    prec.    rec.      F1
------------------------------------------------------------
   Overall      231     146     215    61.27   51.79   56.14
----------
        A0       71      52      49    57.72   59.17   58.44
        A1      101      40      54    71.63   65.16   68.24
        A2       10      17      20    37.04   33.33   35.09
        A3        3       5       9    37.50   25.00   30.00
        A4        2       0       2   100.00   50.00   66.67
    AM-ADV        0       2      13     0.00    0.00    0.00
    AM-CAU        0       0       2     0.00    0.00    0.00
    AM-DIR        0       2       1     0.00    0.00    0.00
    AM-DIS        4       1       9    80.00   30.77   44.44
    AM-EXT        0       1       2     0.00    0.00    0.00
    AM-LOC        2       6       9    25.00   18.18   21.05
    AM-MNR        5       4      18    55.56   21.74   31.25
    AM-MOD       13       0       1   100.00   92.86   96.30
    AM-NEG        3       0       0   100.00  100.00  100.00
    AM-PNC        0       2       5     0.00    0.00    0.00
    AM-TMP       11      12      16    47.83   40.74   44.00
      R-A0        3       0       5   100.00   37.50   54.55
      R-A1        2       0       0   100.00  100.00  100.00
  R-AM-LOC        1       1       0    50.00  100.00   66.67
  R-AM-TMP        0       1       0     0.00    0.00    0.00
------------------------------------------------------------
         V      156       7       7    95.71   95.71   95.71
------------------------------------------------------------
------------------------------------------------------------
              corr.  excess  missed    prec.    rec.      F1
 Unlabeled     292      85     154    77.45   65.47   70.96
------------------------------------------------------------
```

Figure 5.4: Complete Set of Results for Optimum Run on Thesis Test Set

```
Number of Sentences    :        2416
Number of Propositions :        5267
Percentage of perfect props :  18.26

              corr.  excess  missed    prec.    rec.      F1
------------------------------------------------------------
   Overall     6920    5352    7157    56.39   49.16   52.53
----------
        A0     2267    1625    1296    58.25   63.63   60.82
        A1     2461    2056    2466    54.48   49.95   52.12
        A2      420     474     690    46.98   37.84   41.92
        A3       46      58     127    44.23   26.59   33.21
        A4       51      69      51    42.50   50.00   45.95
        A5        1       0       4   100.00   20.00   33.33
    AM-ADV       51      76     455    40.16   10.08   16.11
    AM-CAU       10      17      63    37.04   13.70   20.00
    AM-DIR       16      30      69    34.78   18.82   24.43
    AM-DIS      106      71     214    59.89   33.12   42.66
    AM-EXT       13      20      19    39.39   40.62   40.00
    AM-LOC       87     158     276    35.51   23.97   28.62
    AM-MNR       81     148     263    35.37   23.55   28.27
    AM-MOD      472      21      79    95.74   85.66   90.42
    AM-NEG      212       6      18    97.25   92.17   94.64
    AM-PNC       14      35     101    28.57   12.17   17.07
    AM-PRD        0       0       5     0.00    0.00    0.00
    AM-REC        0       0       2     0.00    0.00    0.00
    AM-TMP      377     339     710    52.65   34.68   41.82
      R-A0      150      77      74    66.08   66.96   66.52
      R-A1       50      46     106    52.08   32.05   39.68
      R-A2        2       1      14    66.67   12.50   21.05
      R-A3        0       0       1     0.00    0.00    0.00
      R-A4        0       1       1     0.00    0.00    0.00
  R-AM-ADV        0       0       2     0.00    0.00    0.00
  R-AM-CAU        0       1       4     0.00    0.00    0.00
  R-AM-EXT        0       0       1     0.00    0.00    0.00
  R-AM-LOC        9       3      12    75.00   42.86   54.55
  R-AM-MNR        1       0       5   100.00   16.67   28.57
  R-AM-TMP       23      20      29    53.49   44.23   48.42
------------------------------------------------------------
         V     5079     170     188    96.76   96.43   96.60
------------------------------------------------------------
------------------------------------------------------------
              corr.  excess  missed    prec.    rec.      F1
 Unlabeled     8717    3555    5360    71.03   61.92   66.17
------------------------------------------------------------
```

Figure 5.5: Complete Set of Results for CoNLL Test Set

Metric	CoNLL-2004		CoNLL-2005	
	Kouchnir (2004)	van den Bosch (2004)	Tjong Kim Sang (2005)	Current
Precision	56.86	67.12	70.70	56.39
Recall	49.95	54.46	69.85	49.16
F1	53.18	60.13	70.27	52.53

Table 5.13: Comparative Results for Systems on SRL (Labeling)

5.5 Discussion of Results

We expected to be able to achieve similar results to the MBL systems at CoNLL 2004 and 2005. Table 5.13 provides a comparison of final overall results for the present system and other MBL systems tested at CoNLL.

Though the current values closely match those of Kouchnir (2004), the test set used in 2004 was different so no direct comparison should be assumed. The only system directly comparable (based on test set used) to the current system is that of Tjong Kim Sang et al. (2005).

Results from the current system are lower, but there are a few differences to consider. The systems of van den Bosch et al. (2004) and Tjong Kim Sang et al. (2005) performed other procedures not applied here. van den Bosch et al. (2004) used a couple optimization strategies, one based on classifier stacking, in development. They claim that "both methods avoid errors in sequences of predictions typically made by simple classifiers that are unaware of their previous or subsequent decisions in a sequence" (p. 102). The system of Tjong Kim Sang et al. (2005) also performed optimization strategies not considered here.

Additionally, Carreras and Màrquez (2005) note that the best-performing systems typically combined more than one technique in determining final results. This is true of the system of Tjong Kim Sang et al. (2005), who used MBL algorithms, maximum entropy models, and support vector machines. Predictions from these different machine learing algorithms were combined for final results. However, the results noted above for this system were only for the MBL algorithm, and thus are similar to the current system.

The system of Tjong Kim Sang et al. (2005) also contained a post-processing module that attempted to correct unlikely role assignments. As before, the results reported above were for the system without this post-processing step.

	Kouchnir (2004)	van den Bosch (2004)	Tjong Kim Sang (2005)	Current	Current (w/ opening only)
A0	63.05	70.18	81.73	59.17	68.33
A1	53.22	59.67	71.89	65.16	71.61

Table 5.14: Comparison of Recall Results for A0 and A1 Arguments

It is interesting to compare results for individual argument types between systems as well. The A0 and A1 arguments were the most prevalent within the test set. Recall scores obtained by the final combination (using the thesis test set) were 59.17 for A0 and 65.16 for A1. However, because of information gathered from the 'amcpresults' file, we know that opening of these argument types would result in recall scores of 68.33 and 71.61, if the corresponding closures were identified correctly.

These values compete well with those of previous systems, as seen in Table 5.14. The system of Tjong Kim Sang et al. (2005) didn't report individual values for only the MBL system, so a direct comparison cannot be made. The values reported in Table 5.14 are for their complete system.

The results from the current system are not directly comparable with those of Kouchnir (2004) or van den Bosch et al. (2004) due to different test sets (CoNLL-2004 versus CoNLL-2005). Thus, direct comparisons to other systems are not possible for any of the cases. However, it appears that other systems generally obtained a higher score for A0 than for A1. This is reversed in the current system.

One interesting point occurs in a feature set one size larger (16 total features) than the chosen one. With the additional Cl feature, more openings for A0 and A1 are correctly identified even when overall results are lower. These would respectively result in recall scores of 69.72 and 72.3 for A0 and A1. It is noted that this A1 score (on the current system, assuming opening only) is higher than that of any of the other systems when considering the recognition of argument opening. This includes the system by Tjong Kim Sang et al. (2005), which was built on a number of algorithms and contained a post-processing module for correcting unlikely roles assigned. This indicates that either the features chosen here or the method used here provided better labeling of A1 arguments. A further exploration of this could provide important principles about recognition and labeling of this argument type.

When the CoNLL test set was run, the trend of A0 scores being higher than A1 scores became evident. No information was collected in 'amcpresults' to determine percentage of correct openings as was done on runs using the thesis test set, so this cannot be used to see how results would improve if closures could be identified correctly. However, because the AO F1 score improved from 59.17 to 60.82 (from the thesis data to the CoNLL test set), it is expected that the score (for openings only) would be greater than that reported in the table (68.33). On the other hand, the A1 score would probably decrease as it did going from the thesis data to the CoNLL test set, as it followed the overall trend.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has demonstrated that AM is a viable algorithm for the task of SRL, thus extending the use of this algorithm to a relatively new natural language processing task. In doing so, a completely original system was also created for this task.

This work has also proved that the use of AM in SRL displays similar issues to TiMBL, which was one of the foremost algorithms applied to this task at CoNLL. For example, redundancy in features is an important consideration (similar to findings of van den Bosch et al. (2004) and Tjong Kim Sang et al. (2005)). Feature selection is critical, and the incremental combination development used here was useful for identifying the best feature sets. Additionally, this work showed that it is difficult to both identify and label arguments simultaneously. Without proper argument boundary recognition, labeling is problematic. Kouchnir (2004) used separate modules for each of these two SRL tasks.

The computational algorithm has been exercised greatly in this work, with large quantities of data being used in runs. AM has processed runs containing more than 1/4 million data exemplars (using DS3 data set), along with almost 5000 test instances, on a regular basis. A few runs were also performed with more than 1/2 million exemplars. The final run included more than 1/4 million data exemplars and almost 150,000 test instances. These data sets appear to be the largest ever processed using the AM algorithm. However, these runs are somewhat slow. A run containing only one feature (DS3 data set, thesis test set) took over six hours to complete, while a 17-feature run lasted approximately fifteen hours. The final run with a substantially increased test set finished in a few hours short of 18 days. A new 64-bit version of the program has not been tested here, but it is expected that its use will result in performance improvements.

Results from the present system are not state-of-the-art yet, but are approaching this target. Since results have increased over the course of this work, it is expected that further principled development will improve the system.

6.2 System Improvement

There are additional possibilities for further work that could be completed to improve results, even though the current version allows for a complete evaluation. These suggestions seem to fall within the different steps of the system architecture, and will be divided in that fashion.

6.2.1 Features

Feature selection is critical for improvement performance. Further work could be done in this area to select more appropriate features for the current system.

Additionally, argument recognition is a critical step. The features used to identify argument boundaries went through a variety of revisions. A more complete evaluation of these could be helpful in improving results of the system. It might be useful to employ a method similar to that of Tjong Kim Sang et al. (2005) for recognizing potential arguments. They used:

> "syntactic trees for deriving instances, partly at the constituent level and partly at the word level. On both levels, we edit the data down to only the predicted positive cases of verb-constituent or verb-word pairs exhibiting a verb-argument relation" (p. 229).

This was similar to what was done here but seemed to be more extensively accomplished. Obviously, this limits the maximum obtainable recall as some arguments could be improperly excluded. However, this also significantly lowers the total number of instances to be tested and thus saves significant amounts of time in testing. This was a critical factor in the present study, as feature combination testing took significant amounts of time with larger cases. Carrying out complete incremental development as described previously was limited by this, so redundancy might still be present within the chosen combination. Implementing something akin to what is described by Tjong Kim Sang et al. (2005) could be helpful for large-scale testing.

6.2.2 Running AM

Some other potential improvements involve multiple runs of AM. First, classifier stacking could be employed, such as done by van den Bosch et al. (2004). Another

possibility for improvement involves the use of two AM runs (or modules as discussed by Kouchnir (2004)). The first run would seek only to recognize arguments, and afterwards a second run would be used to take those arguments and label them. This could be helpful here, due to the potential for features that help in recognition to act redundantly when trying to label the argument. Evidence for this is provided by the fact that labeling does not necessarily increase as recognition does.

Future work could involve identifying two sets of features, similar to what was done by Kouchnir (2004). Recognized arguments from a first set of features would then be piped to a second AM run with features specifically designed for distinguishing argument types for labeling. labeling. This set would be designed to take recognized arguments and

6.2.3 Parsing of AM Results

Another potential change involves the parsing of the 'amcpresults' file. The original code for this picks the result with the highest percentage. Usually this is the '*' (where no argument boundaries are present, indicates within an argument or outside an argument) but in many cases the '*' overpowers all other outcomes in AM. For this reason, different methods for reading the 'amcpresults' file were tried to ignore (to some extent) that item. In one technique, if the percentage guessed toward '*' is under a certain threshold, it would be ignored. A large range of values for this threshold was tested and it did create differences in the results after complete evaluation.

A further implementation of this, which I think is best, is to determine whether a certain type of variable (e.g. opening an argument) has a greater total percentage than the '*' outcome. This would be the case where AM guessed that a certain line should have an outcome of '(A0*' or '(A1*', or if it was just a '*'. If '*' were to have a greater percentage than either of the others it would be taken as the outcome. However, if the total percentage of the open arguments exceeds the percentage of '*' then we would say that this line opens an argument. We would take the larger percentage of these two for the guess. These discounting methods could be further looked at for purposes of improvement.

6.2.4 Post-Processing and Cleaning of Parsed Results

Another possible improvement relates to the observation above. Multiple occurrences of the same argument frequently appeared in close range. It would be useful to identify whether such cases should be treated as a complete argument (spanning

both of the same identified arguments) or whether they should be considered separate arguments (where one of the identified ones is incorrect). A post-processing step (added to the cleaning of AM results) could be used to correct these and other potential problems.

Further suggestions by van den Bosch et al. (2004) could improve output from the classifier system. They noted that the output frequency showed multiple occurrences of the same argument (A0-A5) within the same sentence. This was a rare case in the data, so only one of each type of argument was allowed per sentence. Upon examination of results from the AM system, this same problem is apparent. By correcting this issue within a post-processing mechanism, results could be improved.

Tjong Kim Sang et al. (2005) discusses a post-processing module that was used to correct errors made by their base system. This attempts to identify and correct "unlikely and impossible relation arguments, such as assigning two indirect objects to a verb where only one is possible" (p. 230). The module works by transforming the guessed propositions until they are in line with certain constraints, such as the one proposed by van den Bosch et al. (2004) (described in above quote).

These methods could be implemented in the current system to improve output, though at the moment the only purpose for cleaning the results is to ensure that particular argument boundaries are present. Without proper boundaries the evaluation code fails, so this is a critical step. Another possible method for cleaning the results comes from the previous discussion on the utility of the tool PropBank. If there were good information on typical order argument, this could be implemented in a post-processing mechanism to fix orders that didn't appear to be the norm.

6.3 Extension to Other Languages

Much of the work done for English was intended to prepare the way to do SRL for the Arabic language at SemEval 2007. The task for Arabic at this conference had a similar format to the English SRL tasks at CoNLL 2004 and 2005. In the process of doing the work for English as well as some preliminary ideas for Arabic, a number of Perl scripts have been developed by the author to extract features from the Penn TreeBank formatted sentences. These were done using the Lingua Treebank Perl tool[1]. Some of these features (e.g. lexical items, POS tags) were given for the CoNLL tasks, but as mentioned previously, these scripts were developed for a possible use with the Arabic TreeBank.

[1] http://search.cpan.org/~kahn/Lingua-Treebank-0.13/

The current scripts that have been tested on the Arabic Treebank are able to obtain lexical items, POS tags (both root word as well as complete tag given from the Treebank), affixes and features for nouns and adjectives, and verbal information (including aspect and features). Additionally, scripts are available to obtain trigrams of some of these items.

It is expected that the work accomplished for English SRL using an AM paradigm can be extended to Arabic, and some of the background work has already been prepared. There are a number of differences between English and Arabic that would be important to consider, such as word order and the realization of case.

Bibliography

Ahn, D., Fissaha, S., Jijkoun, V., and de Rijke, M. (2004). The University of Amsterdam at Senseval-3: Semantic roles and logic forms. In Mihalcea, R. and Edmonds, P., editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 49–53, Barcelona, Spain. Association for Computational Linguistics.

Baayen, R. H. and Moscoso del Prado Martín, F. (2005). Semantic density and past-tense formation in three Germanic languages. *Language*, 81(3):666–698.

Babko-Malaya, O. (2005a). Guidelines for Propbank framers. available at: http://verbs.colorado.edu/∼mpalmer/projects/ace/FramingGuidelines.pdf.

Babko-Malaya, O. (2005b). Propbank annotation guidelines. available at: http://verbs.colorado.edu/∼mpalmer/projects/ace/PBguidelines.pdf.

Buchholz, S. (2002). *Memory-Based Grammatical Relation Finding*. PhD thesis, Tilburg University.

Carreras, X. and Màrquez, L. (2003). Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP) 2003*, pages 205–216, Borovets, Bulgaria. Association for Computational Linguistics.

Carreras, X. and Màrquez, L. (2004). Introduction to the CoNLL-2004 shared task: Semantic role labeling. In Ng, H. T. and Riloff, E., editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 89–97, Boston, MA. Association for Computational Linguistics.

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, MI. Association for Computational Linguistics.

Caruana, R. and Freitag, D. (1994). Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36, New Brunswick, NJ. Morgan Kaufman.

Chen, J. and Rambow, O. (2003). Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of the 2003 Conference on Empirical Methods on Natural Language Processing (EMNLP-2003)*, pages 41–48, Sapporo, Japan. Association for Computational Linguistics.

Chieu, H. L. and Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pages 160–163, Edmonton, Canada. Association for Computational Linguistics.

Collins, M. J. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Daelemans, W. and van den Bosch, A. (2005). *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, England.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2003). TiMBL: Tilburg Memory-Based Learner, Version 5.0, Reference Guide. Technical report, Tilburg University. ILK Technical Report 03-08.

Eddington, D. and Lonsdale, D. (2007). Analogical modeling of language: An update. In *19th European Summer School in Logic, Language and Information (ESSLLI 2007): Workshop on Exemplar-Based Models of Language Modeling and Use*, Dublin, Ireland. The Association of Logic, Language and Information (FoLLI).

Eisner, J. (1996). An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania.

Fillmore, C. J., Wooters, C., and Baker, C. F. (2001). Building a large lexical databank which provides deep semantics. In Tsou, B. and Kwong, O., editors, *Proceedings of the Fifteenth Pacific Asian Conference on Language, Information and Computation*, pages 3–26, Hong Kong, China. City University of Hong Kong.

Gildea, D. and Hockenmaier, J. (2003). Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the 2003 Conference on Empirical Methods*

on Natural Language Processing (EMNLP-2003), pages 57–64, Sapporo, Japan. Association for Computational Linguistics.

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

Gildea, D. and Palmer, M. (2002). The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–246, Philadelphia, PA. Association for Computational Linguistics.

Giménez, J. and Màrquez, L. (2003). Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP) 2003*, pages 153–163, Borovets, Bulgaria. Association for Computational Linguistics.

Hammond, M. T. (2003). *Programming for Linguists: Perl for Language Researchers*. Blackwell, Malden, MA.

Hendrickx, I. and van den Bosch, A. (2003). Memory-based one-step named-entity recognition: Effects of seed list features, classifier stacking, and unannotated data. In *Proceedings of the Seventh Conference on Computational Natural Language Learning (CoNLL-2003)*, pages 176–179. Association for Computational Linguistics.

Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas de Gran Canaria, Spain. European Language Resources Association.

Kingsbury, P., Palmer, M., and Marcus, M. (2002). Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference*, San Diego, CA.

Kouchnir, B. (2004). A memory-based approach for semantic role labeling. In Ng, H. T. and Riloff, E., editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 118–121, Boston, MA. Association for Computational Linguistics.

Lin, J.-H. and Vitter, J. S. (1994). A theory for memory-based learning. *Machine Learning*, 17:1–26.

Litkowski, K. (2004). Senseval-3 task: Automatic labeling of semantic roles. In Mihalcea, R. and Edmonds, P., editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 9–12, Barcelona, Spain. Association for Computational Linguistics.

Lonsdale, D. (2002). Data files for Analogical Modeling. In Skousen, R., Lonsdale, D., and Parkinson, D., editors, *Analogical Modeling*, volume 10 of *Human Cognitive Processing*, pages 349–363. John Benjamins Publishing Company, Amsterdam, The Netherlands.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Parkinson, D. B. (2002). Data files for Analogical Modeling. In Skousen, R., Lonsdale, D., and Parkinson, D., editors, *Analogical Modeling*, volume 10 of *Human Cognitive Processing*, pages 365–383. John Benjamins Publishing Company, Amsterdam, The Netherlands.

Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., and Jurafsky, D. (2003). Support vector learning for semantic argument classification. Technical Report TR-CSLR-2003-03, University of Colorado. Center for Spoken Language Research.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 233–240, Boston, MA.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2005). Semantic role labeling using different syntactic views. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 581–588. Association for Computational Linguistics.

Skousen, R. (1989). *Analogical Modeling of Language*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Skousen, R. (2003). Analogical modeling: Exemplars, rules, and quantum mechanics. In *Proceedings of the 29th Annual Meeting of the Berkeley Linguistics Society*, pages 425–439. Berkeley Linguistics Society.

Skousen, R., Lonsdale, D., and Parkinson, D., editors (2002). *Analogical Modeling: An Exemplar-Based Approach to Language*, volume 10 of *Human Cognitive Processing*. John Benjamins Publishing Company, Amsterdam, The Netherlands.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press, Cambridge, MA.

Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 8–15, Sapporo, Japan. Association for Computational Linguistics.

Thompson, C. A., Levy, R., and Manning, C. (2003). A generative model for FrameNet semantic role labeling. In *Proceedings of the Fourteenth European Conference on Machine Learning*, pages 397–408, Croatia. Springer.

Tjong Kim Sang, E. (2001). Memory-based clause identification. In *Proceedings of the Fifth Conference on Computational Natural Language Learning (CoNLL-2001)*, pages 67–69. Association for Computational Linguistics.

Tjong Kim Sang, E., Canisius, S., van den Bosch, A., and Bogers, T. (2005). Applying spelling error correction techniques for improving semantic role labelling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 229–232, Ann Arbor, MI. Association for Computational Linguistics.

van den Bosch, A., Canisius, S., Daelemans, W., Hendrickx, I., and Tjong Kim Sang, E. (2004). Memory-based semantic role labeling: Optimizing features, algorithm, and output. In Ng, H. T. and Riloff, E., editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 102–105, Boston, MA. Association for Computational Linguistics.

van Rijsbergen, C. V. (1979). *Information Retrieval*. Butterworths, London, England.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods on Natural Language Processing (EMNLP-2004)*, pages 88–94, Barcelona, Spain. Association for Computational Linguistics.