The Effect of Selectional Preferences on Semantic Role Labeling

by

Andrew Cleburne Young

Special Honors in the Department of Linguistics

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2009

Katrin Erk

Department of Linguistics

Supervising Professor

Dedicated to my wife Laine and my daughter Kaitlyn.

The Effect of Selectional Preferences on Semantic Role Labeling

Andrew Cleburne Young, B.A.
The University of Texas at Austin, 2009

Supervisor: Katrin Erk

The aim of this paper is to explore the effect that selectional preferences have on the accuracy of semantic role labeling systems in a controlled environment where other unseen factors will not affect the outcome. This will provide evidence either for or against the inclusion of selectional preferences as a standard feature of semantic role labeling systems in the future.

Acknowledgments

I would like to thank my adviser, Katrin Erk, for helping me with this thesis over the past year despite my hectic work schedule. Without her help and motivation this document wouldn't be possible. I would also like to thank Megan Crowhurst, Stephen Wechsler, Jason Baldridge, Robert King, and the rest of the faculty in the linguistics department at the University of Texas at Austin for sharing their love of linguistics with me. I would like to thank Frederick Hoyt as well for getting me interested in linguistics in the first place.

On a more personal note, I would like to thank my wife Laine for supporting me in my studies. Without her I probably would have given up on this thesis a long time ago. I also want to thank my parents and my in-laws for always believing in me. And last, but certainly not least, I want to thank my grandfather, James C. Houston, for introducing me to computer programming in the summer of 1989 and starting me down the path I'm on today.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Semantic Roles and Frame Semantics

Semantic roles[1] represent the underlying relationship between the main verb in a clause and its participants. The concept was first introduced by Charles Fillmore in his paper "The Case For Case"[6], in which he argued that the notion of semantic "cases" (or roles) was central to all languages. Whereas the surface representation of a clause defines a syntactic role for a given word that is perhaps specific to that language – such as "Subject" or "Object" – the deeper meaning of the clause defines a semantic role which is universal – such as "Agent" or "Goal." The word which fills a given semantic role of a verb might appear in more than one syntactic role depending on the structure of the clause. For example, let's look at the following sentences:

(1) John ate the apple at the park.

(2) The apple was eaten at the park by John.

In (1) the syntactic role of "Subject" is filled by the noun phrase "John" and the syntactic role of "Object" is filled by the noun phrase "the apple." How-

[1] Also known as "thematic roles" or "cases".

ever, in (2) the “Subject” role is filled by “the apple” and there is no “Object” role. Both sentences, however, describe the same event. The difference is that (1) is expressed in the active voice and (2) is expressed in the passive voice. Because they both describe the same event, the person who is doing the eating (the “Agent”) and the thing being eaten (the “Theme”) don’t change. In both examples, “John” fills the semantic role of “Agent” and “the apple” fills the semantic role of “Theme.”

The concept of semantic roles is closely linked with the concept of frame semantics, which was introduced by Charles Fillmore as a further development of his ideas on case grammar. [7] The underlying principle of frame semantics is that one cannot understand the meaning of a word without having access to all the essential knowledge that relates to that word. For example, one cannot understand the meaning of “sell” without understanding how commercial transactions work, what money is, what a buyer and seller are, and so on. Thus, the word “sell” evokes a “semantic frame” of meaning relating to the specific concept to which it refers (the selling of goods).

1.2 Automatic Semantic Role Labeling

Automatic semantic role labeling is the task of determining which constituents in a sentence are semantic arguments for a given predicate and then determining the appropriate role for each of those arguments. [8] Labeling the constituents in a sentence with their semantic roles provides important data for systems which require an understanding of the semantic meaning of a given text. Current common uses of this information include automated question answering and information extraction.

As an example, let's look at a hypothetical search engine. For this example, let's assume that the search engine has indexed documents containing the following sentences (among others):

(3) UT Austin is located 20 miles from downtown Round Rock.

(4) You can ride the bus to UT Austin.

(5) I often drive to UT Austin from Round Rock to watch the Longhorns play football.

Now let's say that a user submits the following query to our search engine:

(6) How can I get to UT Austin from Round Rock?

A naïve search engine will gather a list of documents which contain those words and rank them so that documents with more matching words are returned first. This sort of search engine would likely return document (5) as the best match, followed by document (3) and then document (4). However, document (5) and document (3) don't answer the user's question[2]. A more intuitive search engine could look at the semantic representation of the

[2]Although document (5) does indirectly mention a possible means of getting to the destination.

question to find out what the user is actually looking for before determining the ranking of its results. This can be accomplished by assigning semantic roles to both the search query and the documents that have been indexed.

A search engine that assigns semantic roles to its indexed documents might assign roles like these[3]:

(7)
$$\begin{bmatrix} \textsc{Frame} & \text{`Being\_located'} \\ \textsc{Predicate} & \text{`located'} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Theme} & \text{`UT Austin'} \\ \textsc{Location} & \text{`20 miles from downtown Round Rock'} \end{bmatrix} \end{bmatrix}$$

(8) a.
$$\begin{bmatrix} \textsc{Frame} & \text{`Capability'} \\ \textsc{Predicate} & \text{`can'} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Entity} & \text{`You'} \\ \textsc{Event} & \text{`ride the bus to UT Austin'} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textsc{Frame} & \text{`Motion\_directional'} \\ \textsc{Predicate} & \text{`ride'} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Theme} & \text{`You'} \\ \textsc{Carrier} & \text{`the bus'} \\ \textsc{Goal} & \text{`to UT Austin'} \end{bmatrix} \end{bmatrix}$$

[3]These roles are taken from the FrameNet corpus.

(9)
$$\begin{bmatrix} \textsc{Frame} & \text{‘Motion\_directional’} \\ \textsc{Predicate} & \text{‘drive’} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Theme} & \text{‘I’} \\ \textsc{Goal} & \text{‘to UT Austin’} \\ \textsc{Source} & \text{‘from Round Rock’} \\ \textsc{Purpose} & \text{‘to watch the Longhorns play football’} \end{bmatrix} \end{bmatrix}$$

It could then assign roles such as these to the question:

(10) a.
$$\begin{bmatrix} \textsc{Frame} & \text{‘Capability’} \\ \textsc{Predicate} & \text{‘can’} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Entity} & \text{‘I’} \\ \textsc{Event} & \text{‘get to UT Austin from Round Rock’} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textsc{Frame} & \text{‘Motion\_directional’} \\ \textsc{Predicate} & \text{‘get’} \\ \textsc{Roles} & \begin{bmatrix} \textsc{Entity} & \text{‘I’} \\ \textsc{Goal} & \text{‘to UT Austin’} \\ \textsc{Source} & \text{‘from Round Rock’} \end{bmatrix} \end{bmatrix}$$

From this, the search engine can determine that what the user really wants to know are the modes of transportation that are available. It could then look for sentences which contain the “Capability” frame with an “Event” role which contains the “Motion_directional” frame with a “Goal” role containing “UT Austin”. This would rank document (4) first, followed by document (5) and then document (3). Alternatively, the search engine could assume that the user was trying to find the various means of transportation “to UT Austin” and could generate the list of constituents which are assigned to the “Carrier” role in “Motion_directional” frames that also contain a “Goal” role of “to UT Austin”. It could then display those constituents to the user along with the search results.

1.3 Headwords

In order to decide which roles to assign to which words, many systems currently use the idea of headwords. A headword is the direct child of a complex constituent that has the same syntactic category as the parent constituent. For example, let’s take a look at the following sentences:

(11) John eats apples.

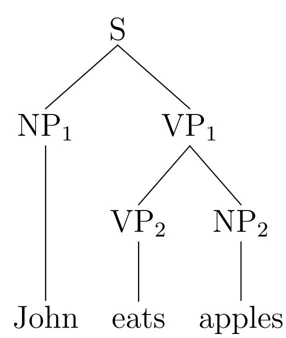

Figure 1.1: Syntax Tree of (11)

(12) John sometimes eats red apples.

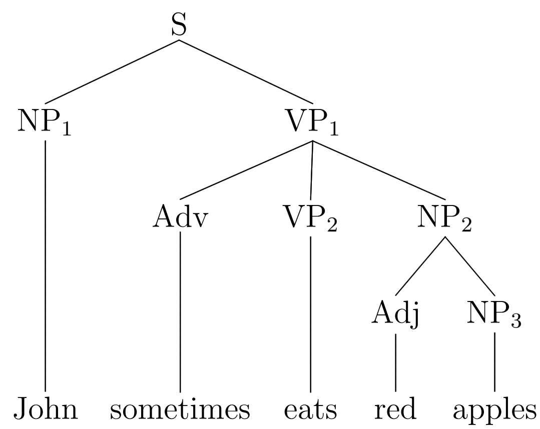

Figure 1.2: Syntax Tree of (12)

In (11) there are three simple constituents (NP$_1$, VP$_2$, and NP$_2$) and one complex constituent (VP$_1$). In this simple case, the headwords for each of the three simple constituents are "John", "eats", and "apples" respectively. The headword for the complex constituent VP$_1$ is taken from the direct descendant of VP$_1$ that is also a VP, namely VP$_2$. This means that the headword for VP$_1$ is "eats". If we embellish this sentence a little bit we get something like (12). Now NP$_2$ is a complex constituent instead of a simple constituent. However, since NP$_2$ gets its headword from the direct descendant with the same category, namely NP$_3$, the headword for NP$_2$ is still "apples". This simple concept allows current systems to decide semantic roles based on what roles they have previously seen assigned to a given constituent, without getting confused by slight changes in the composition of the constituent. To see this in action, let's assume our system sees the following sentence [10]:

(13) I want to eat someplace that's close to campus.

There are two possible interpretations of this sentence. The first interpretation is that “someplace that’s close to campus” is the location where the eating will take place, and therefore fills the “Place” role. The second interpretation is that “someplace that’s close to campus” is the thing that will be eaten, and therefore fills the “Theme” role. Native speakers immediately realize that the first interpretation is the correct one, but a computer has a much harder time determining that.

For example, let’s assume that we have a machine learning system that uses the headword of the constituent following the verb to determine what semantic role to assign to that constituent. First we train the system on the following data:

Verb	**Constituent**	**Role**
eat	apple	Theme
eat	toast	Theme
eat	lunch	Theme

Now let’s assume that we ask this system to label the semantic role of the phrase “someplace that’s close to campus” from(13). The system hasn’t seen the headword “someplace” before, but so far all of the constituents following the verb “eat” have had the “Theme” role, so that is what it will assign to the previously unseen constituent. This would give us the second interpretation for (13), which we know is incorrect.

So how do we fix this? Well, one way is by just giving the system more training data. If the system had already seen a sentence like “I want to eat someplace else”, where “someplace” was assigned the “Place” role, then it would know that when the constituent following “eat” has “someplace” as its headword, then it should assign the “Place” role. This would give us the correct interpretation of (13), but the next time the system saw an unknown noun as a headword it would once again fall back to the “Theme” role. This will almost certainly happen frequently because it is unlikely that all possible headwords will be present in the training data, regardless of its size. The real problem is that seen headwords will be quite sparse in any annotated training corpus compared to the non-annotated text that the system will be asked to work with. This problem is compounded by the fact that manual annotation of a corpus is an expensive and time consuming venture.

1.4 Selectional Preferences

What if instead of just giving the system more training data we made the system smarter by giving it more information about how words are related? This would allow the system to do more with headwords that it hasn’t seen in its training data. For instance, a useful clue in determining which interpretation of (13) is correct is the fact that the “Theme” role of the predicate “eat” is usually filled by something edible. If the system was given information on how words are related, it might discover that “someplace” doesn’t fall into the category of edible things and therefore it would be highly unlikely for it to fill the “Theme” role. This is an example of selectional preferences.

Selectional preferences[4] are constraints that a predicate places on what sorts of words can fill in for its various semantic arguments. [20] There's nothing syntacticly wrong with the second interpretation of (13), but it doesn't make sense semantically because of the selection preferences of "eat"[5].

Due to the sparseness of seen headwords, it would be reasonable to expect that selectional preferences would be a standard feature of modern semantic role labeling systems. However, although some people have done work on selectional preferences, they are still a relatively rarely used feature. This is due mainly to the fact that previous research has included many other features in the labeling process.

The aim of this paper is to explore the effect that selectional preferences have on the accuracy of semantic role labeling systems in a controlled environment where other unseen factors will not affect the outcome. This will provide evidence either for or against the inclusion of selectional preferences as a standard feature of semantic role labeling systems in the future.

[4] Selectional preferences are sometimes also called selectional restrictions.

[5] Unless, of course, the Agent of "to eat" is Godzilla!

Chapter 2

Related Work

This chapter discusses previous research related to semantic role labeling and selectional preferences.

2.1 Semantic Role Labeling

Even though there has been a lot of fundamental research on the task of semantic role labeling, starting with Gildea and Jurafsky [8] and including shared tasks at CoNLL [3, 4] and SENSEVAL 3 [13], previous approaches have either focused largely on headwords or have used selectional preferences as only one of many features.

2.2 Selectional Preferences

Resnik [20] was the first to define a formal computational model for selectional preferences. His goal was to model the selectional preferences of predicates for given arguments with as simple a model as possible. His model uses an information theoretic approach and the idea of word classes – groups of related words – to generalize the results beyond the specific verb/noun pairs that are seen during training.

The model generates two values. The first is the *selectional preference strength* of a given predicate. This number represents how strongly a predicate requires certain word classes as arguments, but doesn't specify what those word classes are. The second is the *selectional association* between a predicate and a given word class.

The selectional preference strength $S(p)$ of a predicate p is computed by first computing two probability distributions and then finding the difference between them[1]:

$$S(p) = D\left(P(C|p)||P(C)\right) = \sum_{c \in C} P(c|p) log \frac{P(c|p)}{P(c)}$$

where $P(C)$ is the overall distribution of the set of all noun classes C and $P(C|p)$ is the probability distribution of the set of all noun classes C paired with the given predicate p.

The selectional association $A(p, c)$ between a predicate p and a word class c is calculated by finding the total amount that c contributes to $S(p)$:

$$A(p, c) = \frac{P(c|p) log \frac{P(c|p)}{P(c)}}{S(p)}$$

The association strength $A(p, n)$ between a predicate p and a noun n is then calculated as the largest association strength of all of the word classes

[1]This is called the Kullback-Leibler (or KL) divergence, which measures the difference between the entropy – or uncertainly – of two probability distributions.

the noun belongs to:

$$A(p,n) = \max_{c \in classes(n)} A(p,c)$$

In Resnik's model the WordNet hierarchy is used to generate the list of classes by assigning a given word to classes for each of its possible meanings and to classes for each of the hypernyms of those meanings. During training, each verb/noun pair adds a partial count to each of the word classes that the noun belongs to. The partial count is inversely proportional to the total number of word classes for that noun:

$$count(v,n) = \frac{1}{|classes(n)|} freq(v,n)$$

where v is a verb, n is a noun, $|classes(n)|$ is the number of word classes to which n belongs, and $freq(v,n)$ is the number of times n was paired with v during training.

These counts are then used to estimate the joint probability $P(v,c)$:

$$P(v,c) = \frac{1}{N} \sum_{n \in words(c)} count(v,n)$$

where c is a word class, $words(c)$ is the set of all nouns in c, and N is the total number of verb/noun pairs seen during training for verb v. The joint probability $P(v,c)$ can then be used to estimate $P(c|v)$ so that selectional preference strengths and selectional associations can be calculated for verb/noun pairs.

2.2.1 Example

Let's assume that we have the following words and word classes:

Word	Word Classes
soup	food, inanimate object
dog	pet, animal, animate object
rabbit	pet, animal, food, animate object
John	person, animate object
lunch	food, event, inanimate object
cafeteria	place, inanimate object

Now let's say that during training the model sees the following verb/noun pairs: "eats/soup", "walks/dog", and "eats/rabbit". The model will count these verb/noun pairs as follows:

- When the model sees the first pair - "eats/soup" - it will add 0.50 to "eats/food" and 0.50 to "eats/inanimate object".
- When it sees the second pair - "walks/dog" - it will add 0.33 to "walks/pet", 0.33 to "walks/animal", and 0.33 to "walks/animate object".
- When it sees the last pair - "eats/rabbit" - it will add 0.25 to "eats/pet", 0.25 to "eats/animal", 0.25 to "eats/food", and 0.25 to "eats/animate object".

After training, the model will contain the following counts:

Class	eats	walks
food	0.75	0.00
inanimate object	0.50	0.00
animate object	0.25	0.33
pet	0.25	0.33
animal	0.25	0.33
person	0.00	0.00
event	0.00	0.00
place	0.00	0.00

These counts are then used to estimate the joint probability distribution $P(v, c)$ using the method detailed above. If we assume an equal distribution of word classes, then we get the following probability distributions:

Class	$P(c)$	$P(c\|eats)$	$P(c\|walks)$
food	0.125	0.375	0.000
inanimate object	0.125	0.250	0.000
animate object	0.125	0.125	0.333
pet	0.125	0.125	0.333
animal	0.125	0.125	0.333
person	0.125	0.000	0.000
event	0.125	0.000	0.000

Using these probability distributions, the model calculates the following selectional preference strengths $S(v)$ and selectional associations $A(v, c)$:

	$eats$	$walks$
$S(v)$	0.585	0.981
$A(v, food)$	0.704	0.000
$A(v, inanimateobject)$	0.296	0.000
$A(v, animateobject)$	0.000	0.333
$A(v, pet)$	0.000	0.333
$A(v, animal)$	0.000	0.333
$A(v, person)$	0.000	0.000
$A(v, event)$	0.000	0.000

Now let's assume that we're using this model to label the words in the sentence "John eats lunch with his dog."[2] Assuming that we know the part of speech of each word in the sentence, we could ask the model to give us the association strength $A(v, n)$ of each of the following verb/noun pairs: "eats/John", "eats/lunch", and "eats/dog".

These are the strengths that would be returned by the model:

Pair	Strength	Class
eats/John	0.000	person or animate object
eats/lunch	0.704	food
eats/dog	0.000	pet, animal, or animate object

This means that out of the three possible choices,"lunch" is most likely to be the argument of "eats" based on the verb's selectional preferences. We can also determine that this version of "lunch" refers to a type of food and not to an event or an inanimate object.

2.2.2 Other Approaches

An alternative approach is to define association strength directly as $P(n|v)$ without relying on the concept of word classes or a predefined word hierarchy. There are a number of ways to do this, such as clustering or computing the similarity of nouns. [12]

In Rooth et al. [22] EM clustering is used to create clusters based on the syntactic role a given noun fills in the training data. Although the model

[2]This also assumes that we're not using the syntactic structure of the sentence as part of our analysis.

uses the idea of word classes (ie. clusters), the classes are discovered during training instead of being prescribed before hand. The model produces a joint probability distribution $p(v, n)$ for all verb/noun combinations that is used directly as the association strength.

In Erk [5] a vector space model is used to compute the similarity of a given word to the words that fill a given role in the training data. This produces a similarity value – $sim(w_0, w)$ – that is used to compute an association strength directly. Erk's model is described in more detail in the next section.

2.3 Word Similarity and Vector Spaces

Erk [5] describes a simple similarity based model for computing selectional preferences. Her model makes use of two corpora, a primary corpus and a generalization corpus. The primary corpus is used to extract tuples of a predicate, an argument position, and a headword. The generalization corpus is then used to compute a vector space with vectors representing the meaning of each of the seen headwords based on the environments in which they are found in the generalization corpus. Once trained, the model computes the selectional preference S of an argument r_p of a predicate p as a weighted sum of similarities:

$$S_{r_p}(w_o) = \sum_{w \in Seen(r_p)} sim(w_0, w) \cdot wt_{r_p}(w)$$

where $Seen(r_p)$ is the set of seen headwords for the argument r_p of predicate p and $wt_{r_p}(w)$ is the weight given to word w for r_p of p.

To do this it first computes a vector for the new headword using the generalization corpus. Next it computes the similarity[3] between this new vector and the vectors for each of the headwords seen in the same argument position for the same predicate in the training corpus. Finally it computes the weighted sum[4] of those similarities, which is the value returned by the model.

[3]This can be computed using several different methods, such as the cosine of the angle between the two vectors.

[4]Erk's paper uses several different weighting methods, but finds that the weighting method has little effect on the accuracy of the model.

Chapter 3

Method

Current research in semantic role labeling has focused on using machine learning systems to label semantic roles based on a large number of features, only one of which is selectional preferences. Because of this, the existing research hasn't determined to what degree selectional preferences help or hinder the ability of these systems to make good labeling choices. In order to determine if selectional preferences improve the performance of a semantic role labeling system, it is necessary to to test their effectiveness in a "pure" setting without many additional features. If it is found that they do improve the performance of semantic role labeling systems, then further research can be done to determine the best way to integrate them into a larger system.

For my experiment, I started by training a machine learning system to label constituents with their semantic roles using a very small benchmark feature set which included only the headword lemma, the predicate, the syntactic role, the part of speech, and the semantic role label. This gave me a benchmark (or control) value. Then I created another dataset which included information about the selectional preferences of the predicate and compared the performance of this system with the performance of the benchmark system.

3.1 Semantic Role Labeling

Semantic role labeling is usually divided into two steps:

1. Decide which constituents should be considered arguments of a given predicate.

2. Decide which roles should be assigned to those arguments.

My experiment focused on the second step because it has been argued that semantic features are more important for the second step than the first step. [18]

3.1.1 Data

There were two corpora available that contained annotations on semantic roles: PropBank [16] and FrameNet [1]. PropBank is a corpus annotated with verbal propositions and their arguments. FrameNet uses the concept of semantic frames where a semantic frame is used to describe an object, state or event. After reviewing both of these corpora, I decided to use the FrameNet corpus for my experiment because FrameNet provides named roles such as "Agent", "Degree", and "Means" whereas PropBank uses enumerated arguments such as "Arg0", "Arg1", and "Arg2". FrameNet's named roles more clearly demonstrate the relationship between words that fill the same role for different predicates. The FrameNet corpus provided me with 133,530 annotated sentences to work with, containing more than 2,970,000 words in total.

3.1.2 Parameters

In order to make the experiment as clean as possible, I needed to be sure that the only variable that changed between the benchmark dataset and the expanded dataset was the addition of selectional preference data. To accomplish this the number of additional features required by the semantic role labeling system was reduced to four: the lemma of the predicate, the lemma of the argument, the argument's part of speech, and the argument's syntactic role. The labeling task was also defined to limit ambiguity by limiting the training and testing data so as to exclude complicated grammatical structures that might cause ambiguities in a predicate's argument structure. Only direct dependants[1] of the predicate were used as semantic role candidates.

3.2 Selectional Preferences Model

To determine the selectional preferences of a given predicate, I used a vector space model similar to the one proposed by Erk [5]. This allowed me to create a highly dimensional vector space with a vector for every word in the training and testing corpora. The direction that the vector pointed was based on its relation to other words in the corpora. Once I had these vectors, I could then compute vectors which represented the average direction of all the words that filled a given role. The process by which this was done is covered in more detail in §3.2.2.

[1] In the experiments described below, we use dependency analyses instead of constituent structure.

3.2.1 Data

Because it is best to have as much data as possible when building a vector space model, I used the British National Corpus for this purpose.

3.2.2 Implementation

The selectional preferences model was implemented using a semantic vector space. The following steps were taken in order to generate the semantic vector space:

1. A list of all of the words in the FrameNet corpus which had been assigned any semantic role was generated. Let U represent this list of words.

2. The Minipar parser[2] - a syntactic parser which uses the concept of Dependency Grammar [14] - was used to create a parsed version of the British National Corpus.

3. A semantic vector space was created using the DependencyVectors[3] software package [15]:

   - The list of words in U and the dependency parsed version of the British National Corpus were given to the software as input.
   - The output of the software was a semantic vector space which included vectors for all of the words in U. Let V represent this vector space.

[2] http://www.cs.ualberta.ca/ lindek/minipar.htm
[3] http://www.nlpado.de/ sebastian/dv.html

Once the vector space was generated, the FrameNet data was divided into a training set, a development set, and a test set. This was done by randomly assigning 1/5 of the FrameNet data to the development set, 1/5 to the test set, and the remaining 3/5 to the training set. The training set was used to train the selectional preferences model and the development set was used to test the effectiveness of the model during development. The test set was put aside for testing the effectiveness of the model after it was fully developed and trained. This method assures that the model doesn't over fit the test data used in the final analysis.

Algorithm 1 Generating the Selectional Preferences Vector Space

```
F ← {the list of all semantic frames in the training data}
V ← {a mapping of lemmas to vectors, as generated by DependencyVectors}
SP ← [] {SP will be a mapping of roles to vectors.}
for f in F do {f is the current semantic frame.}
  R ← {the list of all semantic roles in f}
  for r in R do {r is the current semantic role.}
    L ← {the list of all lemmas in f which are assigned the role r in the training
    data.}
    sp_r ← 0 {sp_r is a vector is vector space V.}
    for l in L do {l is the current lemma.}
      v_l ← V[l] {The vector for lemma l from vector space V.}
      sp_r ← sp_r + v_l
      {sp_r represents the average semantic meaning of all the words which fill
      the role r in the semantic frame f, given the training data. In other words,
      it represents the selectional preference of the frame f towards the semantic
      role r.}
    end for
    SP[r] ← sp_r
  end for
end for
return SP
```

Algorithm 2 Training the Selectional Preferences Model

$F \leftarrow$ {the list of all semantic frames in the training data}
$V \leftarrow$ {a mapping of lemmas to vectors, as generated by DependencyVectors}
$SP \leftarrow$ {a mapping of roles to vectors, as generated by Algorithm 1}
for f in F **do** {f is the current semantic frame.}
  $D_f \leftarrow []$ {D_f will be a list of role / feature set pairs}
  $E \leftarrow$ {the set of all training examples in frame f}
  for e in E **do** {e is the current training example}
    $p \leftarrow$ {the predicate for training example e}
    $A \leftarrow$ {the list of arguments for training example e}
    for a in A **do** {a is the current argument}
      $l_a \leftarrow$ {the lemma for the argument a}
      $v_l \leftarrow V[l_a]$ {v_l is the vector from V that represents the lemma l_a}
      $C_{ea} \leftarrow []$ {C_{ea} will be a mapping of roles to angular cosines}
      $R \leftarrow$ {the list of all semantic roles in f}
      for r in R **do** {r is the current semantic role.}
        $sp_r \leftarrow SP[r]$ {sp_r is the selectional preference vector from SP for r}
        $c_r \leftarrow$ {the cosine of the angle between the two vectors v_l and sp_r}
        $C_{ea}[r] \leftarrow c_r$
      end for
      $fs_{ea} \leftarrow$ {the feature set for the the example e and the argument a as generated by the benchmark model}
      {This feature set includes: }
      { The predicate for this example: p }
      { The lemma for this argument: l_a }
      { The part of speech of l_a (Noun, Verb, Adjective, etc.) }
      { The syntactic role of l_a in this example (Subject, Object, etc.) }
      $fs_{sp} \leftarrow fs_{ea} + C_{ea}$
      $r \leftarrow$ {the semantic role assigned to the argument a (the gold label)}
      $D_f[ea] \leftarrow (r, fs_{sp})$
    end for
  end for
  The role / feature set pairs in D_f are then used to train a classifier for f.
end for

3.3 Example

To demonstrate how the benchmark and selectional preference models differ, I will use a short example training corpus:

(14) John eats apples.

(15) John eats grapes.

(16) John eats outside.

3.3.1 Benchmark Model

The training data given to the benchmark model has these annotations[4]:

Label	**Predicate**	**Lemma**	**Part of Speech**	**Syntactic Role**
Ingestibles	eats	apples	Noun	Object
Ingestibles	eats	grapes	Noun	Object
Place	eats	outside	Noun	Object

Now let's assume that during testing the following sentence with a constituent that has a previously unseen headword is found:

(17) John eats someplace near campus.

This sentence would be assigned the following features by the benchmark model:

[4]We're working on the assumption that the syntactic parser labels "outside" as an object due to the fact that it is a noun following a ditransitive verb. This ambiguity exists because the verb "to eat" can also be intransitive, as is actually the case with (16).

Predicate	**Lemma**	**Part of Speech**	**Syntactic Role**
eats	someplace	Noun	Object

The headword "someplace" wasn't seen in the training data, so the benchmark model assigns it the label that was most common given the other features that were provided. This causes the benchmark model to incorrectly label the constituent "someplace near campus" with the "Ingestibles" role.

3.3.2 Selectional Preferences Model

The same training data is given to the selectional preferences model with these annotations[5]:

Label	**Pred.**	**Lemma**	**POS**	**Syn. Role**	$\mathbf{SP}_{Ingestibles}$	$\mathbf{SP}_{Place}$
Ingestibles	eats	apples	Noun	Object	0.95	0.10
Ingestibles	eats	grapes	Noun	Object	0.95	0.15
Place	eats	outside	Noun	Object	0.25	0.90

Now let's assume that during testing the following sentence with a constituent that has a previously unseen headword is found:

(18) John eats someplace near campus.

The sentence would be assigned the following features by the selectional preferences model:

Predicate	**Lemma**	**POS**	**Syn. Role**	$\mathbf{SP}_{Ingestibles}$	$\mathbf{SP}_{Place}$
eats	someplace	Noun	Object	0.25	0.95

[5]SP_X represents the cosine of the angle between this headword's vector and the average vector for all the headwords that fill role X. In this example the SP_X values are arbitrary and do not represent actual values for the given data.

Since the cosine of the angle between the vector for "someplace" and the selectional preference vector for the "Place" role is larger than the cosine of the angle between the vector for "someplace" and the selectional preference vector for the "Ingestibles" role, our model is able to correctly return the "Place" label for this headword.

Chapter 4

Results

This chapter discusses the results of my experiments. The experiments were conducted in a controlled automatic role labeling setting to evaluate the effectiveness of using selectional preferences with various classifiers.

In the sections below, the "benchmark" model refers to the model described in §3.3.1. The benchmark model does not contain information on selectional preferences. The "full" or "selectional preferences" model refers to the model described in §3.3.2. The full model contains the same information as the benchmark model plus information on selectional preferences.

4.1 Classifiers

In order to investigate whether or not selectional preferences improve the ability of machine learning systems to properly assign semantic role labels, I needed to eliminate the possibility that the choice of machine learning system used in the experiment would affect the results. In order to do this I ran the experiment through three separate machine learning systems. I then examined the results from each system. This way I can also investigate whether or not certain machine learning systems are affected more by the addition of selectional preferences than other systems are. The three systems (or classifiers) that I chose for the experiment were a decision tree classifier, a naïve Bayes classifier, and a support vector machine classifier. Each classifier uses a fundamentally different method of learning and applying the training data[1].

4.2 Decision Tree Classifier

Figure 4.1 and Table 4.1 show the performance of the decision tree classifier[2] using the model that included selectional preferences compared to the performance of the classifier using the model that did not include them.

For frames with between 100 and 1000 training examples, using the selectional preferences model consistently increased the accuracy of the decision tree classifier over the benchmark model. However, the frames with a small number of training examples produced less accurate results with the selectional preferences model than the benchmark model. This was especially noticeable when looking at the frames with between 1 and 20 training examples, where accuracy dropped more than 11%.

[1] A discussion of these methods is beyond the scope of this paper. See [19] for more information on decision tree classifiers, [9] for more information on Bayesian classifiers, and [11] for more information on support vector machines.

[2] The J48 classifier from the Weka project was used for this experiment.

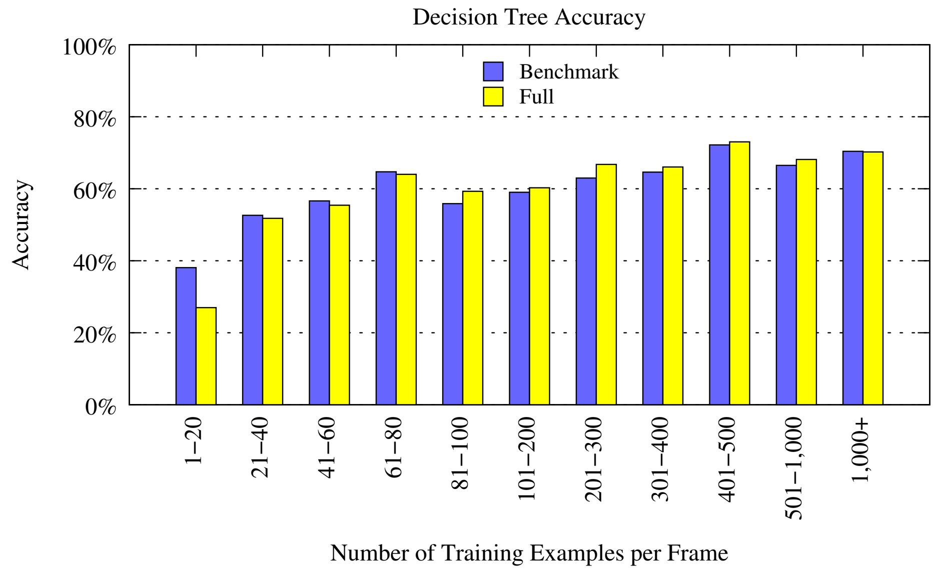

Figure 4.1: Accuracy of Decision Tree Classifier

# of Examples	Benchmark	Full	Difference
1-20	38.095%	26.984%	-11.111%
21-40	52.600%	51.798%	-0.802%
41-60	56.622%	55.432%	-1.19%
61-80	64.732%	64.008%	-0.724%
81-100	55.873%	59.274%	3.401%
101-200	59.013%	60.268%	1.255%
201-300	62.992%	66.763%	3.771%
301-400	64.609%	66.026%	1.417%
401-500	72.196%	73.032%	0.836%
501-1000	66.505%	68.116%	1.611%
1000+	70.403%	70.218%	-0.185%
		Average Difference:	-0.143%

Table 4.1: Accuracy of Decision Tree Classifier

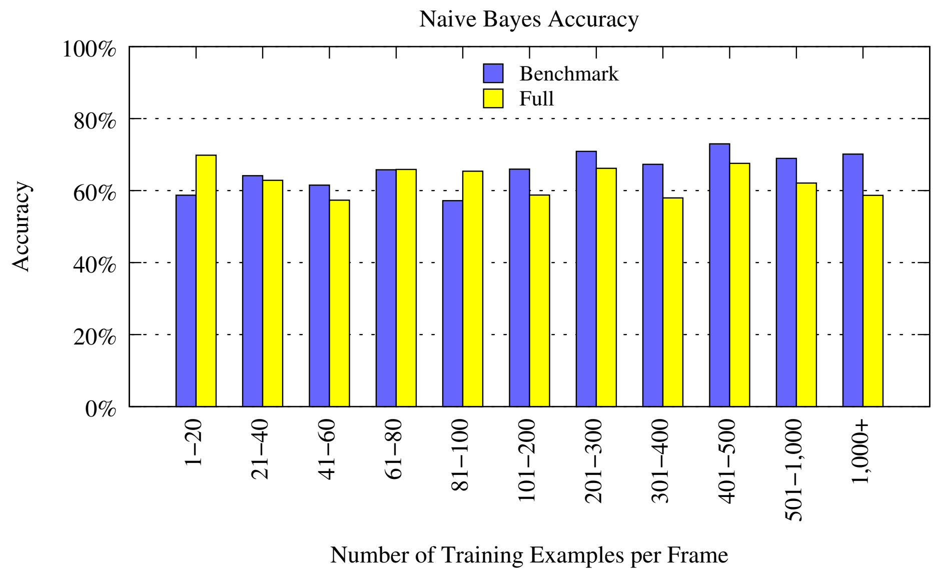

Figure 4.2: Accuracy of Naïve Bayes Classifier

4.3 Naïve Bayes Classifier

Figure 4.2 and Table 4.2 show the performance of the model which included selectional preferences compared to the performance of the benchmark model for the naïve Bayes classifier[3].

Whereas the decision tree classifier did poorly on frames with a small number of training examples and well on those with a large number of training examples, the naïve Bayes classifier responded in exactly the opposite way. Frames with a small number of training examples performed much better with the selectional preferences model than the benchmark model, as shown by the 11% increase in accuracy among frames with between 1 and 20 training

[3]The NaiveBayes classifier from the Weka project was used for this experiment.

# of Examples	Benchmark	Full	Difference
1-20	58.730%	69.841%	11.111%
21-40	64.125%	62.843%	-1.282%
41-60	61.525%	57.352%	-4.173%
61-80	65.769%	65.857%	0.088%
81-100	57.201%	65.359%	8.158%
101-200	65.947%	58.777%	-7.17%
201-300	70.869%	66.186%	-4.683%
301-400	67.276%	57.948%	-9.328%
401-500	72.957%	67.576%	-5.381%
501-1000	68.928%	62.108%	-6.82%
1000+	70.131%	58.680%	-11.451%
		Average Difference:	-2.578%

Table 4.2: Accuracy of Naïve Bayes Classifier

examples. However, large frames had reduced accuracy with the selectional preferences model, especially those frames with more than 1000 training examples.

4.4 Support Vector Machine Classifier

Figure 4.3 and Table 4.3 show the performance of the model which included selectional preferences compared to the performance of the benchmark model for the support vector machine classifier[4].

Out of all three classifiers that were used in this experiment, the support vector machine classifier showed the smallest difference between the selectional

[4]The SMO classifier from the Weka project was used for this experiment.

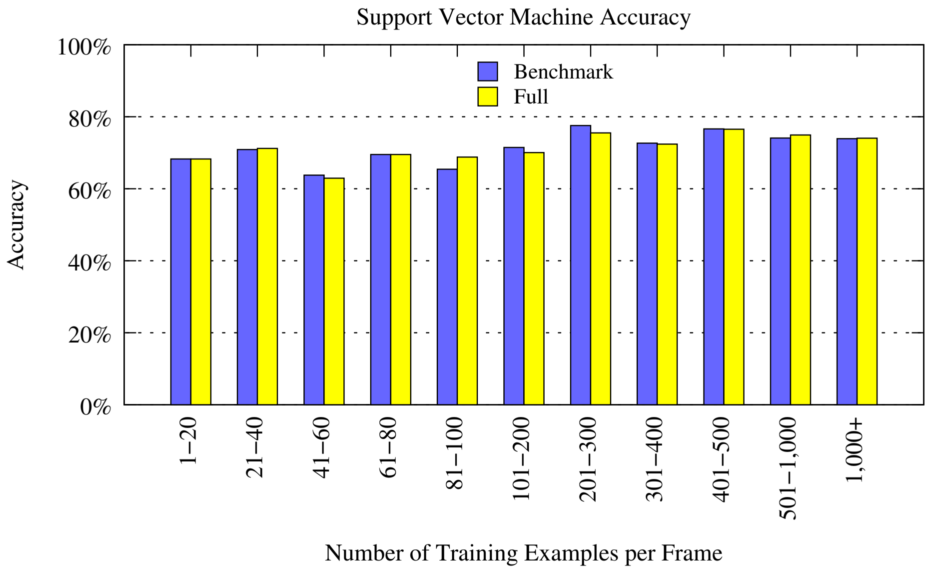

Figure 4.3: Accuracy of Support Vector Machine Classifier

# of Examples	Benchmark	Full	Difference
1-20	68.254%	68.254%	0%
21-40	70.879%	71.202%	0.323%
41-60	63.772%	62.918%	-0.854%
61-80	69.533%	69.497%	-0.036%
81-100	65.419%	68.786%	3.367%
101-200	71.454%	70.028%	-1.426%
201-300	77.566%	75.490%	-2.076%
301-400	72.666%	72.397%	-0.269%
401-500	76.628%	76.536%	-0.092%
501-1000	74.090%	74.929%	0.839%
1000+	73.912%	74.056%	0.144%
		Average Difference:	-0.007%

Table 4.3: Accuracy of Support Vector Machine Classifier

preferences model and the benchmark model. Frames with very small and very large numbers of training examples showed very little difference between the two models. In fact, frames with between 1 and 20 training examples showed no difference at all. However, frames with a medium number of training examples showed more of a difference between the two models. Those with between 81 and 100 training examples showed an improvement with the selectional preferences model over the benchmark model, but those with 101 - 300 training examples showed a decrease in accuracy.

Chapter 5

Conclusion

The results of the experiment show that adding selectional preference data to a model can improve the performance of semantic role labeling systems. However, the degree of improvement is greatly dependent on the classifier used and the size of the training corpus. The accuracy of the naïve Bayes classifier was greatly improved by the selectional preferences model when only a handful of training examples were available. However, in almost all other cases the selectional preferences model hindered the accuracy of the naïve Bayes classifier. On the other hand, the decision tree classifier showed an improvement in accuracy when given the selectional preferences model and more than 80 training examples. This leads me to conclude that different classifiers should be used for different amounts of training data. A larger semantic role labeling system might work better if it used a naïve Bayes classifier with small datasets and a decision tree classifier with large datasets[1].

When the number of training examples was very large, the accuracy

[1] However, it should be noted that the support vector machine outperformed both naïve Bayes and the decision tree classifier even without the selectional preferences model. If feasible, the support vector machine classifier should be used instead of either naïve Bayes or a decision tree classifier.

of the selectional preferences model was very close to that of the benchmark model for the decision tree and support vector machine classifiers. I believe this is due to the fact that these classifiers, when given a large enough dataset, are able to glean enough information from the other features provided to them that the selectional preference data is redundant.

5.1 Future Work

It is clear that selectional preferences do not improve accuracy when a frame has a large number of training examples. More work is needed to discover whether these frames are deriving the information they need for semantic role labeling from previously seen headwords or from previously seen sub-categorization frames. This could be done by carefully selecting training data so as to first limit previously seen headwords while maximizing previously seen sub-categorization frames, then by limiting previously seen sub-categorization frames while maximizing previously seen headwords.

Although I was able to determine overall trends by lumping together frames based on the number of training examples available, I noticed that the results for individual frames within those groups were much more erratic. The accuracy of some frames improved drastically while the accuracy of others decreased just as drastically despite the two frames having approximately the same number of training examples. More work is needed to determine exactly what other factors cause a given frame to perform better than another frame of the same size when given a selectional preferences model.

Appendix A: Results By Frame

	# of	Naïve Bayes Accuracy		Decision Tree Accuracy		Support Vector Machine Accuracy	
Frame	Training Examples	Benchmark	Full	Benchmark	Full	Benchmark	Full
Achieving_first	111	57.143%	47.619%	47.619%	47.619%	59.524%	52.381%
Adjusting	46	42.857%	57.143%	42.857%	42.857%	50.000%	57.143%
Adorning	158	80.357%	73.214%	51.786%	51.786%	85.714%	82.143%
Amassing	120	63.889%	36.111%	58.333%	55.556%	63.889%	63.889%
Amounting_to	101	90.625%	84.375%	81.250%	81.250%	90.625%	87.500%
Apply_heat	141	65.000%	65.000%	65.000%	62.500%	72.500%	72.500%
Arraignment	29	35.714%	35.714%	50.000%	50.000%	71.429%	64.286%
Arrest	131	56.818%	47.727%	34.091%	34.091%	63.636%	70.455%
Arriving	351	77.143%	55.238%	75.238%	74.286%	80.952%	80.000%
Assessing	124	32.727%	34.545%	34.545%	34.545%	40.000%	47.273%
Attaching	414	72.848%	66.225%	68.212%	72.185%	76.159%	76.159%
Avoiding	315	68.421%	58.947%	67.368%	68.421%	75.789%	71.579%
Awareness	974	72.671%	68.634%	63.975%	63.975%	76.398%	76.708%
Becoming	161	48.837%	44.186%	48.837%	48.837%	60.465%	62.791%
Becoming_aware	468	69.444%	62.222%	75.556%	75.000%	73.889%	72.778%
Behind_the_scenes	150	61.667%	61.667%	56.667%	56.667%	71.667%	61.667%
Being_attached	135	68.293%	65.854%	65.854%	65.854%	75.610%	70.732%
Being_born	32	62.500%	62.500%	62.500%	62.500%	87.500%	87.500%
Being_named	73	70.833%	58.333%	54.167%	54.167%	75.000%	75.000%
Biological_urge	145	81.818%	79.545%	75.000%	84.091%	86.364%	86.364%
Birth	65	62.069%	41.379%	51.724%	51.724%	55.172%	51.724%
Body_description_holistic	116	85.714%	85.714%	85.714%	96.429%	100.000%	96.429%
Body_movement	1702	70.000%	55.965%	71.404%	71.053%	75.614%	74.561%
Breathing	188	75.000%	51.786%	69.643%	69.643%	76.786%	76.786%
Businesses	50	66.667%	66.667%	66.667%	66.667%	66.667%	66.667%
Categorization	339	62.931%	62.931%	63.793%	63.793%	68.966%	70.690%
Causation	189	68.519%	61.111%	51.852%	51.852%	77.778%	74.074%
Cause_change_of_position_on_a_scale	198	48.214%	41.071%	46.429%	46.429%	51.786%	55.357%

Frame	# of Training Examples	Naïve Bayes Accuracy		Decision Tree Accuracy		Support Vector Machine Accuracy	
		Benchmark	Full	Benchmark	Full	Benchmark	Full
Cause_expansion	145	60.784%	58.824%	66.667%	66.667%	74.510%	66.667%
Cause_fluidic_motion	134	63.265%	67.347%	63.265%	65.306%	71.429%	73.469%
Cause_harm	987	67.628%	53.205%	66.026%	66.026%	70.513%	71.795%
Cause_to_amalgamate	144	36.538%	36.538%	40.385%	40.385%	44.231%	36.538%
Cause_to_fragment	174	65.079%	55.556%	65.079%	66.667%	68.254%	68.254%
Cause_to_wake	91	50.000%	55.882%	50.000%	52.941%	58.824%	61.765%
Change_event_time	131	63.415%	53.659%	65.854%	70.732%	70.732%	70.732%
Change_position_on_a_scale	906	51.590%	27.915%	50.177%	50.177%	57.244%	56.890%
Clothing	705	80.444%	82.222%	80.000%	82.667%	83.556%	85.333%
Cognitive_connection	75	76.190%	80.952%	66.667%	66.667%	76.190%	76.190%
Collaboration	187	77.778%	61.905%	61.905%	61.905%	77.778%	80.952%
Commerce_buy	167	65.079%	61.905%	60.317%	68.254%	68.254%	69.841%
Commerce_pay	123	61.111%	50.000%	36.111%	36.111%	58.333%	61.111%
Commerce_scenario	51	80.952%	76.190%	80.952%	76.190%	80.952%	85.714%
Commerce_sell	69	52.632%	57.895%	42.105%	57.895%	52.632%	63.158%
Committing_crime	36	37.500%	50.000%	25.000%	25.000%	50.000%	50.000%
Communication	53	57.895%	42.105%	47.368%	47.368%	57.895%	52.632%
Communication_manner	590	74.372%	67.839%	72.362%	75.879%	76.382%	75.879%
Competition	51	50.000%	40.000%	30.000%	30.000%	50.000%	45.000%
Complaining	147	63.043%	54.348%	39.130%	43.478%	73.913%	65.217%
Congregating	154	62.069%	56.897%	65.517%	68.966%	72.414%	72.414%
Contacting	467	47.826%	42.029%	43.478%	43.478%	63.768%	64.493%
Containers	581	87.097%	90.323%	85.484%	91.935%	91.398%	94.086%
Contingency	246	74.026%	71.429%	59.740%	66.234%	89.610%	88.312%
Cotheme	647	70.563%	54.978%	74.026%	74.026%	75.325%	74.892%
Court_examination	33	61.538%	53.846%	46.154%	53.846%	53.846%	61.538%
Cure	229	50.000%	45.000%	46.667%	46.667%	65.000%	58.333%
Custom	91	80.488%	80.488%	80.488%	80.488%	82.927%	82.927%

Frame	# of Training Examples	Naïve Bayes Accuracy		Decision Tree Accuracy		Support Vector Machine Accuracy	
		Benchmark	Full	Benchmark	Full	Benchmark	Full
Departing	396	68.085%	59.574%	56.028%	69.504%	68.794%	70.922%
Desirability	352	74.561%	68.421%	76.316%	76.316%	80.702%	79.825%
Desiring	603	68.063%	64.398%	62.304%	62.304%	69.110%	70.157%
Detaining	68	53.571%	57.143%	64.286%	60.714%	60.714%	60.714%
Difficulty	238	80.952%	82.540%	88.889%	88.889%	88.889%	88.889%
Discussion	239	63.043%	54.348%	58.696%	58.696%	65.217%	65.217%
Duplication	159	63.462%	48.077%	40.385%	40.385%	69.231%	63.462%
Duration	70	83.333%	87.500%	79.167%	79.167%	91.667%	87.500%
Education_teaching	387	47.244%	25.984%	45.669%	44.882%	52.756%	52.756%
Emptying	282	72.043%	62.366%	72.043%	72.043%	76.344%	76.344%
Encoding	234	79.221%	72.727%	68.831%	70.130%	81.818%	79.221%
Entering_of_plea	18	87.500%	75.000%	87.500%	75.000%	62.500%	75.000%
Estimating	69	43.478%	52.174%	56.522%	52.174%	47.826%	47.826%
Evading	81	73.810%	64.286%	76.190%	76.190%	83.333%	78.571%
Event	173	73.469%	69.388%	57.143%	57.143%	73.469%	71.429%
Evidence	444	80.272%	77.551%	79.592%	79.592%	82.313%	82.313%
Examination	89	62.069%	75.862%	72.414%	72.414%	75.862%	75.862%
Expectation	415	82.500%	80.625%	81.250%	81.250%	93.750%	92.500%
Expensiveness	96	64.706%	64.706%	67.647%	67.647%	70.588%	67.647%
Experiencer_subj	1425	72.764%	66.870%	73.780%	73.577%	75.610%	75.813%
Expertise	254	71.765%	60.000%	52.941%	65.882%	69.412%	70.588%
Explaining_the_facts	119	72.093%	74.419%	65.116%	65.116%	72.093%	74.419%
Expressing_publicly	429	58.621%	44.138%	62.069%	63.448%	65.517%	64.138%
Facial_expression	173	81.481%	70.370%	79.630%	74.074%	85.185%	79.630%
Filling	578	73.632%	69.154%	70.647%	76.119%	76.119%	78.607%
Fluidic_motion	400	79.545%	72.727%	74.242%	75.000%	81.061%	80.303%
Gesture	199	64.615%	52.308%	63.077%	70.769%	75.385%	69.231%
Giving	357	71.552%	67.241%	71.552%	71.552%	72.414%	72.414%

Frame	# of Training Examples	Naïve Bayes Accuracy		Decision Tree Accuracy		Support Vector Machine Accuracy	
		Benchmark	Full	Benchmark	Full	Benchmark	Full
Grasp	242	54.321%	45.679%	54.321%	51.852%	66.667%	58.025%
Hair_configuration	55	63.636%	72.727%	68.182%	68.182%	68.182%	72.727%
Hindering	188	60.938%	37.500%	57.812%	57.812%	65.625%	64.062%
Hiring	101	63.636%	57.576%	63.636%	66.667%	69.697%	69.697%
Hostile_encounter	284	53.684%	52.632%	36.842%	36.842%	60.000%	57.895%
Importance	112	80.488%	65.854%	53.659%	53.659%	80.488%	82.927%
Instance	14	66.667%	66.667%	66.667%	33.333%	66.667%	66.667%
Invention	452	62.676%	59.155%	63.380%	63.380%	69.718%	71.127%
Judgment	866	64.093%	50.965%	59.459%	59.073%	67.568%	69.498%
Judgment_direct_address	673	67.857%	64.732%	69.643%	69.643%	78.125%	77.679%
Jury_deliberation	23	100.000%	100.000%	44.444%	44.444%	100.000%	88.889%
Killing	302	61.702%	56.383%	46.809%	46.809%	67.021%	69.149%
Kinship	230	90.541%	93.243%	72.973%	94.595%	94.595%	94.595%
Knot_creation	8	66.667%	100.000%	33.333%	33.333%	66.667%	66.667%
Leadership	295	77.679%	73.214%	66.071%	71.429%	84.821%	82.143%
Light_movement	85	64.103%	71.795%	58.974%	53.846%	74.359%	74.359%
Linguistic_meaning	21	42.857%	42.857%	14.286%	14.286%	71.429%	71.429%
Location_of_light	301	83.158%	81.053%	77.895%	77.895%	88.421%	86.316%

Bibliography

[1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[2] Carsten Brockmann and Mirella Lapata. Evaluating and combining approaches to selectional preference acquisition. In *Proceedings of EACL 2003.*, Budapest, 2003.

[3] Xavier Carreras and Lluís Màrquez, editors. *Proceedings of the CoNLL shared task: Semantic role labeling.*, 2004.

[4] Xavier Carreras and Lluís Màrquez, editors. *Proceedings of CoNLL shared task: Semantic role labeling.*, 2005.

[5] Katrin Erk. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL 2007*, 2007.

[6] Charles J. Fillmore. The case for case. In *Universals in Linguistic Theory*, 1968.

[7] Charles J. Fillmore. Scenes-and-frames semantics. In *Linguistic Structures Processing*, pages 55–82. North Holland Publishing, 1977.

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

[9] George John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.

[10] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2nd edition, 2008.

[11] S.S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design, 1999.

[12] Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[13] Rada Mihalcea and Phil Edmonds, editors. *Proceedings of Senseval-3: The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, 2004.

[14] Joakim Nivre. Sorting out dependency parsing. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL)*, pages 16–27, 2008.

[15] Sebastian Pado and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[16] Martha Palmer, Paul Kingsbury, and Daniel Gildea. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[17] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[18] Sameer S. Pradhan, Wayne Ward, and James H. Martin. Towards robust semantic role labeling. *Computational Linguistics*, 34(2):289–310, 2008.

[19] Ross J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.

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

[21] Philip S. Resnik. Wordnet and class-based probabilities. In *WordNet: An Electronic Lexical Database, chapter 10*, pages 239–263. MIT Press, 1998.

[22] Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. Inducing an semantically annotated lexicon via em-based clustering. In *Proceedings of ACL 1999*, Maryland, 1999.

Vita

Andrew Cleburne Young was born in San Antonio, Texas on November 26^{th}, 1980, the son of Patrick Robert Young and Roberta Ann Houston Young. In the 8^{th} grade he applied to and was accepted by Business Careers High School – a competitive magnet school in San Antonio. During high school he won regional and statewide awards for computer programming and began working for a local Internet company. After high school he moved to Austin, Texas and began his programming career at IBM. After six years at IBM he moved on to several small companies before ending up at Rackspace, where he works on their cloud computing initiative. While working at IBM in 2004, Andrew decided to go back to school to earn his Bachelors degree, and was accepted into the computer science program at the University of Texas at Austin. He changed his major to linguistics in 2006 after taking an introductory linguistics course as part of his core curriculum requirements. During his senior year he applied to the linguistics graduate program at the University of Texas at Austin and was accepted. He begins his graduate studies in 2009.

Permanent address: 6406 Avery Island Ave
Austin, Texas 78727

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.