Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler

Patrick Ye and **Timothy Baldwin**
Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

{jingy,tim}@csse.unimelb.edu.au

Abstract

This paper investigates whether multi-semantic-role (MSR) based selectional preferences can be used to improve the performance of supervised verb sense disambiguation. Unlike conventional selectional preferences which are extracted from parse trees based on hand-crafted rules, and only include the direct subject or the direct object of the verbs, the MSR based selectional preferences to be presented in this paper are extracted from the output of a state-of-the-art semantic role labeler and incorporate a much richer set of semantic roles. The performance of the MSR based selectional preferences is evaluated on two distinct datasets: the verbs from the lexical sample task of SENSEVAL-2, and the verbs from a movie script corpus. We show that the MSR based features can indeed improve the performance of verb sense disambiguation.

1 Introduction

Verb sense disambiguation (VSD) is the task of examining verbs in a given context and specifying exactly which sense of each verb is the most appropriate in that context. VSD is a subtask of word sense disambiguation (WSD). Given a verb sense inventory and a set of verb senses, VSD is essentially a classification task (Yarowsky, 2000).

VSD has not received much attention in the literature of WSD until recently. Most of WSD systems disambiguate verbs in the same way as nouns using mostly collocation based features, and this has led to rather poor VSD performance on corpora such as SENSEVAL-2. One useful but often ignored source of disambiguation information for verbs is the patterns of verb-argument structures. In this paper, we will attempt to capture these patterns through the use of selectional preferences.

In general, selectional preferences describe the phenomenon that predicating words such as verbs and adjectives tend to favour a small number of **noun classes** for each of their arguments. For example, the verb *eat* ("take in solid food") tends to select nouns from to the ANIMATED_THING class as its EATER role, and nouns from the EDIBLE class as its EATEE role.

However, it is possible to extend the concept of selectional preferences to include nouns which function as adjuncts to predicating words. For example, the verb *hit* in the sentence *I hit him with my fists* stands for "deal a blow to, either with the hand or with an instrument", but in the sentence *I hit him with a car*, it stands for "to collide with", with the only difference between the two instances of *hit* being their manner modifiers. Intuitively, the inclusion of verb adjuncts can enrich the semantic roles (SRs) and provide additional disambiguation information for verbs. Therefore, in the rest of this paper, the concept of "semantic role" will include both the arguments and adjuncts of verbs.

All the selectional preference based WSD systems to date have only used the subject and direct object of verbs as semantic roles, extracting the necessary argument structure via hand-crafted heuristics (Resnik, 1997; McCarthy and Carroll, 2003, inter alia). As a result, it is difficult to extend the selectional preferences to anything else. However, with recent progress in Semantic Role Labelling (SRL) technology, it is now possible to obtain additional semantic roles such as the indirect object of ditransitive verbs and the locational, temporal and manner adjuncts.

Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 139–148.

Given the lack of research in VSD using multi-semantic-role based selectional preferences, the main contribution of the work presented in this paper is to show that it is possible to use the multi-semantic-role based selectional preferences extracted using the current state-of-the-art SRL systems to achieve a certain level of improvement for verb VSD. We also give detailed descriptions for all the features, feature selection algorithms and the tuning of the machine learner parameters that we have used in the construction of our VSD system, so that our system can be easily reproduced.

The remainder of this paper is organized as follows: we first introduce the framework for constructing and evaluating the VSD systems in Section 2; we then give detailed descriptions of all the feature types that we experimented with during our research in Section 3; Section 4 introduces the two datasets used to train and evaluate the features; the feature selection algorithms are presented in Section 5; the results of our experiments are presented in Section 6; and finally we conclude in Section 7.

2 VSD Framework

There are three components in our VSD Framework: extraction of the disambiguation features from the input text (feature extraction), selection of the best disambiguation features with respect to unknown data (feature selection), and the tuning of the machine learner's parameters. Since feature extraction is explained in detail in Section 3, we will only disscuss the other two components here.

Within our framework, feature selection is performed only on the training set. We first use the feature selection algorithms described in Section 5 to generate different feature sets, which are used to generate separate datasets. We then perform cross validation (CV) on each dataset, and the feature set with the best performance is chosen as the final feature set.

The machine learning algorithm used in our study is Maxium Entropy (MaxEnt: Berger et al. (1996)[1]). MaxEnt classifiers work by modelling the probability distribution of labels with respect to disambiguation features, the distribution of which is commonly smoothed based on a Gaussian prior. Different values for the Gaussian prior often lead to significant differences in the classification of new data, motivating us to include the tuning of the Gaussian prior in VSD framework.[2]

The tuning of the Gaussian prior is performed in conjunction with the feature selection. The CV for each feature set is performed multiple times, each time with a different parameterisation of the Gaussian prior. Therefore, the final classifier incorporates the best combination of feature set and parameterisation of the Gaussian prior for the given dataset.

3 Features Types

Since selectional preference based WSD features and general WSD features are not mutually exclusive of each other, and it would be less convincing to evaluate the impact of selectional preference based features without a baseline derived from general WSD features, we decided to include a number of general WSD features in our experiments. The sources of these features include: Part-of-Speech tags extracted using a tagger described in Gimnez and Mrquez (2004); parse trees extracted using the Charniak Parser (Charniak, 2000); chunking information extracted using a statistical chunker trained on the Brown Corpus and the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1993); and named entities extracted using the system described in Cohn et al. (2005).

3.1 General WSD Features

There are 3 broad types of general WSD features: n-gram based features of surrounding words and WordNet noun synsets, parse-tree-based syntactic features, and non-parse-tree based syntactic features.

3.1.1 N-gram based features

The following n-gram based features have been experimented with:

Bag of Words Lemmatized open class words in the entire sentence of the target verb. Words that occur multiple times are only counted once.

Bag of Synsets The WordNet (Fellbaum, 1998) synsets for all the open class words in the entire

[1] We used Zhang Le's implementation which is available for download at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

[2] We experimented with the following settings for the standard deviation (with a mean of 0) of the Gaussian prior in all of our experiments: 0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0, 500.0, 1000.0.

sentence; hypernyms for nouns and verbs are also included.

Bag of POS Tags The POS tag of each word within a window of 5 words surrounding the target verb, paired with its relative position and treated as a separate binary feature.

Bag of Chunk Tags The chunk tags (in Inside-Outside-Beginning (IOB) format: Tjong Kim Sang and Veenstra (1999)) surrounding and including the target verb within a window of 5 words, paired with their relative positions.

Bag of Chunk Types The chunk types (e.g. NP, VP) surrounding and including the target verb within a window of 5 words.

Bag of Named Entities The named entities in the entire sentence of the target verb.

Left Words Each lemmatized word paired with its relative position to the left of the target verb within a predefined window.

Right Words Each lemmatized word paired with its relative position to the right of the target verb within a predefined window.

Surrounding Words The union of **Left Words** and **Right Words** features.

Left Words with Binary Relative Position Each lemmatized word and its position[3] to the left of the target verb within a predefined window.

Right Words with Binary Relative Position Each lemmatized word and its binary position to the right of the target verb within a predefined window.

Surrounding Words with Binary Relative Position The union of **Left Words with Binary Position** and **Right Words with Binary Position** features.

Left POS-tags The POS tag of each word to the left of the target verb within a predefined window, paired with its relative position.

Right POS-tags The POS tag of each word to the right of the target verb within a predefined window is paired with its relative position.

Surrounding POS Tags The Union of the **Left POS-tags** and **Right POS-tags** features.

It may seem redundant that for the same window size, the Surrounding-Words (POS) features are the union of the Left-Words (POS) features and the Right-Words (POS) features. However, this redundancy of features makes it convenient to investigate the disambiguation effectiveness of the word collocations before and after the target verb, as well as the syntactic pattern before and after the target verb. Furthermore, we have also experimented with different window sizes for the Surrounding-Words (POS), Left-Words (POS) and Right-Words (POS) to determine the most appropriate window size.[4]

3.1.2 Parse tree based features

The parse tree based syntactic features are inspired by research on verb subcategorization acquisition such as Korhonen and Preiss (2003), and are intended to capture the differences in syntactic patterns of the different senses of the same verb. Given the position of the target verb v in the parse tree, the basic form of the corresponding parse tree feature is just the list of nodes of v's siblings in the tree. Figure 1 shows the parse tree for a sentence containing the ambiguous verb *call*. Given the position of the target verb *called* in the parse tree, the basic form of the features that can be created will be (NP, PP). However, there are 3 additional types of variations that can be applied to the basic features. The first variation is to include the relative positions of the sibling node types as part of the feature: this variation will change the basic feature for *call* to $\{(1, NP), (2, PP)\}$. The second variation is to include the binary relative direction of the siblings to the target verb as part of the feature, i.e. is the sibling to the left or right of the target verb: this variation will change the basic feature for *call* to $\{(right, NP), (right, PP)\}$. The third variation is to include the parent node type as part of the sibling node type to add more information in the syntactic pattern. Figure 2 shows what the original parse tree looks like when every nonterminal is additionally annotated with its parent type. Since the third variation is compatible with the first two variations, we decided to combine them to create the following parse tree

[3] All words to the left of the target verb are given the “left” position, and all the to the right of the target verb are given the “right” position.

[4] In the ranking based evaluation method described in Section 5, only the Surrounding-Words (POS) feature types with the largest window size are used.

based features:

Type 1 Original sibling node types (zero level of parent node annotated) with relative positions.

Type 2 Original sibling node types with relative binary directions.

Type 3 One level of parent-node-annotated sibling node types with relative positions. Using the *call* example, this feature will be $\{(1, \textit{VP-NP}), (2, \textit{VP-PP})\}$.

Type 4 One level of parent-node-annotated sibling node types with relative binary directions. Using the *call* example, this feature will be $\{(right, \textit{VP-NP}), (right, \textit{VP-PP})\}$.

Type 5 Two levels of parent-node-annotated sibling node types with relative positions.

Type 6 Two levels of parent-node-annotated sibling node types with relative binary directions.

On top of the 6 types of purely syntactic based parse tree features, there is an additional type of parse tree based feature designed to capture the **verb-argument structure** of the target verb. The type of features only cover four particular types of parse tree nodes which are immediately after the pre-terminal node of the target verb. The four types of parse tree nodes are: ADVP, NP, PP and clausal nodes.

For an ADVP node, we extract its head adverb H_ADV, and treat the tuple of (**ADVP**, H_ADV) as a separate binary feature.

For an NP node, we first extract its head nouns, then replace each head noun with its WordNet synsets and the hypernyms of these synsets, and treat each of these synsets as a separate binary feature. In order to cover the cases in which the head noun of a NP node is a quantity noun, e.g. *a glass of water*, the head nouns of PPs attached to the NP nodes are also included as head nouns. Furthermore, head nouns which are named entities identified by the system described in Cohn et al. (2005) are replaced by appropriate WordNet synsets.

For a PP node, we first extract its head preposition, then we extract the head noun synsets in the same way as the NP node, and finally we combine each synset with the head preposition to form a separate binary feature.

For a clausal node which is an SBAR, we extract the list of node types of its direct children and arrange them in their original order, and treat this list as a single binary feature.

For a clausal node which is not an SBAR, but has a single non-terminal child node, we first extract the type of this child node, then we extract the list of node types for the children of this child node. The tuple of (child-node-type, list-of-grandchildren-node-types) is then treated as a separate binary feature.

3.1.3 Non-parse tree based syntactic features

There are 3 types of non-parse-tree based syntactic features:

Voice of the verb The voice of the target verb (active or passive).

Quotatives Verbs that appear in directly quoted speech have a greater likelihood of occurring in the imperative and losing the surface subject, e.g. *"Call the police!"*. We therefore include this as a standalone feature.

Additional Chunk based features A number of additional chunk based features are also used to capture the phrase level localized syntactic and collocation patterns from the context to the right of the target verb within a window of between 3 and 10 chunks. For example, using a window of 7, for the verb *kick* in the sentence: *[I/PRP]$_{NP}$ [**kicked**/VBD]$_{VP}$ [him/PRP]$_{NP}$ [out/IN of/IN]$_{PP}$ [the/DT door/NN]$_{NP}$ [through/IN]$_{PP}$ [which/WDT]$_{NP}$ [he/PRP]$_{NP}$ [came/VBD]$_{VP}$*, the first 7 chunks after the chunk that contains **kick** will be used for feature extraction. These additional chunk based features are:

Chunk-type-sequence The concatenation of all the relevant chunk types. For example, using the above *kick* example, this feature will be **NP_PP_NP_PP_NP_NP_VP**.

Regular expression (RE) representation of the chunk types The consecutive identical chunk types in the **Chunk-type-sequence** feature merged into a single symbol. For example, the chunk-type-sequence of **NP_PP_NP_PP_NP_NP_VP** will be represented as **NP_PP_NP_PP_NP+_VP**.

First word of each chunk with the chunk type The list that contains the first word of each chunk will be treated as a separate binary feature. With the *kick* example, this feature would be **him_out_the_through_which_he_came**.

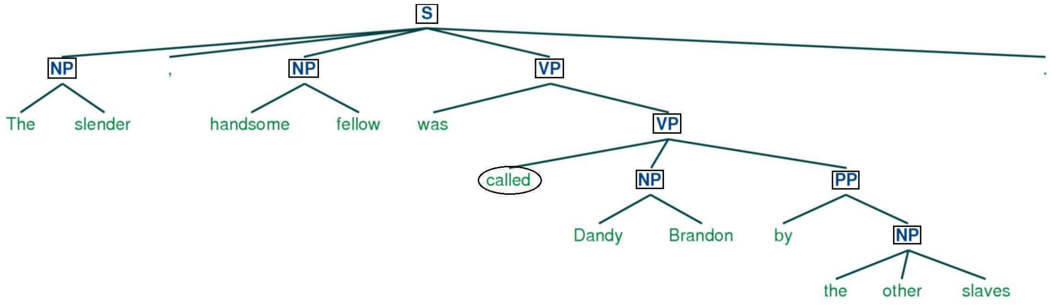

Figure 1: Basic parse tree feature example

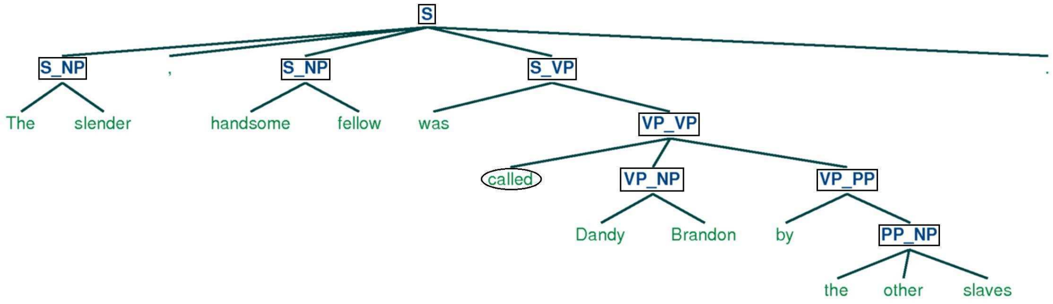

Figure 2: One level parent annotated parse tree

Last word of each chunk with the chunk type The list that contains the last word of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **him_of_door_through_which_he_came**.

First POS tag of each chunk with the chunk type The list that contains the first POS tag of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **PRP_IN_DT_IN_WDT_PRP_VBD**.

Last POS tag of each chunk with the chunk type The list that contains the last POS tag of each chunk will be treated as a separate feature. With the *kick* example, this feature would be **PRP_IN_NN_IN_WDT_PRP_VBD**.

Chunk-type-sensitive combinations This feature is created by merging the chunk types and some additional information associated with the chunks. If a chunk is a PP, then the head preposition will also be part of the feature; if a chunk is an NP and the head word is a question word (*who*, *what*, *when*, *how*, *where* or *why*), the head word itself will be part of the feature, but if the head word is not a question word, its POS tag will be part of the feature; if a chunk is a VP, then the POS tag of the head verb will be part of the feature. Using the above *kick* example, this feature will be: **(NP, PRP)_(PP, out)_(NP, NN)_(PP, through)_(NP, which)_(NP, PRP)_(VP, VBD)**.

3.2 Selectional Preference Based Features

We used the ASSERT[5] system (Hacioglu et al., 2004) to extract the semantic roles from the target sentences. The following selectional preference based features have been experimented with:

WordNet synsets of the head nouns of the SRs For each semantic role, the WordNet synsets of its head noun, paired with the corresponding semantic role.

Semantic role's relative positions These features are designed to capture the syntactic patterns of the target verb and its semantic roles. The relative position is set up such that the first semantic role to the left of the target verb will be given the position of -1, and the first one to the right will be given the position of $+1$.

Lemmatized head noun of each semantic role Similar to the synset features, each semantic role is also paired with its head noun.

[5] We used version 1.4b of this system which can be downloaded from `http://oak.colorado.edu/assert/`

Preposition of the adjunct semantic role If there is an adjunct semantic role which is also a prepositional phrase, the preposition is also paired with the semantic role.

4 Evaluation Datasets

We used two datasets based on distinct text genres to examine the effectiveness of the multi-semantic-role based selectional preferences: the verb dataset from the English lexical-sample subtask of SENSEVAL-2, and a manually sense tagged movie script corpus (MSC).[6] The SENSEVAL-2 data contains 28 polysemous verbs, and 3564 training instances. The movie script corpus contains 8 sense labelled verbs and 538 training instances. Table 1 outlines the composition of the movie script corpus dataset.

The sense tagged movie script corpus is important because it is an integral part of a broader NLP project which aims to generate computer animation by interpreting movie scripts. Since most of the actions in movie scripts are described by verbs, we believe it is necessary to investigate whether knowing the senses of the verbs can improve the accuracy of the animation generation.

The movie script corpus was hand-tagged by two annotators according to WordNet 2.0 senses, and the differences in the annotation were arbitrated by a third judge. Compared with the SENSEVAL-2 data, which comes from the Brown Corpus, the sentences in the movie script corpus tend to be shorter because they describe certain actions to be performed in the movie. Example sentences from this corpus include the following:

1. A rubber dart $\underline{hits}$ the glass and drops into a trash can next to the door .
2. Neo slowly sets down his duffel bag and $\underline{throws}$ open his coat, revealing an arsenal of guns, knives, and grenades slung from a climbing harness.
3. Morpheus tries to $\underline{look}$ not sad.

5 Feature Selection Algorithms

It has been observed that combining certain features together can lead to a decrease in classification accuracy, hence a feature selection process is deemed necessary. Due to the high numbers of individual binary features and feature types, it would be impractical to generate all possible combinations of the individual features or feature types. Therefore, we propose two automatic feature selection algorithms here: the individual feature ranking algorithm and feature type coverage algorithm.

5.1 Individual Feature Ranking Algorithm

This algorithm is based on the work of Baldwin and Bond (2003) and works by first calculating the information gain (IG), gain ratio (GR) and Chi-square statistics (Chi) for each binary feature as 3 separate scores. Then, each score is separately used to rank the features in a way such that the greater the score, the higher the rank. Features which have the same value for a particular score are given the same rank. Once individual ranks have been determined for each feature, the ranks themselves are summed up and used as a new score which is then used to re-rank all the features one last time. This final ranking will be used to perform the feature selection.

Once the final ranking of the features has been calculated, we then generate separate feature sets using the top N percent ranked features, where N ranges from 0 to 100 with an increment of 5.[7] We evaluate these feature sets in Section 2.

5.1.1 Feature Type Coverage Algorithm

The aim of this algorithm is to use the minimal number of feature **types** to generate the best performance. It works in the following way. First, we assign a unique ID to every training instance in the original training set. We then create a separate dataset for each individual feature type (e.g. **Bag Of Words**, **Left Words**, etc.) and evaluate them as per Section 2. Since the single-feature-typed datasets are all created from the same original training set, we can propagate the IDs of the original training instances to the testing instances in the held-out sets of the single-feature-typed datasets. Furthermore, as the CVs are stratified, we can calculate the accuracy of each feature type with respect to each training instance. For example, suppose the $10th$ training instance for the verb *hit* was correctly classified in the held-out set by the classifier trained with only the **verb-argument structure** features, then the accuracy

[6] The entire MSC dataset contains more than 250 movie scripts, but due to limited resources, only 3 scripts were sense tagged, and only 8 high frequency and highly polysemous verbs were chosen for this study.

[7] The top 0% feature set is replaced by the majority-class heuristic

Verb	Freq. in Sense Tagged Corpus	Freq. in entire corpus	No. of Senses in Sense Tagged Corpus	Majority Class Ratio	Inter Annotator Agreement
hit	32	1770	6	.438	.828
lift	31	1051	4	.548	.807
look	164	19480	5	.884	.963
pull	79	5734	7	.410	.744
stop	43	3025	4	.524	.917
take	54	8277	14	.370	.679
throw	29	1940	6	.379	.724
turn	106	8982	10	.736	.953

Table 1: Movie script corpus details

of **verb-argument structure** feature type on this particular training instance would be 1.0. With these training-instance-specific accuracies, we define the **coverage** of a feature type as a mapping from training instance IDs to their individual accuracies. We also use the sum of the accuracies of the individual data instance to assess the overall coverage of particular feature type with respect to a training set.

In order to assess the coverage of combined feature types, we define an additional procedure called *combine* for merging two coverages together to produce a new coverage. The details of this algorithm are shown in Algorithm 1.

On top of the coverages, we also calculate the **applicability** of each feature type as the percentage of held-out instances for which one or more features of that particular type can be extracted. The final phase of the feature selection algorithm is a process of greedily combining the coverages of feature types together until the coverage plateaus to a local maximum. This process is described in Algorithm 2.

6 Results and Discussion

For the SENSEVAL-2 data, the feature selection algorithms were applied on the 10-fold cross validated training set and the chosen features type and the Gaussian smoothing parameters were applied to the test set. For the movie script corpus, the feature selection algorithms were applied to 5-fold nested cross validation of the entire dataset. The final cross validation results are reported here.[8] Furthermore, in order to measure the usefulness of the feature selection algorithm, we have also included results obtained using "Oracled" feature sets and Gaussian prior values which were tuned with respect to the test data. More specifically, we collected the following information for our evaluation:

Fully Oracled Using both oracled feature sets and oracled Gaussian prior values.

FS Oracled Using oracled feature sets but automatically selected Gaussian prior values.

Maxent Oracled Using automatically selected feature sets and oracled Gaussian prior values.

Fully Auto. Using both automatically selected feature sets and Gaussian prior values.

All Features Including all features and using automatically selected Gaussian prior values.

Tables 2 and 3 respectively summarize the evaluation results on the datasets with and without SRL features.[9]

It can be observed that the impact of the feature selection algorithms on the SENSEVAL-2 dataset is similar to that on the MSC dataset. The feature ranking algorithm seems to perform noticeably worse than having no feature selection at all, but the coverage algorithm seems perform mostly better. This shows that feature selection can be a useful process irrespective of the corpus.

The disappointing performance of the feature ranking algorithm on both datasets is caused by the mismatch between the training and the testing data. Recall that this algorithm works by selecting the top N percent of features in terms of their disambiguation power. Since the feature selection is only performed on the training set, the chosen features could be absent from the test set or have different distributions with respect to the verb senses. Verb-by-verb analysis for this algorithm revealed

[8] In nested cross validation, the training set of each fold is further divided into a sub-training and sub-held-out set, and the feature selection and Gaussian smoothing parameters for the proper held-out sets are tuned on a fold-by-fold basis.

[9] The majority class baseline for the MSC dataset is gathered from the primary held-out sets of the nested CV, therefore it is potentially different to the majority class of the entire MSC dataset.

Algorithm 1 The algorithm of *combine*, which merges two coverages to produce a new coverage.

```
Let $I$ be the set of IDs of the training instances
Let $Coverage_1$ and $Coverage_2$ be the two coverages to be merged
Let $NewCoverage$ be the combined coverage of $Coverage_1$ and $Coverage_2$
for $i \in I$ do
    $NewCoverage[i] = \max(Coverage_1[i], Coverage_2[i])$
end for
Return $NewCoverage$
```

Algorithm 2 The incremental process of determining the final set of feature types

```
Let $I$ be the set of IDs of the training instances
$CurrentCoverage = \{(i \rightarrow 0.0) | i \in I\}$
Let $F$ be the set of feature types
$Combine(coverage_1, coverage_2) = \{(i \rightarrow max(coverage_1[i], coverage_2[i])) | i \in I\}$
while True do
    $NCs \leftarrow \{\}$        ▷ Initialize a temporary list to hold the new combined coverages
    for $f_i \in F$ do
        $NewCoverage_i \leftarrow Combine(CurrentCoverage, Coverage_{f_i})$
        Add $NewCoverage_i$ to $NCs$
    end for
    Let $NewCoverage^\star \in NCs$ be the one with highest overall coverage.        ▷ Break tie with the lowest applicability.
    if $CurrentCoverage$ has the same overall coverage as $NewCoverage^\star$ then
        Break
    end if
    $CurrentCoverage \leftarrow NewCoverage^\star$
end while
```

Dataset	Individual Feature ranking					Feat. Coverage		Majority
	Fully	FS	Maxent	Fully	All	Maxent	Fully	Class
	Oracled	Oracled	Oracled	Auto.	Features	Oracled	Auto.	Baseline
SENSEVAL-2	.623	.615	.556	.540	.574	.588	.583	.396
MSC	.774	.743	.602	.577	.690	.712	.712*	.617

Table 2: Disambiguation accuracies with SRL features (* indicates significantly higher performance than the all features baseline (paired t-test, $p > 90\%$))

Dataset	Individual Feature ranking					Feat. Coverage		Majority
	Fully	FS	Maxent	Fully	All	Maxent	Fully	Class
	Oracled	Oracled	Oracled	Auto.	Features	Oracled	Auto.	Baseline
SENSEVAL-2	.606	.595	.544	.529	.558	.583	.576	.396
MSC	.780	.747	.554	.532	.714	.721	.693	.617

Table 3: Disambiguation accuracies without SRL features

that for most verbs, there was very little correlation between the training data and the test data in terms of the value of N verses the disambiguation performance. This is why this algorithm consistently selected suboptimal feature sets which resulted in the poor performance. This mismatching problem is especially severe for the MSC corpus which contains many verb senses that occur only once.

On the other hand, the performance of the coverage based algorithm tends to be similar to or better than that of using all the features. Recall that this algorithm selects feature types in such a way that the coverage of the final feature set is maximized. As a result, even feature types which performed poorly on the training set may be selected as long as they performed well on some of the training instances that other features performed poorly on. Therefore, the features chosen by this algorithm are less likely to overfit the training data.

Overall, it can be observed that for both datasets, most of our automatic classifiers outperform the majority class baseline, which is very encouraging. For the SENSEVAL-2 dataset, the classifiers with SRL features consistently outperform those without. However, for the MSC dataset, the results of the nested cross validation showed that the performance of the automatic classifiers with the SRL features does not consistently outperform those without the SRL features; the oracled classifiers constructed without the SRL features actually consistently outperformed those with the SRL features.

The differences between the results obtained from the two datasets make it difficult to conclude categorically whether SRL can indeed help VSD. However, a closer examination of the datasets reveals that errors in the output of the semantic role labeler, the intransitivity of the verbs, unresolved anaphors, and verbal multiword expressions (verbal MWEs) are the main reasons for the lack of positive contribution from the SRL features.

Most of the verbs on which SRL features performed poorly are intransitive, which means that only one argument type semantic role is available – the subject or the agent role, which mostly occurs to the left of the verb. However, the feature ranking algorithm showed that most of the useful features occur to the **right** of the verb, which is why SRL features tend to perform poorly on intransitive verbs.

The unresolved anaphors also limited the effectiveness of SRL features because they carry almost no disambiguation information, no matter which semantic role they take, and they occur in a significant number of sentences. The anaphor problem is slightly more pronounced in the movie script corpus, because its texts tend to describe consecutive actions performed by the same actor(s) and involving the same objects in the scene, and therefore anaphors tend to occur more frequently.

Verbal MWEs such as *take off* are not detected as a single lexical item, and the verbs themselves tend to have no suitable sense as far as WordNet is concerned. However, they often occur more than once in the data, and since the annotators were forced always to pick at least one sense, these expressions tend to end up as noise in the data.

Finally, it is also possible that the lack of training data in the MSC corpus contributed to the poor performance, since almost every verb in the MSC corpus contains two or more senses which occur less than twice.

7 Conclusions and Future Work

In this paper, we have presented our research on using multi-semantic-role based selectional preferences obtained from a state-of-the-art semantic role labeler. We have shown that this particular type of selectional preferences can indeed improve the performance of verb sense disambiguation. However, this improvement still depends on the performance of the semantic role labeler, the transitivity of the verbs, the resolution of anaphors, and the identification of verbal MWEs.

In future research, we hope to focus on integrating more competent anaphora resolution systems and verbal MWE detectors into our existing VSD framework, and investigating how to mitigate the errors in the semantic role labeler.

Acknowledgements

We would like to thank the three anonymous reviewers for their valuable input on this research. The research in this paper has been supported by the Australian Research Council through Discovery Project grant number DP0663879, and also NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation.

References

Timothy Baldwin and Francis Bond. 2003. Learning the countability of english nouns from corpus data. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 73–80, Sapporo, Japan.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Eugene Charniak. 2000. Maximum-entropy-inspired parser. In *Proceedings of the First Conference on North American Chapter of the ACL*, pages 132–139, San Francisco, USA.

Trevor Cohn, Andrew Smith, and Miles Osborne. 2005. Scaling conditional random fields using error-correcting codes. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 10–17, Ann Arbor, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Jess Gimnez and Llus Mrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46, Lisbon, Portugal.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of the 8th Conference on Natural Language Learning (CoNLL-2004)*, pages 110–113, Boston, USA.

Anna Korhonen and Judita Preiss. 2003. Improving subcategorization acquisition using word sense disambiguation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 48–55, Sapporo, Japan.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654, December.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL SIGLEX Workshop on Tagging Text with Lexical Semantics, Why, What and How?*, pages 52–57, Washington, USA.

Erik Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99: Ninth Conference of the European Chapter of the ACL*, pages 173–179, Bergen, Norway.

David Yarowsky, 2000. *Handbook of Natural Language Processing*, chapter 26. Marcel Dekker.