# An Evaluation Of Predicate Argument Clustering Using Pseudo-Disambiguation

**Christian Scheible**

Institute for Natural Language Processing
Azenbergstr. 12, 70174 Stuttgart, Germany
scheibcn@ims.uni-stuttgart.de

**Abstract**

Schulte im Walde et al. (2008) presented a novel approach to semantic verb classication. The predicate argument model (PAC) presented in their paper models selectional preferences by using soft clustering that incorporates the Expectation Maximization (EM) algorithm and the MDL principle. In this paper, I will show how the model handles the task of differentiating between plausible and implausible combinations of verbs, subcategorization frames and arguments by applying the pseudo-disambiguation evaluation method. The predicate argument clustering model will be evaluated in comparison with the latent semantic clustering model by Rooth et al. (1999). In particular, the influences of the model parameters, data frequency, and the individual components of the predicate argument model are examined. The results of these experiments show that (i) the selectional preference model overgeneralizes over arguments for the purpose of a pseudo-disambiguation task and that (ii) pseudo-disambiguation should not be used as a universal indicator for the quality of a model.

## 1. Introduction

Semantic verb classification is an important task in computational linguistics. There have been various approaches to this problem, and there are many applications which can benefit from semantic verb classes. The intention of semantic verb classification is to provide an abstraction over verbs that share semantic properties. For example, the verbs *say*, *tell*, *talk*, and *ask* could be in a class which contains verbs about verbal expression.

Schulte im Walde et al. (2008) presented a new kind of semantic verb classification model. This model combines Expectation Maximization (EM) based clustering with a selectional preference model with the capability of abstracting over argument nouns. One major problem of semantic verb clustering is data sparseness, since the stochastic model is trained with data extracted from text corpora. The predicate argument clustering model (PAC) aims to tackle this problem by modeling selectional preference using WordNet (Miller et al., 1990) and the Minimum Description Length (MDL) principle (Rissanen, 1978). The idea of a clustering algorithm with selectional preferences is based on the notion that verbs belonging to the same semantic class have a similar set of selectional preferences.

The task of evaluating such a model can be accomplished by techniques which are commonly used to evaluate different kinds of verb-argument classification problems. The particular method which I will employ is called pseudo-disambiguation in which an algorithm is tested for its ability to discriminate between semantically plausible and implausible input. The goal of this paper is to describe the process of evaluating the PAC model in comparison with latent semantic clustering (LSC), its predecessor, using pseudo-disambiguation.

## 2. Background

### 2.1. Latent Semantic Clustering

Rooth et al. (1999) presented an approach for the automatic semantic classification of interrelated dimensions, for example a verb and its arguments. They use a soft clustering model which is also the foundation of the PAC model.
The model clusters a given set of verbs-arguments pairs using the EM algorithm (Baum, 1972). The authors use a clustering model based on the intuition that hidden information about semantic classes is contained in the training data. Each cluster is intended to contain verbs that share semantic properties and thus constitute a semantic class.
The model implicitly provides smoothing for unseen combinations of verbs and arguments since words from each dimension dimensions (verbs and nouns in the original experiment) can be combined independently. In order to evaluate their model, the authors carry out a pseudo-disambiguation task.
In the experiment described by the authors, the model is trained with verb-argument data extracted from a text corpus. Table 1 contains some example data. Each of these verb-argument tuples consists of a verb (e.g. *abandon*), its subcategorization frame (e.g. *SUBJ:NP* with a subject and an NP which is the direct object), a slot of the frame (e.g. *SUBJ*) and its nominal argument (e.g. *Australia*). In the original LSC experiments, the frame, the slot, and the argument are combined.

| Verb | Frame | Function | Argument |
|---|---|---|---|
| abandon | SUBJ:NP | SUBJ | Australia |
| give | SUBJ:NP | NP | detail |

Table 1: Example input for LSC.

In the resulting model, verbs and nouns are considered to be independent given the cluster $c$, and their co-occurrence is represented solely by their affiliation with the same cluster. The probability of a tuple $\langle v, n_1, ..., n_k \rangle$ is accordingly defined as

$$p(v, n_1, ..., n_k) = \sum_c p(c)\, p(v|c)\; * \prod_{i=1}^{k} p(n_i|c) \qquad (1)$$

where $c$ is a cluster, $v$ a verb and $n_1, ..., n_k$ nouns and their subcategorization functions. The number of clusters has to be specified for each model. The original version of LSC [1] only has two dimensions, representing verbs and nouns, respectively, in the original experiments. This can easily be extended to an arbitrary number as shown above. However, although the number of dimensions $k$ is now variable, it is still fixed for a single model instance.
Clustering is achieved using the EM algorithm. The details of EM training are given in (Rooth et al., 1999).
LSC was evaluated using pseudo-disambiguation. Models were trained with 25 to 100 clusters and 50 EM iterations and yielded an accuracy of approximately 80%. For models with more than 100 clusters, an overfitting effect occurred.

### 2.2. Predicate Argument Clustering

The PAC model extends the idea of LSC by incorporating information about selectional preference using the MDL principle to generalize over arguments. The model is capable of handling subcategorization frames with arbitrary numbers of slots.

| Verb | Frame | Arguments | | |
|---|---|---|---|---|
| give | SUBJ:NP | rule | consumer | |
| happen | SUBJ | incident | | |
| walk | SUBJ:P:NP | worker | off | job |

Table 2: Example input for PAC.

Each verb-argument tuple now consists of a verb (e.g. *give*), its subcategorization frame (e.g. *SUBJ:NP*) consisting of separate slots, and an argument for each slot (e.g. *rule* and *consumer*). *SUBJ:P:NP* is an example of a frame that contains a subject and a PP which consists of a preposition and a noun.

#### 2.2.1. Probabilistic Model

The following equation extends the definition of the LSC model (Equation 1) to take subcategorization frames and selectional preference into account:

$$p(v, f, a_1, ..., a_{|f|}) =$$
$$\sum_c \left[ p(c)\, p(v|c)\, p(f|c) \times \prod_{i=1}^{|f|} \sum_{r \in R} p(r|c, f, i)\, p(a_i|r) \right] \qquad (2)$$

It describes the generation of a verb-argument tuple consisting of a verb $v$, a subcategorization frame $f$ with $|f|$ slots and a argument for each of these slots. Its probability is determined by (i) calculating the product of the probabilities $p(a_i|r)$ of each possible path between each argument $a_i$ and the selectional restriction $r$ in the a-priori model $R$ and the probability $p(r|c, f, i)$ of a selectional restriction $r$ given the cluster $c$ and the $i$th slot in the subcategorization frame $f$ in the selectional restriction model (which will be defined below), (ii) multiplying them with the conditional probabilities of the verb $v$ and the frame given the cluster, respectively and (iii) calculating the sum over all clusters. A description of the a-priori and selectional preference models as well as the details of the training algorithm will be given in the following sections.

#### 2.2.2. EM Training

The probability parameters of the model can be determined by using the EM algorithm. Since there are two types of hidden variables now, namely the clusters and the selectional restrictions, we need to deviate from the original training method.
The model describes a stochastic process which generates the training data. This process can be expressed through a PCFG which generates the verb-argument tuples. Thus the Inside-Outside algorithm (Lari and Young, 1990) can be applied to train our clustering model. Rules for a PCFG can be obtained as given in Schulte im Walde et al. (2008):

- The start symbol is TOP.
- For each cluster $c$, add TOP $\rightarrow$ V$_c$ A$_c$ with probability $p(c)$.
- For each word in cluster $c$, add V$_c$ $\rightarrow$ $v$ with probability $p(v|c)$.
- For each subcategorization frame $f$ of cluster $c$ with length n, add $A_c \rightarrow f R_{c,f,1,\text{entity}} \ldots R_{c,f,n,\text{entity}}$ with probability $p(f|c)$.
- For each transition from a node $r$ to a node $r'$ in the selectional preference model for slot $i$ of the subcategorization frame $f$ of cluster $c$, we add a rule $R_{c,f,i,r} \rightarrow R_{c,f,i,r'}$ whose probability is the transition probability from $r$ to $r'$ in the respective WordNet-HMM.
- For each terminal node $r$ in the selectional preference model, we add a rule $R_{c,f,i,r} \rightarrow R_r$ whose probability is 1. With this rule, we transfer from the selectional restriction model to the corresponding node in the a-priori model.
- For each transition from a node $r$ to a node $r'$ in the a-priori model, we add a rule $R_r Rr'$ whose probability is the transition probability from $r$ to $r'$ in the a-priori WordNet-HMM.
- For each word node $a$ in the a-priori model, we add a rule $R \rightarrow a$ whose probability is 1.

A single tuple can be represented as a parse using this grammar. In our example tree (Figure 1), we generate the tuple ⟨give, SUBJ:NP, consumer, rule⟩. We chose cluster 3 and two paths through the selectional preference and a-priori models, one for each argument, respectively. At $R_{abstract}$ and $R_{person}$, we transfer from the selectional preference into the a-priori model. After choosing a path through the a-priori model, we finally arrive at a terminal node which contains an argument.

#### 2.2.3. WordNet as a Markov Model

PAC includes a model for selectional preferences which learns them by representing classes of preference through nodes from WordNet (Miller et al., 1990). For example, if words like *coffee*, *tea*, and *milk* occur frequently in a slot, the model may choose a node like *beverage* to represent them. All nodes below *beverage* are removed. The model

[1]I used an implementation written by Helmut Schmid, which is freely available at `http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LSC.html`.

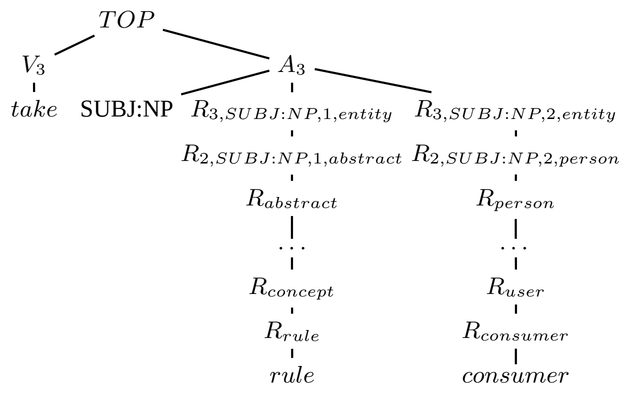

Figure 1: Example parse tree.

determines the set of nodes that represents the training data best during EM training. These selectional preference models are built separately for each slot of each subcategorization frame. Abstraction over nouns can help in sparse-data situations where previously unseen combinations of verbs and arguments occur.

Following Abney and Light (1999), WordNet is first turned into a Markov model. WordNet can be viewed as a graph consisting of non-terminal (concepts) and terminal nodes (words). In the Markov model, a node $s_x$ exists for each node $x$ of the WordNet graph. Since some versions of WordNet do not have a unique top node, an artificial one is inserted which has all WordNet top nodes as hyponyms. The transition probability from a node $s_x$ to $s_y$ is non-zero iff $x$ is a hyponym of $y$ or $x$ is a word in synset $y$. The probabilities of all links from a node to his hyponyms sum up to 1.

There exists one general instance of this model called the **a-priori model**. In addition, each slot of each subcategorization frame has a WordNet Markov model called the **selectional preference models**. These models only consist of partial copies of WordNet to reflect the idea of generalization presented above. At their terminal nodes, these models have links to the corresponding hyponym nodes in the a-priori model. The terminal nodes are determined according to the MDL principle (described in Sections 2.2.4. and 2.2.5.). The set of all terminal nodes in a selectional preference model is called a **cut**.

### 2.2.4. MDL principle

The MDL principle (Rissanen, 1978) states that the quality of a model can be improved by minimizing its **description length**. The core idea behind this assumption is that the model that can optimally compress the training data is its best representation. The more regularities and redundancies are detected, the higher the degree of compression. This section gives an outline of the MDL principle.

The description length is composed of the **model length** and the **data length**. The model length is the number of bits which are necessary to represent the model, including its parameters. The data length in turn is the number of bits which are needed to describe the training data by using the current model. The model length (ML) is defined as

$$\text{ML} = \frac{k}{2} * log_2|S|. \tag{3}$$

$k$ is the number of concepts in the selectional preference models above and including the cut, and $|S|$ is the size of the sample (the sum of the frequencies of training data). The data length (DL) is the number of bits necessary to encode the training data using the current model. It can be calculated as

$$\text{DL} = -\sum_{y \in S} log_2 p_\theta(y), \tag{4}$$

with the probability of a noun being dependent on the probability of the WordNet concept it belongs to.
The description length is then defined as

$$L_\text{d} = w * ML + DL, \tag{5}$$

where $w$ is a weight factor that can put an emphasis on either the model length or the description length. The standard value of $w$ is 1, meaning that model length and description length are equally important. If $w$ is set to a larger value, the model length gets penalized more, which leads to the selectional preference models becoming smaller in depth, and vice versa.

### 2.2.5. EM training and the MDL principle

In Section 2.2.3. I described the relationship between the a-priori and the selectional preference models. Terminal nodes of the selection preference model have links to nodes corresponding to their hyponyms in the a priori model. The EM algorithm is designed to fit the training data. Since detailed selectional preference models describe the training data better than shallow ones, the EM algorithm prefers them, thus the selectional preference models grow during training and overfitting effects occur. Since we want to maintain a degree of abstraction, we will restrict the growth of the models by applying the MDL principle.

The MDL principle can be integrated into EM training through the following steps. When the training begins, all selectional preference models consist only of the top node. In each iteration, the cuts of the models are redefined. First, we expand each node in a cut towards its hyponyms. Then, the Inside-Outside-Algorithm is applied. Afterwards, starting at the terminal nodes of the model, we decide for each node if keeping it in the selection is beneficial. This decision is made by comparing the description lengths of the model with and without the node. If keeping the node increases the description length, it is pruned from the model. This method differs from the one described by Schulte im Walde et al. (2008) in that nodes can now be pruned individually as opposed to pruning all nodes in a cut. Finally, as a last step, the probabilities are maximized.

## 3. Pseudo-Disambiguation

Pseudo-disambiguation is a method that tests if a model is capable of distinguishing between semantically probable and improbable inputs. It was developed as a way to circumvent the lack of properly annotated data for the evaluation of word sense disambiguation algorithms. In this paper, it will be applied to evaluate whether the verb clusterings induced by the PAC model represent semantic properties appropriately.

## 3.1. Definition of the Task

The main idea behind pseudo-disambiguation is to create semantically implausible data from existing real data by replacing one word in a verb-argument tuple with other elements that are equally frequent but not semantically related. There is however no rule on how these replacements have to be chosen. In this paper, only verbs were exchanged.

Pseudo-disambiguation has first been used to evaluate word sense disambiguation (WSD) algorithms. The word *suit* for example is ambiguous, and a conceivable subset of labels for its readings could for example consist of {*clothing, law, card game*}. In a typical WSD task, an ambiguous word is presented in the context of a sentence, for example "*He wears a black suit.*" The test whether a model is capable of disambiguating the word senses of *suit* is to have the model choose the sense from the aforementioned set of labels. The correctness of such an assertion can however only be tested if a set of test sentences annotated with these labels is available. Creating such a set involves manual labelling and is thus time-consuming.

Due to this problem, Schütze (1992) and Gale et al. (1992) created a different WSD task where the model will be presented with an artificially ambiguous word which will take the place of *suit*. This word is called a **pseudo-word** and is created by combining *suit* with an equally frequent word from a different semantic class, for example *banana*, resulting in *banana-suit*. Each occurrence of *suit* and *banana* in the test corpus will then be replaced with this pseudo-word, and the WSD algorithm has to find out which of the two words was replaced in each case. With this method, it is possible to create a test set automatically with no manual labeling being involved.

Following Rooth et al. (1999) the method is applied to verb-argument tuples. Here, the model is presented two word sequences in which one of the places, in this case the verb, is ambiguous, for example:

*read man book*

*fly man book*

The word pair that produces the ambiguity, in this case $\langle \text{read}, \text{fly} \rangle$ will be called a pseudo-word. The second sequence is less probable than the first sequence from a semantical point of view. The model has to find out which of the two words was the original one, thus its ability of recognizing semantically plausible input is tested.

## 3.2. Data Acquisition

The data used for my experiments was acquired using BitPar (Schmid, 2004) which had been trained on the Penn Treebank (Marcus et al., 1993). We parsed the Reuters newswire corpus (Rose et al., 2002) which produced 4882100 Viterbi parses.

From these, we extracted $2{,}375{,}359$ verb-argument tuples ($1{,}208{,}039$ types), where token means a single occurrence of a verb-argument tuple in the corpus, type means all of its occurrences. Each tuple consists of a verb, its subcategorization frame and the lemmatized lexical heads of the arguments for each of the frame's argument slots. I looked only at active forms and I discarded all tuples containing words which were not included in WordNet 3.0 (Miller et al., 1990) which was used for the selectional preference model. To eliminate noise, I removed all tuples which occurred only once or contained a verb, a subcategorization frame, or a noun that did not appear more than once. However, I also conducted experiments containing singulars due to an argument by Dagan et al. (1999) that it was possible to build models that include such data without any loss of quality.

## 3.3. Generation of Pseudo-Words

The verb-argument tuples were split into 90% training and 10% test tuple tokens; each occurrence of a tuple was counted as a separate token. I used this segmentation for all following experiments except for the frequency variation experiments. I removed tuples containing any of the 25 most frequent verbs (*represent, show, see, follow, seek, use, raise, make, support, take, include, boost, buy, hit, get, reach, reflect, produce, meet, report, hold, cover, face, become, create*) from the test corpus since the differences between their frequencies were too high which could lead to a bias towards the more frequent replacement word.

In order to generate pseudo-words, each verb was assigned to word with the next higher corpus frequency as its replacement (Table 4). This way I could assure that both the original type and its replacement were roughly equally frequent. This process is also fairly random, as it is not likely that two words that are paired up are semantically related.

| | |
|---|---|
| use (5041) | approve (5102) |
| create (5039) | use (5041) |
| become (4722) | create (5039) |
| support (4652) | become (4722) |
| increase (4574) | support (4652) |

Table 3: 5 most frequent pseudo-words for the Reuters corpus, frequency counts in brackets.

All in all, 2790 pseudo-words were created. Table 3 lists the ten most frequent ones, where the left word is replaced by the right word. Table 4 shows how these words are then used to create fake tuples. For each tuple in the test set I created a fake tuple by replacing the verb with the corresponding partner in the pseudo-word. Fake tuples that already appeared in the test corpus as a real tuple were removed.

| | | | |
|---|---|---|---|
| cut | SUBJ:NP | Congress | budget |
| increase | SUBJ:NP | Congress | budget |
| support | SUBJ:NP | politician | bill |
| become | SUBJ:NP | politician | bill |

Table 4: Random tuples and their corresponding fake tuples.

## 3.4. Experiments

I set up multiple tasks which differed in the way of selecting the test set and of pre-processing the data, as well as in the choice of model parameters. In these experiments, each tuple and its pseudo-tuple were assigned likelihood values by each of the particular models. Likelihood values were calculated as defined in Equation 1 for LSC and in Equation 2 for PAC.

In order to determine accuracy, I compared the likelihoods of each tuple and its fake tuple. If the likelihood of the original tuple was greater than the one assigned to the fake tuple, the choice was counted as correct. Each tie was counted as $0.5$, which simulates a random choice of correct or wrong. According to these rules, I formalized the accuracy $A$ as $(\text{number of correct choices} + 0.5 \times \text{number of ties})/N$. $N$ denotes the overall number of tuples in the test corpus.

Models, if not noted otherwise, were trained with 50 clusters and 50 iterations, following the results of the evaluation of LSC (Rooth et al., 1999) and PAC (Schulte im Walde et al., 2008) and the experiments in Section 3.4.5..

### 3.4.1. Repeating the Experiment from Rooth et al.

The first experiment repeats the original LSC evaluation. Here, relations between arguments of subcategorization frames were not modeled, which was impossible due to the dimension number constraint in LSC. Instead, only the co-occurrence of a verb, a subcategorization frame, one argument slot and one argument was included in the model. I transformed our data sets into the format that was used to evaluate LSC by creating a new tuple for each subcategorization function in each tuple. For example,

*give SUBJ:NP Congress budget*

yielded two new tuples:

*give SUBJ:NP-SUBJ–Congress*
*give SUBJ:NP-NP–budget*

This transformation produced 2,669,727 verb-argument pairs. I trained both LSC and PAC models with this data. For PAC, I trained two versions of the model, one that uses the selectional preference model and one that does not. In this experiment, LSC has an accuracy of 91.27%, and PAC has 83.49%. When selectional preference models are not used in PAC, the accuracy reaches to 91.72%. This suggests that the abstractions made by the MDL model cause problems.

### 3.4.2. Subjects and Objects

Since LSC needs a fixed number of dimensions, experiments with the complete data set are impossible. Thus, I decided to carry out an experiment in which only tuples containing the most frequent frame (*SUBJ:NP*, i.e. a subject and a direct object) are used. This leaves $488{,}665$ tuples in the corpus. In order to process this data with LSC, subcategorization frame information had to be removed.

LSC achieved an accuracy of 93.08%, PAC 74.32%. As mentioned earlier, one reason for the inferior accuracy of the PAC model could be the degree of abstraction in the selectional restriction models. Manual examination of the models revealed another source of problems related to clustering. The clusters turned out to be inhomogeneous to a certain degree. For example, different tuples containing the verb *to alarm* are placed into various clusters although there is no ambiguity. This causes all tuples with this verb to be identified as fake. There are more verbs of this kind that lead to the same problem, for example *rationalise*, *devote*, *persist*, *rain*, *disrupt*. The clusters to which these verbs are assigned often either contain many common-place verbs like *make*, *take*, *show*, and *buy* or consist of semantically unrelated verbs. Most of the aforementioned examples are categorized correctly by LSC. On the other hand, PAC performs well on very high-frequent verbs in the training corpus.

### 3.4.3. Complete Data

For the next experiment, I used training and test data containing other frames than *subject-object*, which consisted of 1,414,360 tuples. LSC allows only a fixed number of arguments, so these experiments could only be carried out with a PAC model. To test the claim by Dagan et al. (1999) that including singular events in the corpus could improve a model, I trained another model where tuples with a frequency of 1 were not removed. This data contained 2,375,359 tuples. In order to find a baseline to which we can compare our results, we define a simple back-off method that decides the question of a tuple being real or fake only based on frequency counts $c$ from the training data. This algorithm, given in pseudocode in Figure 2, decides which tuple $\langle v, f\rangle$ is the original based on a back-off scheme. If at least $v$ or $v'$ and $f$ were observed together, their joint frequency is used for the decision, otherwise the frequencies of the overall occurrences of $v$ and $v'$ were used.

```
if $c(\langle v, f\rangle) > 0$ or $c(\langle v', f\rangle) > 0$ then
    if $c(\langle v, f\rangle) > c(\langle v', f\rangle)$ then
        return $v$
    else
        return $v'$
    end if
else if $c(v') > c(v)$ then
    return $v'$
else
    return $v$
end if
```

Figure 2: Baseline algorithm based on verbs subcategorization frames.

Note that this method only uses co-occurrence data of verbs and frames and does not consider arguments at all. In these experiments, PAC achieved an accuracy of 89.71% (80.64% with singletons), the baseline achieved 83.0%.

First, accuracy was higher when singletons were excluded. Possible reasons for this will be examined in the following sections. Second, the frequency-based baseline method achieves a fairly good result even though using less data than the PAC models. This suggests that knowledge about subcategorization frames is sufficient in this kind of task, and that those methods perform well at pseudo-disambiguations which can fit the training data, which is the case with a frequency-based method. In the next section I will propose a baseline method that is able to beat both LSC and PAC under certain conditions.

To understand the impact of frames on accuracy, Table 5 shows accuracy values for different frames. We can see that

*SUBJ:NP* is actually the frame with the lowest accuracy value, while tuples containing less frequent frames tend to have higher accuracies. It might be easier for PAC to discover fake tuples with these frames for the same reasons it is for the baseline method.

| Frame | Frequency | Accuracy |
|---|---|---|
| SUBJ:NP | 40,440 | 84.53 |
| SUBJ | 16,120 | 91.64 |
| SUBJ:P:NP | 9,749 | 94.64 |
| SUBJ:S | 8,929 | 94.60 |
| SUBJ:SBAR | 7,310 | 96.60 |
| SUBJ:NP:P:NP | 5,351 | 87.42 |
| SUBJ:PRT:NP | 1,781 | 94.83 |
| SUBJ:ADJP-PRD | 1,021 | 98.04 |
| SUBJ:NP:S | 967 | 99.58 |
| SUBJ:PRT | 958 | 96.97 |

Table 5: Accuracy values for the 10 most frequent frames.

### 3.4.4. Frequency Experiments

Since PAC performed much worse than LSC, I executed further experiments that focused on tuple frequencies on the *SUBJ:NP* data. First, I changed the composition of the test set. I excluded 50% of all tuples with a frequency of 2 from the corpus and added them to the test set. This way unseen tuples were guaranteed to occur in the test data. In addition, I randomly selected 10% of the remaining tuple tokens and added them to the test set as well.

I then defined another baseline method given in Figure 3, similar to the one defined above, which only relies on the frequencies of the original verb $v$, the replacement verb $v'$, the subject $s$, and the object $o$.

```
if $f(\langle v, s, o\rangle) > 0$ and $f(\langle v', s, o\rangle) > 0$ then
    if $f(\langle v, s, o\rangle) > f(\langle v', s, o\rangle)$ then
        return $v$
    else
        return $v'$
    end if
else if $f(\langle v, o\rangle) > 0$ and $f(\langle v', o\rangle) > 0$ then
    if $f(\langle v, o\rangle) > f(\langle v', o\rangle)$ then
        return $v$
    else
        return $v'$
    end if
else if $f(v') > f(v)$ then
    return $v'$
else
    return $v$
end if
```

Figure 3: Baseline algorithm based on verbs, subjects and objects.

Figure 4 shows accuracy values in relation to the coverage of the test tuples through the training data. VSO denotes tuples that were seen during training, VO consists of tuples of which the verb and the object but not the subject were seen, and V contains tuples of which only the verb appeared during training. For VSO and VO, the baseline method is better than the clustering models. This shows that the clustering models are only useful in sparse data situations. PAC seems to be fairly consistent for all kinds of tuples whereas

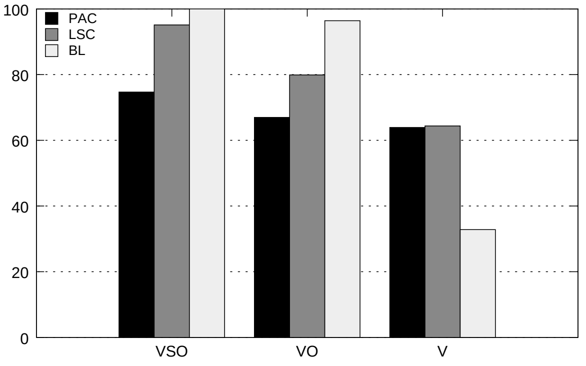

Figure 4: Accuracy by occurrence of verbs, subjects, and objects.

the accuracy of LSC drops when it encounters unseen data. However, PAC is also less accurate than LSC in all situations.

Figure 5 shows the accuracy values broken down by frequency. Both LSC and PAC perform better with tuples which occurred more often in the training corpus. Thus, the inferior performance of PAC is not related to tuple frequency but rather to the problems stated in the previous sections.

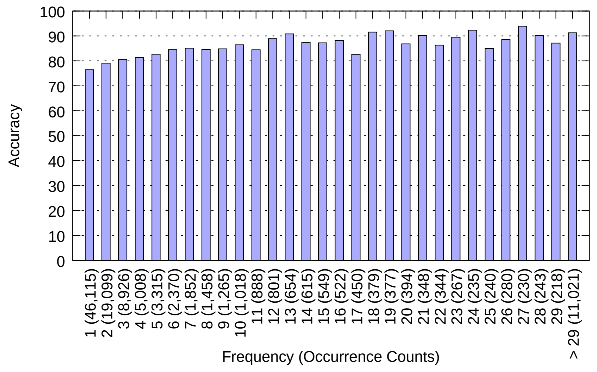

Figure 5: Accuracy by frequency in PAC (number of tuples with those frequencies in brackets).

### 3.4.5. Variations of the Model Parameters

In this section, the effects of model parameter variations are examined. Figure 6 shows that the number of EM iterations causes the accuracy of PAC to increase until it begins to converge at over 74% after about 50 iterations. LSC appears to reach a stable level of over 93% after 35 iterations. The need for more iterations of an PAC model could be explained by the fact that the MDL models have to be trained as well.

Another parameter that can be varied is the number of clusters. I tried increasing the number of clusters from 50 to 100, however, I was not able to observe a significant change in accuracy (74.32% to 74.5%). This seems to match the experiences made in the evaluation of LSC (cf. Rooth et al. (1999)).

Lastly, we change the MDL weight $w$ in Equation 5 which determines the influence of model length and description length. When $w$ is set to a smaller value, the model length becomes less influential, thus enabling the resulting models to become larger. For the standard setting ($w = 1$), the accuracy is 70.47%. When using $w = 0.5$, the selectional

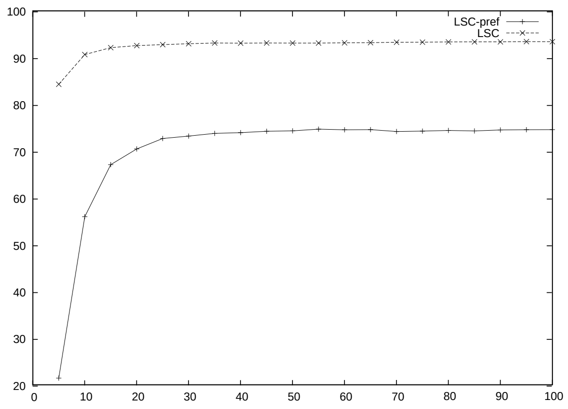

Figure 6: Accuracy by iteration in PAC.

preference models grow larger and the accuracy increases to 73.97%, and when using $w = 2$, the selectional preference models are more shallow and the accuracy decreases to 68.83%. These observations confirm the hypothesis that the lower accuracy of the PAC model is related to the degree of abstraction in the selectional preference model.

### 3.4.6. Discussion

All accuracy values show that PAC is inferior to LSC in a pseudo-disambiguation task. Multiple issues that are responsible were pointed out.

The first problem is caused by inconsistent clustering. Frequent tuples are represented well, however there seems to be an issue with the classification of middle-frequent data. This issue can also be illustrated by looking at accuracy values for this data. Figure 5 shows unexpected, significant jumps in those regions.

The second problem lies with the MDL model. It seems that the models seem to overgeneralize to the point where they have difficulties in distinguishing real data from fake data. The fact that the description length weight $w$ has a considerable influence on accuracy gives further evidence for the degree of abstraction being too high. For example, some of the examined clusters' selectional preference models mainly had high-level nodes like *thing* and *location* as terminal nodes. Clark and Weir (2002) report similar results from their experiments.

As Dagan et al. (1999) point out, performance at pseudo-disambiguation is not necessarily meaningful to a model's performance in a non-artificial task. Clark and Weir (2002) hint at this issue as well. Experiments by Wagner et al. (2009) on word sense disambiguation show that due to sparse-data problems PAC produces better results than LSC. In addition, the assumption that choosing pseudo-words from a frequency-sorted list is not necessarily true as can be demonstrated using the example tuples given in Table 4. *Cut* and *increase* are in fact semantically related which makes the pseudo-disambiguation task more difficult.

## 4. Related Work

### 4.1. Similarity-Based Models of Word Co-Occurrence Probabilities

Dagan et al. (1999) give an overview of probabilistic methods which can be applied to model word co-occurrence and describe a new approach that includes information on word similarity. Given a combination of words, these methods yield a likelihood value for this combination. Determining the likelihood of combinations of verbs, frames and arguments can be regarded as an instance of this task.

The authors compare similarity-based models against traditional back-off and maximum likelihood estimation (MLE) methods. Similarity-based language models as defined by the authors have three components: A scheme for deciding which word pairs require an estimate, a method for combining information, and a similarity measure. The authors propose mathematical methods for each of these components. The authors suggest Katz' back-off as a decision scheme, weighted linear combinations as their combination method, and compare the merits of Kullback-Leibler divergence, Jensen-Shannon divergence, the $L_1$ norm, and confusion probability as similarity measures.

For models containing these components, pseudo-disambiguation tasks were carried out. The data used for their experiments was obtained by tagging a corpus with parts of speech and by then searching for patterns to extract noun-verb pairs. The resulting verb-argument tuples were split into training and test data 80% to 20%. The authors trained an MLE and a Katz back-off model with data containing singletons. In addition, each method was trained with data from which singletons were eliminated, a step which the authors hoped would improve model quality.

The similarity-based methods consistently outperform the baseline models with the Jensen-Shannon divergence yielding the lowest error rates. Also, models which include singular events produce even better results than those which did not.

### 4.2. Similarity Class Model

Clark and Weir (2002) present a statistical model for the probability of a noun appearing as an argument of a given predicate called Similarity Class model. It estimates the probability of a word being generalized as a certain set of WordNet synsets using Bayes' theorem and chi-square tests. To determine the optimal degree of generalization, the authors apply a chi-square test to compare the significance of choosing between two sets of synsets.

The authors evaluate their model using pseudo-disambiguation. Among others, they compare their model with the MDL pruning method by (L. and Abe, 1998).

The similarity class model correctly identified approximately 72% of the test cases, whereas the MDL model had an accuracy of only approximately 63%. Measurements of the average depth of generalization showed that MDL tends to chose a higher level of abstraction than the similarity class method. The authors suggest that MDL overgeneralizes when applied in a pseudo-disambiguation task. They also state that they find the task "somewhat artificial".

### 4.3. A Simple Similarity-Based Model for selectional preferences

Erk (2007) proposes another model for selectional preferences. Her approach makes use of two not necessarily different corpora which serve different purposes.

From the *primary corpus*, verb-argument tuples are extracted. These tuples consist of a predicate, an argument position and a headword. The *generalization corpus* is used to calculate similarities of words. The author proposes Cosine similarity, Dice coefficient, Jaccard coefficient, and mutial-information based metrics by Lin and Hindle, respectively as possible similarity measures.

Erk compares her model to LSC in a pseudo-disambiguation experiment. This instance differs from the ones we previously examined in that semantic roles are used instead of grammatical relations. Thus, pseudo-words are created for nouns instead of verbs. The author finds similarity models to produce an error rate of about 16%, an LSC model with 30 clusters performs worse at about 31%.

## 5. Conclusion

In this paper, I provided a qualitative and quantitative evaluation of the PAC model. I used pseudo-disambiguation, which is a well-established evaluation method in computational linguistics. I pointed out three specific problems:

First, some verbs were frequently misclassified. This problem does not seem to be affected by changing the number clusters.

Second, the abstraction over nouns in the selectional preference models seems to harm the overall performance in a pseudo-disambiguation task. The degree of abstraction can be directly influenced by changing the model's MDL parameter, which provides a degree of freedom regarding the trade-off between fitting the training data and abstraction.

Third, pseudo-disambiguation evaluates only whether a model can distinguish between semantically plausible and implausible input. This relies heavily on the choice of the test set and its relation to the training corpus. It is thus not necessarily an indicator for how suitable the model is for specific applications. I provided evidence based on experiments in this paper and results from others which supports the notion that while the method is capable of asserting the quality of the probabilities supplied by the model, it might be less useful for predicting its performance in real-world tasks.

## 6. Acknowledgements

Many thanks to Helmut Schmid and Sabine Schulte im Walde for supervising my Studienarbeit and to Jason Utt for his useful comments.

## 7. References

S. Abney and M. Light. 1999. Hiding a semantic hierarchy in a Markov model. In *Proceedings of the ACL Workshop on Unsupervised Learning in Natural Language Processing*, pages 1–8.

L.E. Baum. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3(1):1–8.

S. Clark and D. Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

I. Dagan, L. Lee, and F. Pereira. 1999. Similarity-based models of cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

K. Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 216–223, Prague, Czech Republic, June.

W. Gale, K.W. Church, and D. Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *AAAI Fall Symposium Series: Probabilistic Approaches to Natural Language (Working Notes)*, pages 54–60.

Hang L. and N. Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, 24(2):217–244.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech Language*, 4(1):35 – 56.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–312.

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 104–111, Morristown, NJ, USA.

T.G. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus Volume 1-from yesterdays news to tomorrows language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 827–833.

Christian Scheible. 2009. Evaluating a Verb Clustering Model Using Pseudo-Disambiguation. Studienarbeit.

H. Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 162, Morristown, NJ, USA. Association for Computational Linguistics.

S. Schulte im Walde, C. Hying, C. Scheible, and H. Schmid. 2008. Combining EM Training and the MDL Principle for an Automatic Verb Classification incorporating Selectional Preferences. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 496–504, Columbus, OH.

H. Schütze. 1992. Context space. In *AAAI Fall Symposium Series: Probabilistic Approaches to Natural Language (Working Notes)*, pages 113–120. Cambridge, MA, USA.

W. Wagner, H. Schmid, and S. Schulte im Walde. 2009. Verb Sense Disambiguation using a Predicate-Argument-Clustering Model. In *Proceedings of the CogSci Workshop on Distributional Semantics beyond Concrete Concepts*, pages 23–28, Amsterdam, The Netherlands.