

# Automatic Semantic Relation Extraction with Multiple Boundary Generation

**Brandon Beamer and Alla Rozovskaya and Roxana Girju**

University of Illinois at Urbana-Champaign  
{bbeam, rozovska, girju}@uiuc.edu

## Abstract

This paper addresses the task of automatic classification of semantic relations between nouns. We present an improved WordNet-based learning model which relies on the semantic information of the constituent nouns. The representation of each noun's meaning captures conceptual features which play a key role in the identification of the semantic relation. We report substantial improvements over previous WordNet-based methods on the 2007 SemEval data. Moreover, our experiments show that WordNet's IS-A hierarchy is better suited for some semantic relations compared with others. We also compute various learning curves and show that our model does not need a large number of training examples.

## Introduction

The identification of semantic relations is at the core of Natural Language Processing (NLP) and many applications such as automatic text understanding. Furthermore, semantic relations represent the core elements in the organization of lexical semantic knowledge bases intended for inference purposes. In the past few years at many workshops, tutorials, and competitions this research topic has received considerable interest from the NLP community.

Semantic relation identification is the problem of recognizing, for example, the CAUSE-EFFECT (cycling, happiness) relation in the sentence *He derives great joy and happiness from cycling*. This task requires several local and global decisions needed for relation identification. This involves the meaning of the two noun entities along with the meaning of other words in context.

In this paper we present an efficient WordNet-based learning model that identifies and extracts noun features from the WordNet IS-A backbone, which was designed to capture and relate noun senses. The basic idea is that noun - noun pairs which have the same or similar sense collocation tend to encode the same semantic relation. We perform various experiments on the 2007 SemEval Task 4 dataset compare the results against other state-of-the-art WordNet-based algorithm (Moldovan and Badulescu 2005) and the top-ranked systems in SemEval (Girju et al. 2007).

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The results show that our WordNet-based semantic relation model places 5th (cf. F-measure) with respect to the 15 participating systems in the SemEval competition, which is significant taken into consideration that our model uses only the WordNet noun IS-A hierarchy. Moreover, we also compute the learning curves for each relation and show that this model does not need a lot of training data to learn the classification function.

The paper is organized as follows. In the next section we present previous work, followed by the description of the SemEval task and datasets. In Section 4 we present the model and in Section 5 show the result of various experiments. We conclude with a discussion section.

## Previous Work

Most of the attempts in the area of noun - noun semantic interpretation have studied the problem in different limited syntactic contexts, such as noun-noun compounds and other noun phrases (e.g., “N preposition N”, “N, such as N”), and “N verb N”. Recent works in this area follow roughly two main approaches: interpretation based on semantic similarity with previous seen examples (Nastase et al. 2006), and semantic disambiguation relative to an underlying predicate or semantically-unambiguous paraphrase (Lapata 2002), (Kim and Baldwin 2006).

Most methods employ rich ontologies and disregard the sentence context in which the nouns occur, partly due to the lack of annotated contextual data on which they are trained and tested, and partly due to the claim that ontological distinctions are more important than the information derived from the context in which the nouns occur. In this paper, our experimental results also support this claim. However, we show that some semantic relations are better suited for WordNet-based models than others, and that contextual data are important in the performance of a noun - noun semantic parser.

Moreover, most approaches use supervised learning employing various feature sets on fairly small datasets. For example, (Nastase et al. 2006) train their system on 600 noun-modifier pairs classified into six high-level semantic relations (Cause, Participant, Spatial, Temporal, Quality). The problem is that these relations are not uniformly distributed in the dataset. Moreover, most of the 2007 SemEval partic-

ipant systems<sup>1</sup> were trained on an average of 140 examples per relation. This raises questions about the effectiveness of the algorithms and their capability to generalize over seen examples in order to efficiently classify unseen instances. Additionally, it is difficult to talk about the learning curve of each semantic relation, and thus, impossible to draw valid conclusions about the difficulty of the task across different relations.

In this paper we train our interpretation model on various large annotated datasets and present observations on the learning curves generated for each relation.

## Classification of Semantic Relations between Nominals

The SemEval 2007 task on Semantic Relations between Nominals is to identify the underlying semantic relation between two nouns in the context of a sentence. The SemEval effort focuses on seven separate semantic relations: *Cause-Effect*, *Instrument-Agency*, *Product-Producer*, *Origin-Entity*, *Theme-Tool*, *Part-Whole*, and *Content-Container*. The dataset provided consists of 140 training and about 70 test sentences for each of the seven relations considered.

In each training and test example sentence, the nouns are identified and manually labeled with their corresponding WordNet 3.0 senses. Moreover, each example is accompanied by the heuristic pattern (query) the annotators used to extract the sentence from the web and the position of the arguments in the relation. Positive and negative examples of the Cause-Effect relation are listed in (1) and (2) below. Cause-effect relations are semantically similar to other relations such as Temporal, Source, Origin-Entity, and Product-Producer. Instances encoding these relations are called near-miss examples, as shown in (2).

(1) “He derives great joy and  $\langle e_1 \rangle$ happiness $\langle /e_1 \rangle$  from  $\langle e_2 \rangle$ cycling $\langle /e_2 \rangle$ .” WordNet( $e_1$ ) = “happiness%1:12:00:.”<sup>2</sup>, WordNet( $e_2$ ) = “cycling%1:04:00:.”, Cause-Effect( $e_2, e_1$ ) = “true”, Query = “happiness from \*”

(2) “Women may experience  $\langle e_1 \rangle$ anxiety $\langle /e_1 \rangle$  from the  $\langle e_2 \rangle$ time $\langle /e_2 \rangle$  they first learn about the breast abnormality.” WordNet( $e_1$ ) = “anxiety%1:12:00:.”, WordNet( $e_2$ ) = “time%1:11:00:.”, Cause-Effect( $e_2, e_1$ ) = “false”; Query = “anxiety from \*”

The task is defined as a binary classification problem. Thus, given a pair of nouns and their sentential context, a semantic interpretation system decides whether the nouns are linked by the target semantic relation. Based on the information employed, systems can be classified in four categories: (A) systems that use neither the given WordNet synsets nor the queries, (B) systems that use only WordNet senses, (C) systems that use only the queries, and (D) systems that use both.

In this paper we present a knowledge-intensive semantic interpretation system of type-B that relies on WordNet semantic features employed in a supervised learning model.

<sup>1</sup>Few systems, including UIUC used external data sets as well.

<sup>2</sup>The number refers to the noun’s WordNet synset sense key.

## The Model

### Model formulation

We consider a set  $V = \mathcal{E} \cup \mathcal{R}$ . The set  $\mathcal{E} = \{E_1, E_2\}$  ranges over a set of entity types  $L_{\mathcal{E}}$ . The value (called label) assigned to  $E_i \in \mathcal{E}$  is denoted  $s_{E_i} \in L_{\mathcal{E}}$ . The second set  $\mathcal{R}$  is defined as binary relations over  $\mathcal{E}$ . Specifically, for each pair of entities  $\langle E_1, E_2 \rangle$  we use  $\mathcal{R}$  to denote the binary relation  $(E_1, E_2)$  or  $(E_2, E_1)$ . The set of relation labels is  $L_{\mathcal{R}}$  and the label assigned is denoted  $r \in L_{\mathcal{R}}$ .

For example, in the sentence *He derives great joy and happiness from cycling*,  $\mathcal{E} = \{\text{“happiness”}, \text{“cycling”}\}$  and the values of  $s_{E_1}$  and  $s_{E_2}$  are *happiness#2* and *cycling#1* respectively. The set of values  $L_{\mathcal{E}}$  corresponds to the set of all WordNet noun synsets. The set of relation labels  $L_{\mathcal{R}}$  is the set of seven SemEval relations.

The task is defined as a classification problem with the prediction function  $F : (L_{\mathcal{E}} \times L_{\mathcal{E}}) \rightarrow L_{\mathcal{R}}$ .

Let  $T$  be the training set of examples or instances  $T = ((s_{E_1} s_{E_2})^{l_1} r^1 \dots (s_{E_1} s_{E_2})^{l_l} r^l) \subseteq ((L_{\mathcal{E}} \times L_{\mathcal{E}}) \times L_{\mathcal{R}})^l$  where  $l$  is the number of examples  $(s_{E_1} s_{E_2})$  each accompanied by its corresponding semantic relation label  $r \in L_{\mathcal{R}}$ . The problem is to decide which semantic relation to assign to a new, unseen example  $(s_{E_1} s_{E_2})^{l+1}$ . In order to classify a given set of examples (members of  $L_{\mathcal{E}} \times L_{\mathcal{E}}$ ), one needs some kind of measure of the similarity (or the difference) between any two given members.

### Input representation

Like other approaches (Moldovan and Badulescu 2005), (Nastase et al. 2006), our algorithm uses the WordNet IS-A noun hierarchy and semantic boundaries constructed from this hierarchy to classify new unseen noun - noun pairs.

The features used in our learning model are the WordNet semantic classes of the noun constituents. The semantic class of a noun specifies its Word-Net sense (synset) in the context of the sentence and implicitly points to all its hypernyms. Thus, the training examples are represented as a 3-tuple called a *data point*:  $\langle s_{E_1}, s_{E_2}, r \rangle$ , where  $s_{E_i}$  are WordNet noun synsets (in the hypernymy chain of each noun entity synset) and  $r \in L_{\mathcal{R}}$  is the category into which they should be classified.

Here,  $s_{E_i}$  represents the WordNet sense (synset) of the corresponding noun constituent, and  $r$  is the relation encoded by the two concept nouns. For example, the noun - noun pair *mouth - girl* encoding a PART-WHOLE relation has the representation  $\langle \text{mouth\#2}, \text{girl\#2}, \text{Part-Whole} \rangle$ .

### Learning Model

The learning model presented here is a significant improvement over the SemScat model introduced in (Moldovan and Badulescu 2005). We summarize here the algorithm and point out our contribution.

The main idea of the SemScat model is to find the best set of noun semantic classes that would separate the positive and the negative examples, and that would accurately classify unseen instances. This is done by finding a *boundary* (a division in the WordNet noun hierarchy) that would best generalize over the training examples.

A semantic boundary is a set of synsets  $G_k$  which ranges over  $L_{\mathcal{E}}$  such that for every synset in the WordNet noun hierarchy,  $s_{E_i} \in L_{\mathcal{E}}$ , one of the following is true: (a)  $s_{E_i} \in G_k$ ; (b)  $H(s_{E_i}) \cap G_k \neq \emptyset$ ; (c)  $\exists s_{E_j} \in G_k$  such that  $s_{E_i} \in H(s_{E_j})$ , where  $H(x)$  is the set of hypernyms of  $x \in L_{\mathcal{E}}$ .

In other words, a semantic boundary is a division in the noun hierarchy such that any synset in the hierarchy can clearly be defined as on, above, or below the division<sup>3</sup>.

In the next subsection we present an improved boundary detection algorithm. In particular, we introduce a more efficient probability function for better boundary specialization (Steps 3 and 4).

### Boundary Detection Algorithm

A list of boundaries  $G = \{G_1, G_2, \dots, G_n\}$  is generated using the training set through an iterative process called semantic scattering. We start with the most general boundary containing only the top-most WordNet noun synset (*entity#1*) and then specialize it based on the training data until no more useful specializations can be made.

#### Step 1. Create Initial Boundary

The algorithm begins by creating an initial boundary  $G_1 = \{\text{entity}\#1\}$ ; *entity#1* is the most general synset in the WordNet noun hierarchy. The training data examples are then *mapped* to the boundary.

Mapping a data point to a boundary consists of executing a breadth first search of the WordNet noun hierarchy from each  $s_{E_i}$  in the data point, traversing along hypernym relation edges until the nearest boundary member synset is found. Hence the mapping function is  $M : L_{\mathcal{E}} \times G \rightarrow L_{\mathcal{E}}$  where  $M(s_{E_i}, G_k) = s_{E_j} \in G_k$  is the first boundary member synset touched by the breadth-first search. Hence mapping a data point  $\langle s_{E_1}, s_{E_2}, r \rangle$  to  $G_k$  yields  $\langle M(s_{E_1}, G_k), M(s_{E_2}, G_k), r \rangle$ . For example, in Step 1, the initial datapoint  $\langle \text{sandwich}\#1, \text{bag}\#1, \text{Content-Container} \rangle$  maps to the boundary and becomes  $\langle \text{entity}\#1, \text{entity}\#1, \text{Content-Container} \rangle$ . If a noun in a 3-tuple has no hypernym which is a boundary element, this means it is already above the boundary. In this case the 3-tuple in question is ignored and does not become a part of the mapped data set.

#### Step 2. Calculate statistics

Once the training data set is mapped to the current boundary, statistics on it are gathered. These statistics generate a function  $C(s_{E_1}, s_{E_2}, r) = N$ , where  $N$  is the number of data points  $\langle s_{E_1}, s_{E_2}, r \rangle$  in the mapped training data. From this count function, a probability function  $P$  is generated where

$$P(r|s_{E_1}, s_{E_2}) = \frac{C(s_{E_1}, s_{E_2}, r)}{\sum_{r_i \in L_{\mathcal{R}}} C(s_{E_1}, s_{E_2}, r_i)}$$

At this point the current boundary is pushed onto the boundary list  $G$  and the gathered statistics for this boundary are stored for later reference. Because we always push

<sup>3</sup>In general, the noun hierarchy is depicted with the more general synsets on top, becoming more specific as one traverses down. Hence synsets are often referred to as being *above* or *below* other synsets. If synset A is above synset B in the hierarchy, it generally means that A is a hypernym of B.

new boundaries onto the end of the list, the list will always be sorted by specificity.

#### Step 3. Identify the Most Ambiguous Pair

After statistics on the mapped data set are calculated and the corresponding probability function for the current boundary is generated, the algorithm identifies the two most ambiguous synsets in the boundary. This is done via a weighted measure of entropy. In general, the entropy of two synsets  $\langle s_{E_1}, s_{E_2} \rangle$  is

$$E(s_{E_1}, s_{E_2}) = \sum_r -\log_2 (P(r|s_{E_1}, s_{E_2})) P(r|s_{E_1}, s_{E_2}).$$

And the weighted entropy is

$$W(s_{E_1}, s_{E_2}) = E(s_{E_1}, s_{E_2}) \frac{N}{\text{MAX}(N_{max} - N_{min}, 1)},$$

where  $N$  is the total number of times  $\langle s_{E_1}, s_{E_2} \rangle$  occurs in the mapped data set,  $N_{max}$  is the number of times  $\langle s_{E_1}, s_{E_2} \rangle$  occurs with its most popular relation and  $N_{min}$  is the number of times  $\langle s_{E_1}, s_{E_2} \rangle$  occurs with its least popular relation.

Weighting the entropy in this manner gives higher measurements to ambiguous noun pairs which occur a lot in the dataset and lower measurements to ambiguous noun pairs which occur infrequently in the data set. We want to locate the most prevalent ambiguous pairs first because the more prevalent ambiguous pairs present the highest opportunity to subcategorize and in the event that we want our algorithm to stop training early (e.g., to save space and time) we want the best boundary to have synsets which are as specific as possible.

If during this step it is discovered that the highest weighted entropy among all the boundary synsets is 0 – meaning either the boundary is below all our data points or none of the boundary elements are ambiguous anymore – the algorithm halts and the training phase is complete.

#### Step 4. Specialize the boundary

The two noun synsets with the highest weighted entropy are the next candidates for boundary specialization. Boundary specialization is nothing more than replacing these noun synsets in the boundary with their hyponyms.

The first boundary  $G_1 = \{\text{entity}\#1\}$  thus specializes to  $G_2 = \{\text{physical\_entity}\#1, \text{abstract\_entity}\#1, \text{thing}\#8\}$  because  $\{\text{physical\_entity}\#1, \text{abstract\_entity}\#1, \text{thing}\#8\}$  is the hyponym set of *entity#1*. Then on the next iteration if say *abstract\_entity#1* and *physical\_entity#1* were the most ambiguous weighted pair—which is likely—they would be replaced by their hyponyms and so on.

Once the boundary has been specialized, the original data set is mapped to the new boundary and the process iterates (from step 2) until the highest entropy among any two boundary elements is 0, or until the boundary specializes to be lower than all the data points, in which case there are no statistics from which to calculate entropy. At the end of the training phase the algorithm will have generated the list of boundaries  $G$ , sorted by specificity, with their corresponding probability functions.

#### Step 5. Category Prediction

When the system is asked to predict the category of two

synsets  $\langle s_{E_1}, s_{E_2} \rangle$ , it maps them to the best (most specific/lowest) boundary which has relevant statistics. Otherwise said, it maps them to the best boundary such that, according to the corresponding count function  $C$ ,  $\sum_{r \in L_{\mathcal{R}}} C(s_{E_1}, s_{E_2}, r) > 0$ . The category predicted  $r_i$  is  $\arg \max_i P(r_i | s_{E_1}, s_{E_2})$ . For example, if the system were asked to categorize  $\langle \text{sandwich\#1}, \text{bag\#1} \rangle$ , it might map the noun pair to  $\langle \text{substance\#7}, \text{container\#1} \rangle$ —assuming the relevant boundary had these members—and the system might predict, based off of this boundary’s probability function  $P(r | \text{substance\#7}, \text{container\#1})$ , that  $\langle \text{sandwich\#1}, \text{bag\#1} \rangle$  should be categorized as *Content-Container*.

## Improvements and Intuitions

**1) One Boundary vs. Many Boundaries.** The most important difference between our algorithm and Semantic Scattering is while Semantic Scattering strives to discover one optimal boundary, our algorithm keeps track of numerous boundaries, each more specialized than the previous. This has a few ramifications.

First, when predicting the category of two nouns which are both above the optimal boundary in Semantic Scattering, it is unclear what to do. The possibilities are to not make a decision, to make a default decision of either true or false, or to map them to *entity\#1*. Each of these solutions has problems. Mapping them to *entity\#1* completely ignores the semantic classes of the nouns and results in classifying all noun pairs above the boundary to the same relation, whichever is most popular among the nouns higher up in the WordNet hierarchy. Attempting to map them *down* onto the boundary goes against the philosophy of the approach in the first place. While one can be certain that replacing a noun with its hypernym preserves truth, the reverse is not the case. Not making a decision or making a default true/false decision seems to be the most reasonable solution; this however leads to a performance decrease.

Second, nouns which are much lower in the WordNet noun hierarchy than Semantic Scattering’s optimal boundary lose much of their semantic categorical information when they are mapped to the boundary. The larger this gap is between the boundary and the nouns to be classified, the less useful the categorical generalizations become. Indeed, Semantic Scattering has to deal with a delicate balance between overspecializing its boundary—risking becoming too specialized to classify some unseen nouns—and underspecializing its boundary—risking becoming too general to be useful. The best end result one can hope for when using Semantic Scattering is a boundary optimal in the average case.

Our algorithm instead keeps track of as many boundaries as the training data can provide. These boundaries span from the most general to the most specific possible given the training data. When new nouns are categorized, the boundary which categorizes them is chosen based on the input nouns’ locations in the hierarchy. By catering the boundary which is used to the nouns that need to be categorized, our algorithm minimizes the loss of information that occurs when nouns are mapped to a boundary. Also, since our algorithm initializes itself with the boundary  $\{\text{entity\#1}\}$ , there will always be a boundary which is above the the nouns in the hierarchy.

Lastly, with our algorithm, if a noun to be categorized is very far below the best boundary, the reason is because the training data set didn’t have enough data to create a more specific boundary, not because the algorithm chose to settle on a higher boundary. Hence the blame shifts from the algorithm to the training data, where it belongs.

**2) Stability.** Another key difference between our algorithm and Semantic Scattering is that Semantic Scattering randomly selects a small portion of the training data to be a development set. With Semantic Scattering, decisions on whether to further specialize the boundary are made by testing the boundary on the development set every iteration. When the performance begins to decrease, the training stops and the current boundary is considered to be optimal.

The problem with this approach is that since the development set is randomly selected every time the system is trained, the same training data can (and often does) yield numerous “optimal” boundaries. The larger the training data set is the less of a problem this turns out to be, but in reality training data is often sparse and given the same training data the boundary selected by Semantic Scattering to be optimal tends to unstable and thus, unreliable.

Our algorithm does not choose only one boundary, and thus overfitting the training data is not an issue. Because of this, our algorithm has no need to test the performance of each boundary along the way and thus does away with the development set. Given the same training data, our algorithm will produce the same set of trained boundaries every time. Our system is therefore more stable and more reliable than Semantic Scattering.

## Experiments

We performed three experiments. The first two experiments were performed only on the SemEval dataset and focus on the behavior of our model. Specifically, we show how our model compares against the Semantic Scattering and the top-ranked systems in the 2007 SemEval competition. Furthermore, we distinguish between two types of data instances, depending on the type of information required for the identification of a relation within an instance. We compare the performance on both types of data and discuss the suitability of the our model in this context. Finally, experiment III shows the learning curves for each relation. In what follows, we refer to Semantic Scattering as *SemScat1* and to our algorithm as *SemScat2*.

### Experiment I

Experiment I evaluates SemScat2 models with respect to the SemEval test data. Table 1 presents accuracy results of SemScat1 and SemScat2, where a model is trained for each relation on the training data from SemEval and tested on the corresponding SemEval test set. SemScat2 outperforms SemScat1 on all relations, except Theme-Tool, with an absolute increase of 6% on average.

Table 2 shows that SemScat2 places 5th with respect to the top-ranked systems in the SemEval competition. This is despite the fact that it does not make use of sentence context, making a prediction using the noun–noun pair only.

Relation	SemScat1 [% Acc.]	SemScat2 [% Acc.]
Cause-Effect	60.8	73.0
Instrument-Agency	50.7	70.0
Product-Producer	65.5	67.9
Origin-Entity	59.7	63.6
Theme-Tool	63.6	54.5
Part-Whole	63.4	70.4
Content-Container	60.8	67.6
<b>Average</b>	<b>60.7</b>	<b>66.8</b>

Table 1: Experiment I results: Models are trained on SemEval training data and tested on SemEval test data. Acc. means “Accuracy”.

System	F	Acc
UIUC	72.4	76.3
FBK-IRST	71.8	72.9
ILK	71.5	73.2
UCD-S1	66.8	71.4
<b>SemScat2</b>	<b>65.8</b>	<b>66.8</b>

Table 2: Experiment I results: Comparison of SemScat2 with top-ranked B-systems of the SemEval competition. F and Acc mean “F1” and “Accuracy” respectively.

## Experiment II

As in Experiment I, only the SemEval data are used. However, for each relation, the test and training sets are lumped together and 10-fold cross-validation is performed, yielding a prediction for each example.

Relation	SemScat1 [% Acc.]	SemScat2 [% Acc.]
Cause-Effect	57	69
Instrument-Agency	61	70
Product-Producer	64	63
Origin-Entity	61	68
Theme-Tool	62	63
Part-Whole	72	71
Content-Container	58	75
<b>Average</b>	<b>62</b>	<b>68</b>

Table 3: Experiment II results: 10-fold cross-validation on SemEval data. Columns 2 and 3 show accuracy for each relation of SemScat1 and SemScat2 respectively. Acc. means “Accuracy”.

Table 3 compares the performance of SemScat1<sup>4</sup> and SemScat2 with respect to the accuracy of each algorithm on each relation. The results show that SemScat2 significantly outperforms SemScat1 on Cause-Effect, Instrument-Agency, Origin-Entity, Content-Container. On the remaining three relations, the two algorithms exhibit comparable results. Overall, SemScat2 outperforms SemScat1 by 6%.

Moreover, for each relation, we split the lumped training and test examples into *regular* and *context-sensitive*. *Regular* examples are those where the relation between the two

<sup>4</sup>It should be noted that since the choice of development set for SemScat1 is crucial for the performance, the performance of SemScat1 can change dramatically on the same test set with the same training set due to a choice of the development set.

given nouns can be determined out of context. *Context-sensitive* examples are those in which sentence context is required for their correct interpretation. This split was performed manually by one of the authors. Consider, for example, sentences (3) and (4) with respect to the “Cause-Effect” relation. In (3) we can say with high confidence that “Cause-Effect” relation is “True”. By contrast, in (4) it is the sentence context that determines the answer. We consider example (3) regular and (4) context-sensitive.

(3) “The period of  $\langle e_1 \rangle$ tumor shrinkage $\langle /e_1 \rangle$  after  $\langle e_2 \rangle$ radiation therapy $\langle /e_2 \rangle$  is often long and varied”.

(4) “The following are very basic tips which may help you manage your  $\langle e_1 \rangle$ anxiety $\langle /e_1 \rangle$  in the  $\langle e_2 \rangle$ exam $\langle /e_2 \rangle$ .” Comment: Time; the context does not imply that the exam is the cause for the anxiety.

Each relation contains between 26 and 60 context-sensitive examples. Table 4 compares the performance in accuracy of SemScat2 in 10-fold cross-validation on regular (column 2) and context-sensitive (column 3) examples. In parentheses, we list the performance on positive and negative examples within each group<sup>5</sup>.

Relation	Examples (pos.; neg.)	
	Regular [% Acc.]	Context-sensitive [% Acc.]
Cause-Effect	71 (79; 63)	63 (71; 50)
Instrument-Agency	78 (82; 74)	37 (42; 32)
Product-Producer	61 (80; 34)	71 (78; 33)
Origin-Entity	70 (71; 69)	58 (56; 67)
Theme-Tool	66 (54; 72)	48 (42; 58)
Part-Whole	75 (82; 70)	42 (54; 31)
Content-Container	77 (85; 70)	63 (57; 69)
<b>Average</b>	<b>71 (75; 65)</b>	<b>55 (57; 49)</b>

Table 4: Experiment II results on regular and context-sensitive SemEval examples. Columns 2 and 3 show accuracy for each relation of SemScat2 on regular and context-sensitive examples, respectively.

We observe that consistently across all relations, accuracy on both positive and negative examples is better in the regular group than in the corresponding context-sensitive group. Overall, performance on regular examples is considerably higher for all relations (with the exception of Product-Producer) with an average accuracy of 71% for regular examples and 55% for context-sensitive examples (cf. last row of Table4). For Product-Producer, the proportion of negative examples within each group is much lower than the one for the other relations, which explains the performance.

Separating regular examples allows us also to see which relations are best captured with the WordNet hierarchy. In particular, the results on regular examples in Table 4 demonstrate that the best-processed relations are Instrument-Agency, Part-Whole, and Content-Container, while the poorest is Product-Producer.

<sup>5</sup>Positive and negative examples are those labeled as “True” and as “False”, respectively in the Gold Standard.

### Experiment III

In this experiment, we determine the learning curve of each relation. Because the SemEval data contain only 140 examples per relation, which is not sufficient to obtain an accurate learning curve, we use additional datasets<sup>6</sup>. Since the number of examples varies considerably from one relation to another, we group the relations into classes: Class I (approx. 130 examples per relation) contains {Content-Container, Instrument-Agency, Theme-Tool} and class II (approx. 1,000 examples) has {Origin-Entity, Cause-Effect, Product-Producer, Part-Whole}.

Figures 1 and 2 show the learning curves for each class. Each figure displays the results of both SemScat1 and SemScat2. The models are tested on SemEval test data and trained on all other data available.

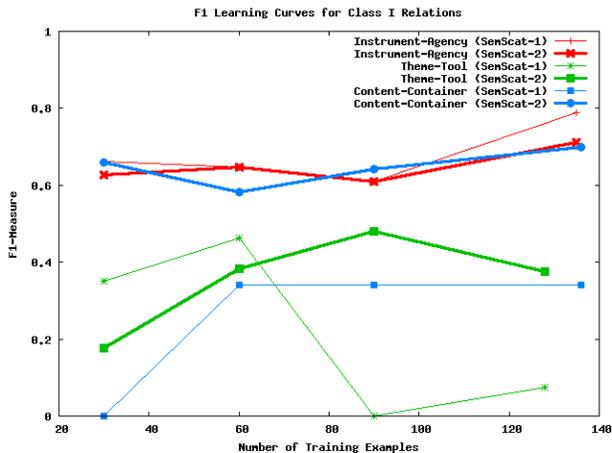


Figure 1: The learning curves for Class I relations.

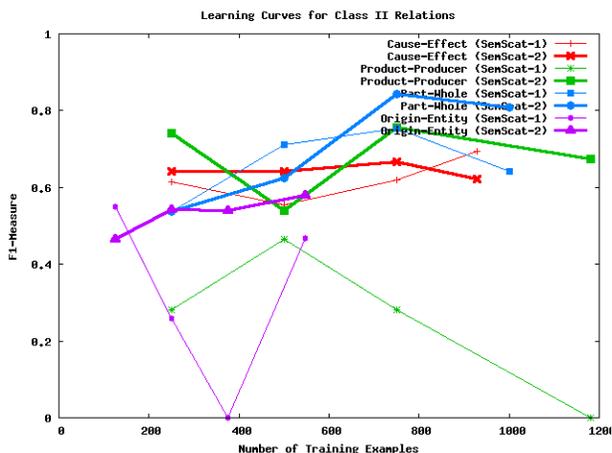


Figure 2: The learning curves for Class II relations.

There are several observations to be made. First, at each level of training, our system either outperforms Semantic

<sup>6</sup>The datasets are described in (Beamer et al. 2007).

Scattering by a significant margin, or performs similarly to it. Second, we note the stability of our system when compared to Semantic Scattering. Our system's performance tends to increase or fluctuate around a relatively stable value when trained on larger data sets, while Semantic Scattering's performance almost seems unrelated to the data set size. This unpredictable behavior is a result of the randomly selected development set which Semantic Scattering uses during training. Third, the learning curves of our system look quite flat. The low saturation point shows that our system needs a relatively small amount of training data to achieve maximum performance.

### Discussion and Conclusions

The contributions of the paper are as follows. First, we have presented a learning model that identifies and extracts noun features efficiently from WordNet's IS-A backbone. Second, our experiments provide more insights into the problem of noun – noun semantic parsing. More specifically, we have shown that our model is superior to the model introduced in (Moldovan and Badulescu 2005) both in terms of performance accuracy and system stability. Moreover, our system places 5th with respect to the top-ranked systems in the SemEval competition. We believe this is an important result, given that the model is based only on WordNet. We have also shown that WordNet structure is capable of capturing some relations better than others. Additionally, we have made a distinction between regular examples and those that require sentence context for the relation identification. The system performs much better on regular examples, as expected. Finally, learning curves show that the task difficulty varies across relations and that the learned representation is highly accurate so the performance results suggest the upper bound on what this representation can do.

### References

Beamer, B.; Bhat, S.; Chee, B.; Fister, A.; Rozovskaya, A.; and Girju, R. 2007. UIUC: A Knowledge-rich Approach to Identifying Semantic Relations between Nominals. In *The 4th ACL Workshop on Semantic Evaluations*.

Girju, R.; Nakov, P.; Nastase, V.; Szpakowicz, S.; Turney, P.; and Yuret, D. 2007. Semeval-2007 Task 04: Classification of Semantic Relations between Nominals. In *The 4th ACL International Workshop on Semantic Evaluations*.

Kim, S. N., and Baldwin, T. 2006. Interpreting Semantic Relations in Noun Compounds via Verb Semantics. In *International Computational Linguistics Conference (COLING)*.

Lapata, M. 2002. The Disambiguation of Nominalizations. *Computational Linguistics* 28(3):357–388.

Moldovan, D., and Badulescu, A. 2005. A Semantic Scattering Model for the Automatic Interpretation of Genitives. In *The Human Language Technology Conference (HLT)*.

Nastase, V.; Shirabad, J. S.; Sokolova, M.; and Szpakowicz, S. 2006. Learning Noun-Modifier Semantic Relations with Corpus-based and WordNet-based Features. In *The 21st National Conference on Artificial Intelligence (AAAI)*.