AI Magazine Volume 18 Number 4 (1997) (© AAAI)

Corpus-Based Approaches to Semantic Interpretation in Natural Language Processing

Hwee Tou Ng and John Zelle

■ In recent years, there has been a flurry of research into empirical, corpus-based learning approaches to natural language processing (NLP). Most empirical NLP work to date has focused on relatively low-level language processing such as part-of-speech tagging, text segmentation, and syntactic parsing. The success of these approaches has stimulated research in using empirical learning techniques in other facets of NLP, including semantic analysis—uncovering the meaning of an utterance. This article is an introduction to some of the emerging research in the application of corpus-based learning techniques to problems in semantic interpretation. In particular, we focus on two important problems in semantic interpretation, namely, word-sense disambiguation and semantic parsing.

Getting computer systems to understand natural language input is a tremendously difficult problem and remains a largely unsolved goal of AI. In recent years, there has been a flurry of research into empirical, corpus-based learning approaches to natural language processing (NLP). Whereas traditional NLP has focused on developing hand-coded rules and algorithms to process natural language input, corpus-based approaches use automated learning techniques over corpora of natural language examples in an attempt to automatically induce suitable language-processing models. Traditional work in natural language systems breaks the process of understanding into broad areas of syntactic processing, semantic interpretation, and discourse pragmatics. Most empirical NLP work to date has focused on using statistical or other learning techniques to automate relatively low-level language processing such as part-of-speech tagging, segmenting text, and syntactic parsing. The success of these approaches, following on the heels of the success of similar techniques in speech-recognition research, has stimulated research in using empirical learning techniques in other facets of NLP, including *semantic analysis*—uncovering the meaning of an utterance.

In the area of semantic interpretation, there have been a number of interesting uses of corpus-based techniques. Some researchers have used empirical techniques to address a difficult subtask of semantic interpretation, that of developing accurate rules to select the proper meaning, or *sense*, of a semantically ambiguous word. These rules can then be incorporated as part of a larger system performing semantic analysis. Other research has considered whether, at least for limited domains, virtually the entire process of semantic interpretation might yield to an empirical approach, producing a sort of semantic parser that generates appropriate machine-oriented meaning representations from natural language input. This article is an introduction to some of the emerging research in the application of corpus-based, learning techniques to problems in semantic interpretation.

Word-Sense Disambiguation

The task of word-sense disambiguation (WSD) is to identify the correct meaning, or *sense*, of a word in context. The input to a WSD program consists of real-world natural language sentences. Typically, a separate phase prior to WSD to identify the correct part of speech of

Sense Number	Sense Definition
1	Readiness to give attention
2	Quality of causing attention to be given
3	Activity, subject, and so on, that one gives time and attention to
4	Advantage, advancement, or favor
5	A share (in a company, business, and so on)
6	Money paid for the use of money

Table 1. Sense Definitions of the Noun Interest.

the words in the sentence is assumed (that is, whether a word is a noun, verb, and so on). In the output, each word occurrence *w* is tagged with its correct sense, in the form of a sense number *i*, where *i* corresponds to the *i*-th sense definition of *w* in its assigned part of speech. The sense definitions are those specified in some dictionary. For example, consider the following sentence: In the *interest* of stimulating the economy, the government lowered the *interest* rate.

Suppose a separate part-of-speech tagger has determined that the two occurrences of *interest* in the sentence are nouns. The various sense definitions of the noun *interest*, as given in the *Longman Dictionary of Contemporary English* (LDOCE) (Bruce and Wiebe 1994; Procter 1978), are listed in table 1. In this sentence, the first occurrence of the noun *interest* is in sense 4, but the second occurrence is in sense 6. Another wide-coverage dictionary commonly used in WSD research is WORDNET (Miller 1990), which is a public-domain dictionary containing about 95,000 English word forms, with a rather refined sense distinction for words.

WSD is a long-standing problem in NLP. To achieve any semblance of understanding natural language, it is crucial to figure out what each individual word in a sentence means. Words in natural language are known to be highly ambiguous, which is especially true for the frequently occurring words of a language. For example, in the WORDNET dictionary, the average number of senses for each noun for the most frequent 121 nouns in English is 7.8, but that for the most frequent 70 verbs is 12.0 (Ng and Lee 1996). This set of 191 words is estimated to account for about 20 percent of all word occurrences in any English free text. As such, WSD is a difficult and prevalent problem in NLP.

WSD is also an essential part of many NLP applications. In information retrieval, WSD has brought about improvement in retrieval accuracy. When tested on part of the TREC corpus, a standard information-retrieval test collection, WSD improves precision by about 4.3 percent (from 29.9 percent to 34.2 percent) (Schütze and Pedersen 1995). Similarly, in machine translation, WSD has been used to select the appropriate words to translate into a target language. Specifically, Dagan and Itai (1994) reported successful use of WSD to improve the accuracy of machine translation. The work of Schütze and Pedersen (1995) and Dagan and Itai (1994) clearly demonstrates the utility of WSD in practical NLP applications.

Early work on WSD, such as Kelly and Stone (1975) and Hirst (1987), used hand coding of knowledge to disambiguate word sense. The knowledge-acquisition process can be laborious and time consuming. For each word to be disambiguated, the appropriate inference knowledge must be handcrafted. It is difficult to come up with a comprehensive set of the necessary disambiguation knowledge. Also, as the amount of disambiguation knowledge grows, manual maintenance and further expansion become increasingly complex. Thus, it is difficult to scale up manual knowledge acquisition to achieve wide coverage for real-world sentences.

The recent surge in corpus-based NLP research has resulted in a large body of work on WSD of unconstrained real-world sentences. In contrast to manually hand coding disambiguation knowledge into a system, the corpus-based approach uses machine-learning techniques to automatically acquire such disambiguation knowledge, using sense-tagged corpora and large-scale linguistic resources

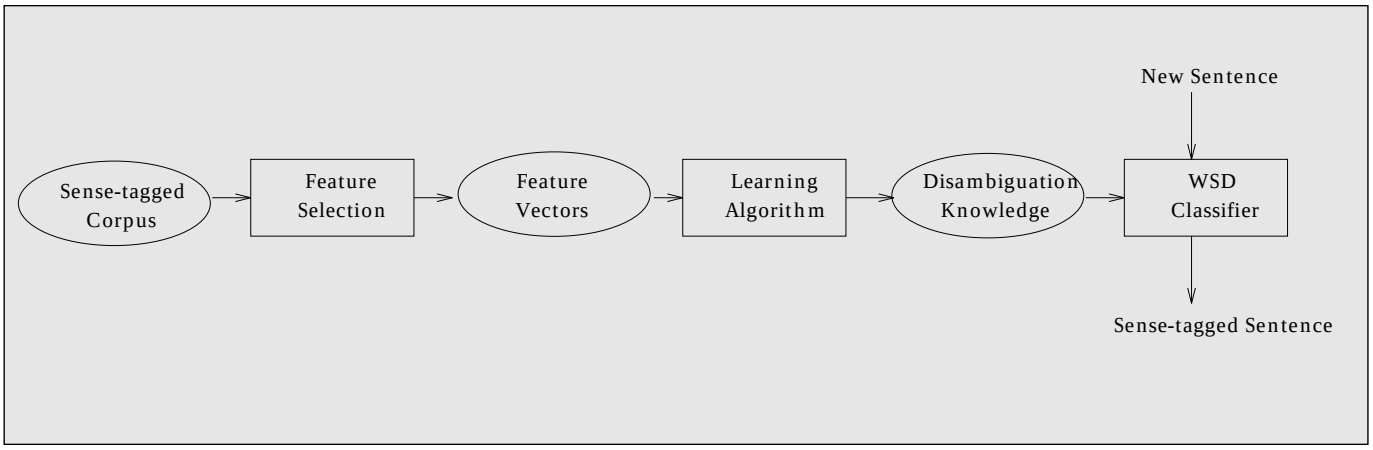

Figure 1. Supervised Word-Sense Disambiguation.

such as online dictionaries. As in other corpus-based learning approaches to NLP, more emphasis is now placed on the empirical evaluation of WSD algorithms on large quantities of real-world sentences.

Corpus-based WSD research can broadly be classified into the supervised approach and the unsupervised approach.

Supervised Word-Sense Disambiguation

In the *supervised approach*, a WSD program learns the necessary disambiguation knowledge from a large sense-tagged corpus, in which word occurrences have been tagged manually with senses from some wide-coverage dictionary, such as the LDOCE or WORDNET. After training on a sense-tagged corpus in which occurrences of word *w* have been tagged, a WSD program is then able to assign an appropriate sense to *w* appearing in a new sentence, based on the knowledge acquired during the learning phase.

The heart of supervised WSD is the use of a supervised learning algorithm. Typically, such a learning algorithm requires its training examples, as well as test examples, to be encoded in the form of feature vectors. Hence, there are two main parts to supervised WSD: (1) transforming the context of a word *w* to be disambiguated into a feature vector and (2) applying a supervised learning algorithm. Figure 1 illustrates the typical processing phases of supervised WSD.

Feature Selection Let *w* be the word to be disambiguated. The words surrounding *w* will form the context of *w*. This context is then mapped into a feature vector $((f_1 v_1)\ldots(f_n v_n))$, which is a list of features f_i and their associated values v_i. An important issue in supervised WSD is the choice of appropriate features. Intuitively, a good feature should capture an important source of knowledge critical in determining the sense of *w*.

Various kinds of feature representing different knowledge sources have been used in supervised WSD research. They include the following:

Surrounding words: *Surrounding words* are the *unordered* set of words surrounding *w*. These surrounding words typically come from a fixed-size window centered at *w* or the sentence containing *w*. Unordered surrounding words, especially in a large context window, tend to capture the broad topic of a text, which is useful for WSD. For example, surrounding words such as *bank*, *loan*, and *payment* tend to indicate the "money paid for the use of money" sense of *interest*.

Local collocations: A *local collocation* refers to a short sequence of words near *w*, taking the word order into account. Such a sequence of words need not be an idiom to qualify as a local collocation. Collocations differ from surrounding words in that word order is taken into consideration. For example, although the words *in*, *the*, and *of* by themselves are not indicative of any particular sense of *interest*, when these words occur in the sequence "in the interest of," it always implies the "advantage, advancement, or favor" sense of *interest*. The work of Kelly and Stone (1975), Yarowsky (1993), Yarowsky (1994), and Ng and Lee (1996) made use of local collocations to disambiguate word sense.

Syntactic relations: Traditionally, selectional restrictions as indicated by syntactic relations such as subject-verb, verb-object, and adjective-noun are considered an important source of WSD knowledge. For example, in the

sentence "he sold his interest in the joint venture," the verb-object syntactic relation between the verb *sold* and the head of the object noun phrase *interest* is indicative of the "share in a company" sense of *interest*. In disambiguating the noun *interest*, the possible values of the verb-object feature are the verbs (such as *sold*) that stand in a verb-object syntactic relation with *interest*.

Parts of speech and morphological forms: The part of speech of the neighboring words of *w* and the morphological form of *w* also provide useful knowledge to disambiguate *w* (Ng and Lee 1996; Bruce and Wiebe 1994). For example, in Ng and Lee (1996), there is a part-of-speech feature for each of the ±3 words centered around *w*. There is also one feature for the morphological form of *w*. For a noun, the value of this feature is either singular or plural; for a verb, the value is one of infinitive (as in the uninflected form of a verb such as *fall*), present-third-person-singular (as in *falls*), past (as in *fell*), present-participle (as in *falling*), or past-participle (as in *fallen*).

Most previous research efforts on corpus-based WSD use one or more of the previously given knowledge sources. In particular, the work of Ng and Lee (1996) attempts to integrate this diverse set of knowledge sources for corpus-based WSD. Some preliminary findings suggest that local collocation provides the most important source of disambiguation knowledge, although the accuracy achieved by the combined knowledge sources exceeds that obtained by using any one of the knowledge sources alone (Ng and Lee 1996). That local collocation is the most predictive seems to agree with past observation that humans need a narrow window of only a few words to perform WSD (Choueka and Lusignan 1985).

One approach taken to select and decide on the explicit interaction among the possible features is from Bruce and Wiebe (1994) and Pedersen, Bruce, and Wiebe (1997), who explore a search space for model selection in the context of building a probabilistic classifier.

Learning Algorithms Having decided on a set of features to encode the context and form the training examples, the next step is to use some supervised learning algorithm to learn from the training examples. A large number of learning algorithms have been used in previous WSD research, including Bayesian probabilistic algorithms (Pedersen and Bruce 1997a; Mooney 1996; Bruce and Wiebe 1994; Leacock, Towell, and Voorhees 1993; Gale, Church, and Yarowsky 1992a; Yarowsky 1992), neural networks (Mooney 1996; Leacock, Towell, and Voorhees 1993), decision lists (Mooney 1996; Yarowsky 1994), and exemplar-based algorithms (Ng 1997a; Mooney 1996; Ng and Lee 1996; Cardie 1993).

In particular, Mooney (1996) evaluated seven widely used machine-learning algorithms on a common data set for disambiguating six senses of the noun *line* (Leacock, Towell, and Voorhees 1993). The seven algorithms that he evaluated are (1) a Naive-Bayes classifier (Duda and Hart 1973), (2) a perceptron (Rosenblatt 1958), (3) a decision tree learner (Quinlan 1993), (4) a *k* nearest-neighbor classifier (exemplar-based learner) (Cover and Hart 1967), (5) logic-based DNF (disjunctive normal form) learners (Mooney 1995), (6) logic-based CNF (conjunctive normal form) learners (Mooney 1995), and (7) a decision-list learner (Rivest 1987). We briefly describe two of the learning algorithms that have been used for WSD: a naive-Bayes algorithm and an exemplar-based algorithm, PEBLS.

Naive-Bayes: The naive-Bayes algorithm (Duda and Hart 1973) is based on Bayes's theorem:

$$P(C_i \mid \wedge v_j) = \frac{P(\wedge v_j \mid C_i)P(C_i)}{P(\wedge v_j)} \; i = 1 \ldots n,$$

where $P(C_i \mid \wedge v_j)$ is the probability that a test example is of class C_i given feature values v_j. ($\wedge v_j$ denotes the conjunction of all feature values in the test example.) The goal of a naive-Bayes classifier is to determine the class C_i with the highest conditional probability $P(C_i \mid \wedge v_j)$. Because the denominator $P(\wedge v_j)$ of the previous expression is constant for all classes C_i, the problem reduces to finding the class C_i with the maximum value for the numerator.

The naive-Bayes classifier assumes independence of example features, so that

$$P(\wedge v_j \mid C_i) = \prod_j P(v_j \mid C_i).$$

During training, naive-Bayes constructs the matrix $P(v_j \mid C_i)$, and $P(C_i)$ is estimated from the distribution of training examples among the classes.

PEBLS: PEBLS is an exemplar-based (or nearest-neighbor) algorithm developed by Cost and Salzberg (1993). It has been used successfully for WSD in Ng (1997a) and Ng and Lee (1996). The heart of exemplar-based learning is a measure of the similarity, or distance, between two examples. If the distance between two examples is small, then the two examples are similar. In PEBLS, the distance between two symbolic values v_1 and v_2 of a feature *f* is defined as

$$d(v_1, v_2) = \sum_{i=1}^{n} | P(C_i \mid v_1) - P(C_i \mid v_2) |,$$

where *n* is the total number of classes. $P(C_i \mid v_1)$ is estimated by $N_{1,i}/N_1$, where $N_{1,i}$ is the num-

ber of training examples with value v_1 for feature f that is classified as class i in the training corpus, and N_1 is the number of training examples with value v_1 for feature f in any class. $P(C_i \mid v_2)$ is estimated similarly. This distance metric of PEBLS is adapted from the value-difference metric of the earlier work of Stanfill and Waltz (1986). The distance between two examples is the sum of the distances between the values of all the features of the two examples.

Let k be the number of nearest neighbors to use for determining the class of a test example, $k \geq 1$. During testing, a test example is compared against all the training examples. PEBLS then determines the k training examples with the shortest distance to the test example. Among these k closest-matching training examples, the class that the majority of these k examples belong to will be assigned as the class of the test example, with tie among multiple majority classes broken randomly.

Mooney (1996) reported that the simple naive-Bayes algorithm gives the highest accuracy on the line corpus tested. The set of features used in his study only consists of the unordered set of surrounding words. Past research in machine learning has also reported that the naive-Bayes algorithm achieved good performance on other machine-learning tasks, in spite of the conditional independence assumption made by the naive-Bayes algorithm, which might be unjustified in some of the domains tested.

Recently, Ng (1997a) compared the naive-Bayes algorithm with the exemplar-based algorithm PEBLS on the DSO National Laboratories corpus (Ng and Lee 1996), using a larger value of k (k = 20) and 10-fold cross validation to automatically determine the best k. His results indicate that with this improvement, the exemplar-based algorithm achieves accuracy comparable to the naive-Bayes algorithm on a test set from the DSO corpus. The set of features used in this study only consists of local collocations.

Hence, both the naive-Bayes algorithm and the exemplar-based algorithm give good performance for WSD. The potential interaction between the choice of features and the learning algorithms appears to be an interesting topic worthy of further investigation.

One drawback of the supervised learning approach to WSD is the need for manual sense tagging to prepare a sense-tagged corpus. The work of Brown et al. (1991) and Gale, Church, and Yarowsky (1992a) tried to make use of aligned parallel corpora to get around this problem. In general, a word in a source language can be translated into several different target language words, depending on its intended sense in context. For example, the tax sense of *duty* is translated as *droit* in French, whereas the obligation sense is translated as *devoir* (Gale, Church, and Yarowsky 1992a). An aligned parallel corpus can thus be used as a natural source of *sense tags*, occurrences of the word *duty* that are translated as *droit* have the tax sense of *duty*, and so on. The shortcoming of this approach is that sense disambiguation is now task specific and language specific. It is tied to the machine-translation task between the two languages concerned.

Another trick that has been used to generate sense-tagged data is to conflate two unrelated English words, such as *author* and *baby* into an artificial compound word *author-baby* (Schütze 1992). All occurrences of the words *author* and *baby* in the texts are replaced with the compound word *author-baby*. The goal is then to disambiguate occurrences of *author-baby* as *author* or *baby*, with the correct answers being the original word forms in the texts. Although this approach has the advantage of an unlimited supply of tagged data without manual annotation, the disambiguation problem is highly artificial, and it is unclear how the disambiguation accuracy obtained should be interpreted.

Unsupervised Word-Sense Disambiguation

In the WSD research literature, *unsupervised WSD* typically refers to disambiguating word sense without the use of a sense-tagged corpus. It does not necessarily refer to clustering of unlabeled training examples, which is what unsupervised learning traditionally means in machine learning.

Most research efforts in unsupervised WSD rely on the use of knowledge contained in a machine-readable dictionary (Lin 1997; Resnik 1997; Agirre and Rigau 1996; Luk 1995; Wilks et al. 1990). A widely used resource is WORDNET. Besides being an online, publicly available dictionary, WORDNET is also a large-scale taxonomic class hierarchy, where each English noun sense corresponds to a taxonomic class in the hierarchy. The is-a relationship in WORDNET's taxonomic class hierarchy is an important source of knowledge exploited in unsupervised WSD algorithms.

We illustrate unsupervised WSD with an algorithm of Resnik (1997) that disambiguates noun senses. His algorithm makes use of the WORDNET class hierarchy and requires that sentences in a training corpus be parsed so that syntactic relations such as subject-verb, verb-object, adjective-noun, head-modifier, and modifier-head can be extracted from a sen-

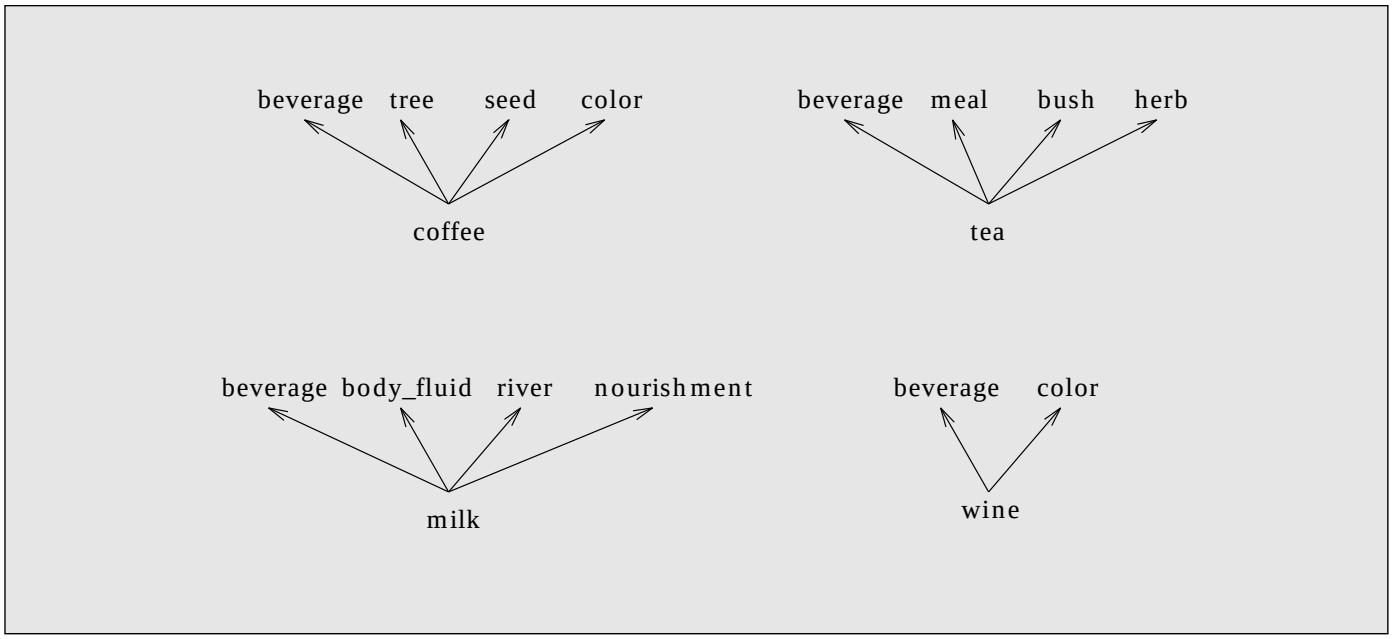

Figure 2. Unsupervised Word-Sense Disambiguation: Resnik's Method.

tence. Each syntactic relation involves two words: (1) the noun *n* to be disambiguated and (2) a verb (in subject-verb, verb-object relations), an adjective (in adjective-noun relation), or another noun (in head-modifier, modifier-head relations).

Consider the following example to explain Resnik's algorithm: Suppose we want to disambiguate the noun *coffee* in a test sentence containing the verb-object syntactic relation *drink coffee*. The noun *coffee* has four senses in WORDNET: coffee as a kind of (1) beverage, (2) tree, (3) seed, or (4) color. The algorithm examines the parsed training corpus, looking for all occurrences of *drink x* in a verb-object relation. Examples of possible past occurrences are *drink tea, drink milk,* and *drink wine*. The goal is to be able to determine that the nouns *tea, milk, wine,* and so on, are most similar to the beverage sense of *coffee* without requiring that *tea, milk, wine,* and so on, be manually tagged with the correct sense in the training corpus.

It turns out that this goal can readily be achieved. The key observation is that although each of the four nouns *coffee, tea, milk,* and *wine* has multiple senses, the sense that is most commonly shared by these four nouns is the beverage sense. Figure 2 lists all the senses of these four nouns in WORDNET. For example, the noun *tea* has four upward-pointing arrows to its four senses, namely, as a kind of (1) beverage, (2) meal, (3) bush, or (4) herb. The beverage sense is shared by all four nouns, whereas the color sense is shared by only two nouns (coffee and wine), with the remaining senses only pointed to by one individual noun.

A frequency-counting scheme will be able to identify this target beverage sense. Specifically, let *n* be a noun with senses $s_1, \ldots, s_k$. Suppose the syntactic relation *R* holds for *n* and the verb *p*. For *i* from 1 to *k*, Resnik's (1997) method computes

$C_i = \{c \mid c \text{ is an ancestor of } s_i\}$

$$a_i = \max_{c \in C_i} \left(\sum_{w \in c} \frac{N_R(p,w)}{N_t(w)} \right),$$

where $N_R(p,w)$ is the number of times the syntactic relation *R* holds for word *w* and *p*, and $N_t(w)$ is the number of taxonomic classes to which *w* belongs. The goal is to select the sense s_i with the maximum value of a_i.

In the previous example, *n* is the noun *coffee*, with s_1 = beverage, s_2 = tree, s_3 = seed, and s_4 = color. Suppose that each of the syntactic relations *drink coffee, drink tea, drink milk,* and *drink wine* occurs exactly once in our raw corpus. For simplicity, let us consider only the concepts listed in figure 2, ignoring other ancestor concepts not shown in the figure. Then $a_1 = 1/4 + 1/4 + 1/4 + 1/2$, $a_2 = a_3 = 1/4$, and $a_4 = 1/4 + 1/2$. Hence, the beverage sense s_1 with the highest value a_1 is chosen. See Resnik (1997) for the details.

The work of Agirre and Rigau (1996) also uses the taxonomic class hierarchy of WORDNET to achieve WSD. However, instead of using syntactic relations, their algorithm uses the surrounding nouns in a window centered at *n*, the noun to be disambiguated. The idea is to locate a class that contains, proportionately speaking,

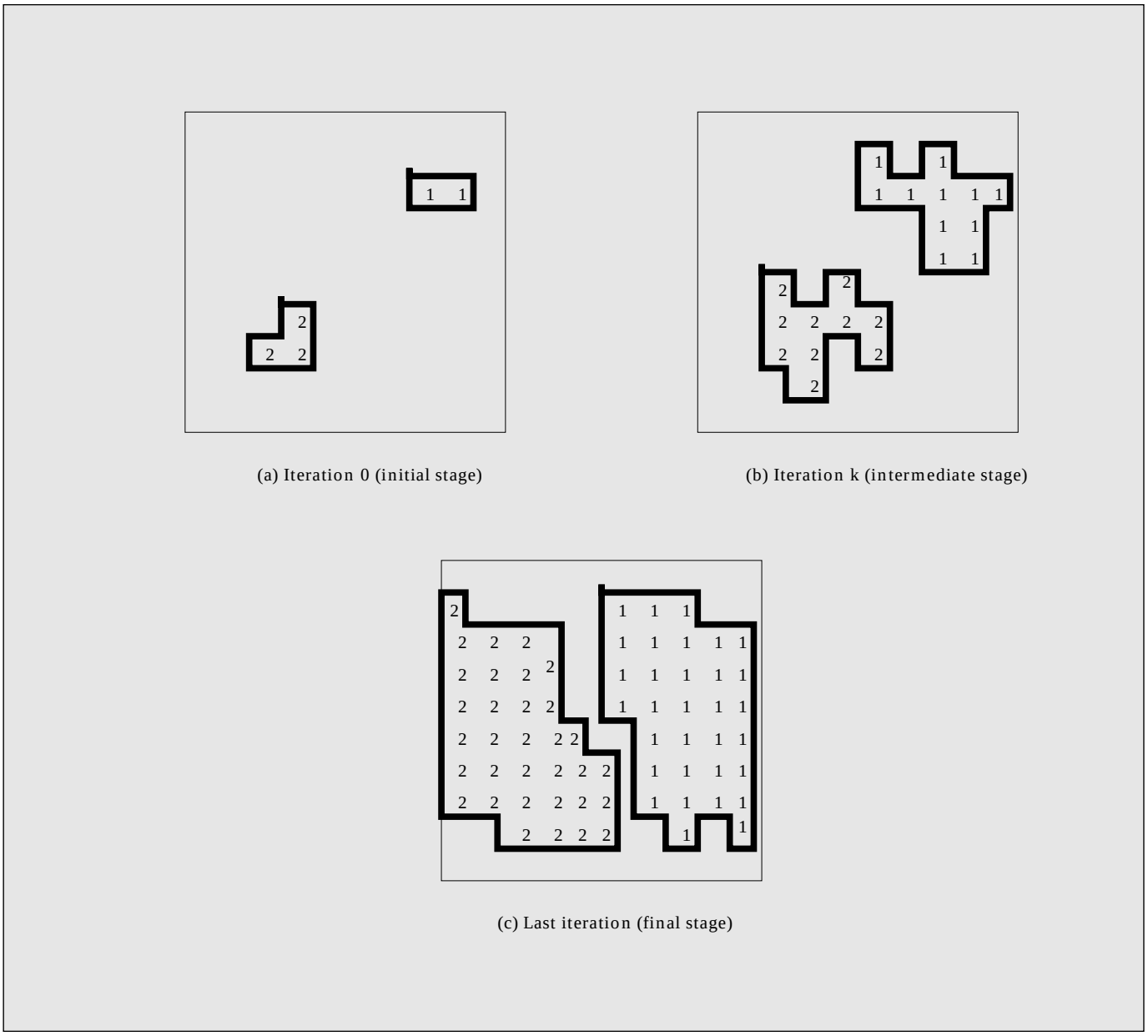

Figure 3. Unsupervised Word-Sense Disambiguation: Yarowsky's Method.

the highest number of senses of the nouns in the context window. The sense of *n* under this class is the chosen sense. The precise metric is formulated in the form of *conceptual density.*

LDOCE is another large-scale linguistic resource exploited in past work on unsupervised WSD, such as the work of Wilks et al. (1990) and Luk (1995), who use the dictionary definitions of LDOCE and the cooccurrence statistics collected to achieve WSD.

The unsupervised WSD method of Yarowsky (1995) iteratively uses the decision list supervised learning algorithm. It requires only a small number of sense-tagged occurrences of a word *w* to start with. These initial sense-tagged occurrences form the seeds, as illustrated in figure 3a. In this figure, an example is either labeled with a 1 or a 2, denoting whether it is classified as sense 1 or sense 2 by the algorithm.

At each iteration of the algorithm, new untagged occurrences of *w* that the algorithm can confidently assign senses to are tagged with senses. These are untagged examples where the probability of classification, as assigned by the decision list algorithm, is above a certain threshold. This process is illustrated in figure 3b, where more examples are labeled with

senses at an intermediate stage. A new classifier is then built from the new, larger set of tagged occurrences, using the decision-list–learning algorithm. More untagged occurrences are then assigned senses in the next iteration, and the algorithm terminates when all untagged occurrences of *w* in a test corpus have been assigned word sense, as shown in figure 3c. Yarowsky reported that his method did as well as supervised learning, although he only tested his method on disambiguating binary, coarse-grained senses.

The work of Schütze (1992) also adopted an unsupervised approach to WSD, relying only on a large untagged raw corpus. For each word *w* in the corpus, a vector of the cooccurrence counts of the surrounding words centered at a window size of 1000 characters around *w* is created. These vectors are clustered, and the resulting clusters represent the various senses of the word. To make the computation tractable, he used singular-value decomposition to reduce the dimensions of the vectors to around 100. A new occurrence of the word *w* is then assigned the sense of the nearest cluster. The accuracy on disambiguating 10 words into binary distinct senses is in excess of 90 percent.

Performance

As pointed out in Resnik and Yarowsky (1997), the evaluation of empirical, corpus-based WSD has not been as rigorously pursued as other areas of corpus-based NLP, such as part-of-speech tagging and syntactic parsing. The lack of standard, large, and widely available test corpora discourages the empirical comparison of various WSD approaches.

Currently, several sense-tagged corpora are available. These include a corpus of 2,094 examples about 6 senses of the noun *line* (Leacock, Towell, and Voorhees 1993); a corpus of 2,369 sentences about 6 senses of the noun *interest* (Bruce and Wiebe 1994); SEMCOR (Miller et al. 1994b), a subset of the BROWN corpus with about 200,000 words in which all content words (nouns, verbs, adjectives, and adverbs) in a running text have manually been tagged with senses from WORDNET; and the DSO corpus (Ng and Lee 1996), consisting of approximately 192,800 word occurrences of the most frequently occurring (and, hence, most ambiguous) 121 nouns and 70 verbs in English. The last three corpora are publicly available from New Mexico State University, Princeton University, and Linguistic Data Consortium (LDC), respectively.

A baseline called the most frequent heuristic has been proposed as a performance measure for WSD algorithms (Gale, Church, and Yarowsky 1992b). This heuristic simply chooses the most frequent sense of a word *w* and assigns it as the sense of *w* in test sentences without considering any effect of context. Any WSD algorithm must perform better than the most frequent heuristic to be of any significant value.

For supervised WSD algorithms, an accuracy of about 73 percent to 76 percent is achieved on the line corpus (Mooney 1996; Leacock, Towell, and Voorhees 1993); 87.4 percent on the interest corpus (Ng and Lee 1996); 69.0 percent on the SEMCOR corpus (Miller et al. 1994b); and 58.7 percent and 75.2 percent on two test sets from the DSO corpus (Ng 1997a). In general, the disambiguation accuracy depends on the particular words tested, the number of senses to a word, and the test corpus from which the words originate. Verbs are typically harder to disambiguate than nouns, and disambiguation accuracy for words chosen from a test corpus composed of a wide variety of genres and domains, such as the Brown corpus, is lower than the accuracy of words chosen from, say, business articles from the *Wall Street Journal* (Ng and Lee 1996).

Given an adequate number of sense-tagged training examples (typically of a few hundred examples), state-of-the-art WSD programs outperform the most frequent heuristic baseline. For example, figure 4 shows the learning curve of WSD accuracy achieved by LEXAS, an exemplar-based, supervised WSD program (Ng 1997b). The accuracy shown is averaged over 43 words of the DSO corpus, and each of these words has at least 1300 training examples in the corpus. The figure indicates that the accuracy of supervised WSD is significantly higher than the most frequent heuristic baseline.

The accuracy for unsupervised WSD algorithms is harder to assess because most are tested on different test sets with varying difficulty. Although supervised WSD algorithms have the drawback of requiring a sense-tagged corpus, they tend to give a higher accuracy compared with unsupervised WSD algorithms. For example, when tested on SEMCOR, Resnik's (1997) unsupervised algorithm achieved accuracies in the range of 35.3 percent to 44.3 percent for ambiguous nouns. Because WORDNET orders the sense definitions of each word from the most frequent to the least frequent, the baseline method of most frequent heuristic can be realized by always picking sense one of WORDNET. As reported in Miller et al. (1994b), such a most frequent heuristic baseline achieved accuracy of 58.2 percent for all ambiguous words.

Similarly, the conceptual density method of

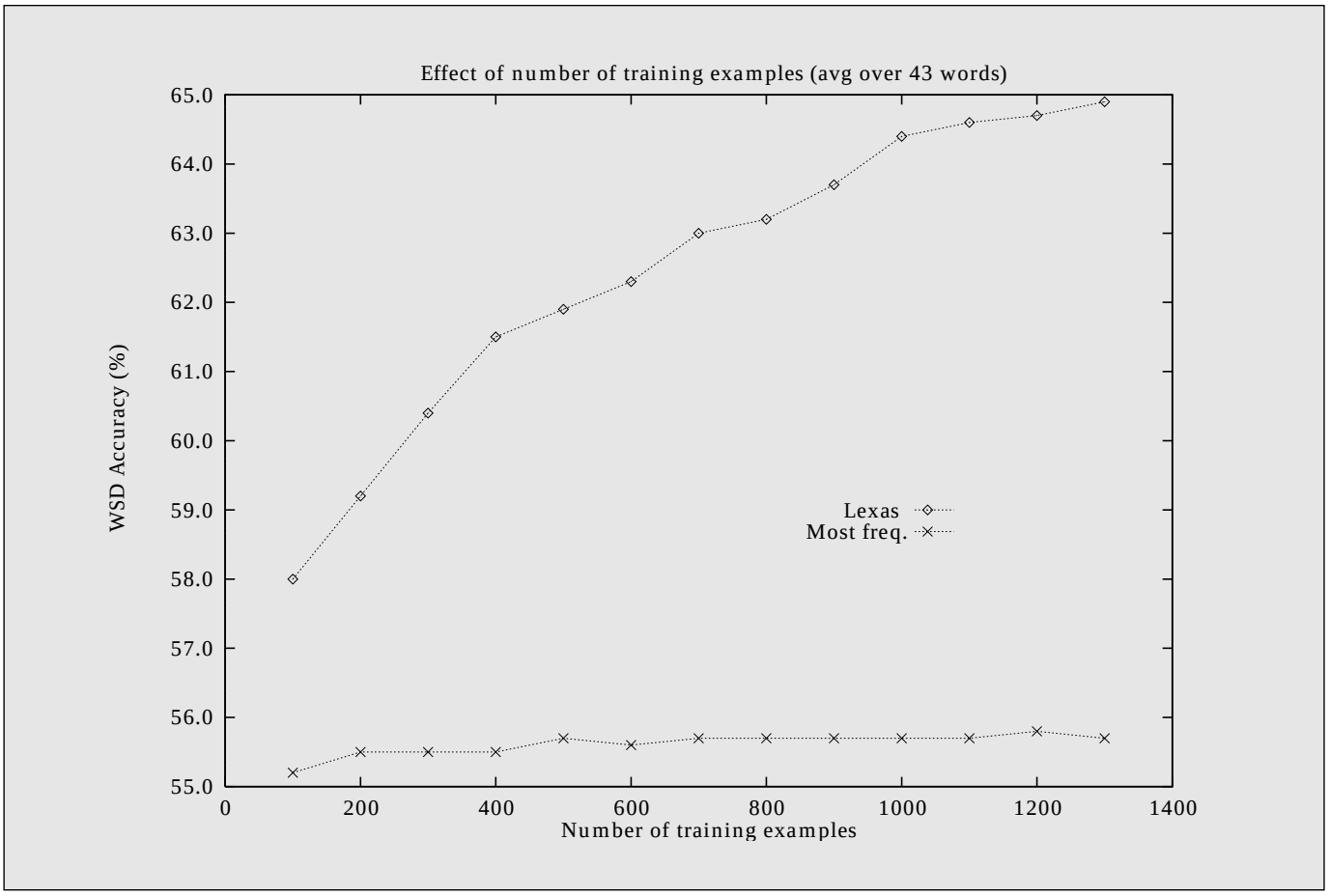

Figure 4. Effect of Number of Training Examples on LEXAS Word-Sense Disambiguation Accuracy Averaged over 43 Words with at Least 1300 Training Examples.

Agirre and Rigau (1996) was also implemented and tested on a subset of the MUC-4 terrorist corpus in Peh and Ng (1997). The results indicate that on this test corpus, the accuracy achieved by the conceptual density method is still below the most frequent heuristic baseline. Recently, Pedersen and Bruce (1997a, 1997b) also used both supervised and unsupervised learning algorithms to disambiguate a common set of 12 words. Their results indicate that although supervised algorithms give average accuracies in excess of 80 percent, the accuracies of unsupervised algorithms are about 66 percent, and the accuracy of the most frequent heuristic is about 73 percent. However, the accuracies of unsupervised algorithms on disambiguating a subset of five nouns are better than the most frequent baseline (62 percent versus 57 percent). These five nouns have two to three senses to a noun. In general, although WSD for coarse-grained, binary word-sense distinction can yield accuracy in excess of 90 percent, as reported in Yarowsky (1995), disambiguating word senses to the refined sense distinction of, say, the WORDNET dictionary is still a challenging task, and much research is still necessary to achieve broad-coverage, high-accuracy WSD (Ng 1997b).

Semantic Parsing

The research discussed to this point involves using empirical techniques to disambiguate word sense, a difficult subpart of the semantic-interpretation process. This section considers research that goes a step farther by using empirical approaches to automate the entire process of semantic interpretation.

Abstractly viewed, semantic interpretation can be considered as a problem of mapping some representation of a natural language input into a structured representation of meaning in a form suitable for computer manipulation. We call this the *semantic parsing problem*. The semantic parsing model of language acquisition has a long history in AI research

Sentence: "Show me the morning flights from Boston to Denver."

Meaning:
```
SELECT flight_id
FROM flights
WHERE from_city = Boston
        AND to_city = Denver
        AND departure_time <= 12:00
```

Figure 5. An Example of Semantic Parsing by a Database-Query Application.

(Langley 1982; Selfridge 1981; Sembugamoorthy 1981; Anderson 1977; Reeker 1976; Siklossy 1972, Klein and Kuppin 1970). This early work focused on discovering learning mechanisms specific to language acquisition and the modeling of cognitive aspects of human language learning. The more recent work discussed here is based on the practical requirements of robust NLP for realistic NLP systems. These approaches use general-purpose machine-learning methods and apply the resulting systems on much more elaborate NLP domains.

Modern approaches to semantic parsing can vary significantly in both the form of input and the form of output. For example, the input might be *raw data*, strings of phonemes spoken to a speech-recognition system or a sentence typed by a user, or it might be significantly preprocessed to turn the linear sequence of sounds or words into a more structured intermediate representation such as a parse tree that captures the syntactic structure of a sentence. Similarly, the type of output that might be useful is strongly influenced by the task at hand because as yet, there is little agreement on what a general meaning-representation language should look like.

Research on empirical methods in semantic parsing has focused primarily on the creation of natural language front ends for database querying. In this application, a sentence submitted to the semantic parser is some representation of a user's question, and the resulting meaning is a formal query in a database query language that is then submitted to a database-management system to retrieve the (hopefully) appropriate answers. Figure 5 illustrates the type of mapping that might be required, transforming a user's request into an appropriate SQL command.

The database query task has long been a touchstone in NLP research. The potential of allowing inexperienced users to retrieve useful information from computer archives was recognized early on as an important application of computer understanding. The traditional approach to constructing such systems has been to use the current linguistic representation to build a set of rules that transforms input sentences into appropriate queries. Thus, query systems have been constructed using *augmented transition networks* (Woods 1970), an "operationalization" of context-free grammars with associated actions to produce semantic representations; *semantic grammars* (Hendrix, Sagalowicz, and Slocum 1978; Brown and Burton 1975), *context-free grammars* having nonterminal symbols that represent the expression of domain-specific concepts; and *logic grammars* (Abramson and Dahl 1989; Warren and Pereira 1982), general phrase-structure grammars that encode linguistic dependencies and structure-building operations using logical unification.

Traditional approaches have suffered a number of shortcomings related to the complexity of the knowledge that must be expressed. It takes considerable linguistic expertise to construct an appropriate grammar. Furthermore, the use of domain-specific knowledge is required to correctly interpret natural language input. To handle this knowledge efficiently, it is often intertwined with the rules for representing knowledge about language itself. As the sophistication of the input and the database increase, it becomes exceedingly difficult to craft an accurate and relatively complete interface. The bottom line is that although it is possible to construct natural language interfaces by hand, the process is time and expertise intensive, and the resulting interfaces are often incomplete, inefficient, and brittle. Database applications are an attractive proving ground for empirical NLP. Although challenging, the understanding problem has built-in constraints that would seem to make it a feasible task for automatic acquisition. The problem is circumscribed both by the limited domain of discourse found in a typical database and the simple communication goal, namely, query processing. It is hoped that techniques of empirical NLP might enable the rapid development of robust domain-specific database interfaces with less overall time and effort.

An empirical approach to constructing natural language interfaces starts with a training corpus comprising sentences paired with appropriate translations into formal queries. Learning algorithms are utilized to analyze the training data and produce a semantic parser that can map subsequent input sentences into

appropriate queries. The learning problem is depicted in figure 6. Approaches to this problem can be differentiated according to the learning techniques used. As in other areas of empirical NLP, some researchers have attempted to extend statistical approaches, which have been successful in domains such as speech recognition, to the semantic parsing problem (Miller et al. 1996; Miller et al. 1994a; Pieraccini, Levin, and Lee 1991). Kuhn and De Mori (1995) describe a method based on semantic classification trees, a variant of traditional decision tree–induction approaches familiar in machine learning. The CHILL system (Zelle and Mooney 1996; Zelle 1995) uses techniques from *inductive logic programming*, a subfield of machine learning, that investigates the learning of relational concept definitions.

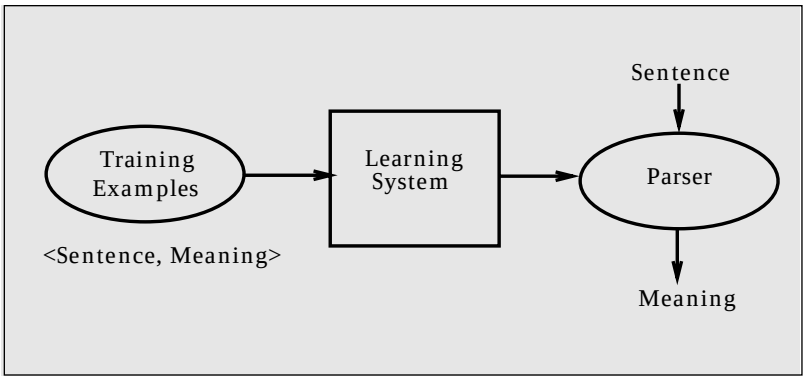

Figure 6. Learning Semantic Mappings.

A Statistical Approach

Research at BBN Systems and Technologies as part of the Advanced Research Projects Agency–sponsored Air Travel Information Service (ATIS) Project (Miller et al. 1994a) represents a paradigm case of the application of statistical modeling techniques to the semantic parsing problem. The target task in the ATIS competition is the development of a spoken natural language interface for air travel information. In the BBN approach, the heart of the semantic parser is a statistical model that represents the associations between strings of words and meaning structures. Parsing of input is accomplished by searching the statistical model in an attempt to find the most likely meaning structure given the input sentence. The learning problem is to set the parameters of the statistical model to accurately reflect the relative probabilities of various meanings, given a particular input.

Representations Meanings are represented as a tree structure, with nodes in the tree representing concepts. The children of a node represent its component concepts. For example, the concept of a flight might include component concepts such as airline, flight number, origin, and destination. Figure 7 shows a tree that might represent the sentence from our previous example. The nodes shown in circles are the nonterminal nodes of the tree and represent concepts, and the rounded rectangles are terminal nodes and represent something akin to lexical categories. This tree essentially represents the type of analysis that one might obtain from a semantic grammar, a popular approach to the construction of database interfaces.

The basic requirements of the representation are that the semantic concepts are hierarchically nested, and the order of nodes in the tree matches the ordering of words in the sentence (that is, the tree can be placed in correspondence with the sentence without any crossing edges). This meaning representation is not a query language as such but partially specified trees form a basic framelike annotation scheme that is automatically translated into a database query. The frame representation of our example is shown in figure 8.

Learning a Statistical Model In recent incarnations (Miller et al. 1996), the construction of the frame representation is a two-step process: First, a statistical parsing process is used to produce a combined syntactic-semantic parse tree. This tree is based on a simple syntactic parsing theory, with nodes labeled both for syntactic role and semantic category. For example, the flight node of figure 7 would be labeled flight/np to show it is a node representing the concept of a flight and plays the role of a noun phrase in the syntax of the sentence. This tree structure is then augmented with frame-building operations. The top node in the tree is associated with the decision of what the overall frame should be, and internal nodes of the tree are associated with slot-filling operations for the frame. Each node gets tagged as filling some particular slot in the frame (for example, origin) or the special *null tag* to indicate it does not directly fill a slot. The input to the learning phase is a set of sentences paired with syntactic-semantic parse trees that are annotated with frame-building actions.

The initial parsing process is similar to other approaches proposed for statistically based syntactic parsing. Because our main focus here is on semantic mapping, we do not go into the details; however, a general overview is in order. The statistical model treats the derivation of the parse as the traversal of a path through a probabilistic recursive transition network.

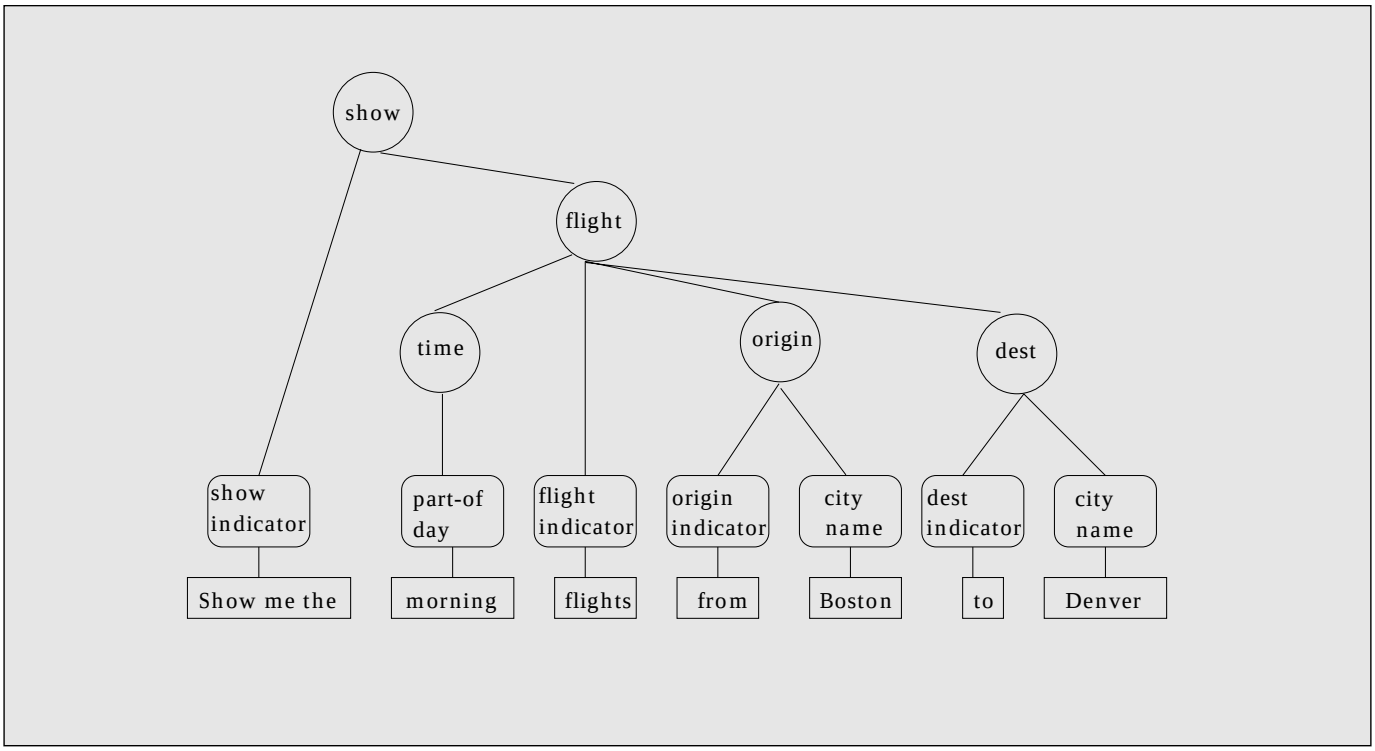

Figure 7. A Tree-Structured Meaning Representation.

FRAME: Air-Transportation
SHOW: (flight-information)
ORIGIN: (City "Boston")
DEST: (City "Denver")
TIME: (part-of-day "MORNING")

Figure 8. Frame Representation of Flight-Information Query.

Probabilities on transitions have the form $P(state_n \mid state_{n-1}, state_{up})$. That is, the probability of a transition to $stat\ e_n$ depends on the previous state and the label of the network we are currently traversing. For example, $P(location/pp \mid arrival/vp\_head, arrival/vp)$ represents the probability that within a verb phrase describing an arrival, a *location*/*pp* node (a prepositional phrase describing a location) occurs immediately following an *arrival/vp_head* node (the main verb in a phrase associated with an arrival). The probability metric assigned to a parse is then just the product of each transition probability along the path corresponding to the parse tree. The transition probabilities for the parsing model are estimated by noting the empirical probabilities of the transitions in the training data.

In the database interface task, the trained parsing model is searched to produce the *n* most probable parse trees corresponding to the sequence of words in an input sentence. These *n* best parses are then submitted to the semantic interpretation model for annotation with frame-building operations. As mentioned, the meanings M_s are decomposed into a decision about the frame type *FT* and the slot fillers *S*. The probability of a particular meaning M_s given a tree is calculated as

$$P(M_s \mid T) = P(FT,S \mid T)$$
$$= P(FT)P(T \mid FT)P(S \mid FT,T) \ .$$

That is, the probability of a particular frame and set of slot fillers is determined by the a priori likelihood of the frame type, the probability of the parse tree given the frame type, and the probability that the slots have the assigned values given the frame type and the parse tree. $P(FT)$ is estimated directly from the training data. $P(T \mid FT)$ is obtained by rescoring the transition probabilities in the parse tree, as described earlier, but with the extra frame-type information. Finally, the probabilistic model for slot filling assumes that fillers can accurately be predicted by considering only the frame type, the slot operations already performed,

and the local parse tree context. This local context includes the node itself, two left siblings, two right siblings, and four immediate ancestors. The final probability of a set of slot fillings is taken as the product of the likelihood of each individual slot-filling choice. The training examples are insufficient for the direct estimation of all the parameters in the slot-filling model; so, the technique relies on heuristic methods for growing statistical decision trees (Magerman 1994).

Using the learned semantic model is a multiphase process. The *n* best syntactic-semantic parse trees from the parsing model are first rescored to determine $P(T \mid TF)$ for each possible frame type. Each of those models is combined with the corresponding prior probability of the frame type to yield $P(FT)P(T \mid FT)$. The *n* best of these theories then serve as candidates for consideration with possible slot fillers. A beam search is utilized to consider possible combinations of fill operations. The final result is an approximation of the *n* most likely semantic interpretations.

The parsing and semantic analysis components described here have been joined with a statistically based discourse module to form a complete, trainable natural language interface. The system has been evaluated on some samples from the ATIS corpus. After training on 4000 annotated examples, the system had a reported error rate of 21.6 percent on a disjoint set of testing examples. Although this performance would not put the system at the top of contenders in the ATIS competition, it represents an impressive initial demonstration of a fully trainable statistical approach to the ATIS task.

Semantic Classification Trees

The CHANEL system (Kuhn and De Mori 1995) takes an approach to semantic parsing based on a variation of traditional decision tree induction called *semantic classification trees* (SCTs). This technique has also been applied to the ATIS database querying task.

Representations The final representations from the CHANEL system are database queries in SQL. Like the statistical approach discussed previously, however, the output of semantic interpretation is not a directly executable query but an intermediate form that is then turned into an SQL query. The meaning representation consists of a set of features to be displayed and a set of constraints on these features. The representation of our sentence is depicted in figure 9. Basically, this is a listing of the subset of all possible attributes that might be displayed and a subset of all possible constraints on attributes. Using this representation reduces the problem of semantic mapping to basic classification. For each possible attribute, the system must determine whether it is in the final query.

```
DISPLAYED ATTRIBUTES = {flight.flight_info}

CONSTRAINTS =  { flight.from_city = Boston,
                 flight.to_city = Denver,
                 flight.dep_time <1200
               }
```

Figure 9. Attributes and Constraints of Flight-Information Query.

In the CHANEL system, the input to the semantic interpretation process is a partially parsed version of the original sentence. This initial processing is performed by a handcrafted local chart parser that scans through the input sentence for words or phrases that carry constraints relevant to the domain of the database. The target phrases are replaced by variable symbols, and the original content of each symbol is saved for later insertion into the meaning representation. In our example sentence, the chart parser might produce an output such as "show me the TIM flights from CIT to CIT." Here, TIM is the symbol for a word or phrase indicating a time constraint, and CIT is the symbol for a reference to a city. This sentence would then be sent to an SCT-based robust matcher to generate the displayed attribute list and assign roles to constraints.

Learning Semantic Classification Trees An SCT is a variant of traditional decision trees that examines a string of symbols and classifies it into a set of discrete categories. In a typical classification tree, each internal node represents a question about the value of some attribute in an example. An SCT is a binary decision tree where a node represents a pattern to match against the sentence. If the pattern matches, we take the left branch; if it doesn't, then the right branch is taken. When a leaf node is reached, the example is classified with the category assigned to the leaf.

The patterns stored in the nodes of an SCT are a restricted form of regular expression consisting of sequences of words and arbitrarily large gaps. For example, the pattern, < + flight + from +> would match any sentence containing the words *flight* and *from*, provided they appeared in this order with at least one word between them. The special symbols < and > indicate the start and end of the sen-

tence, and + represents a gap of at least one word. A word in a pattern can also be replaced with a set of words indicating *alternation*, a choice of words that could be inserted at that point in the pattern.

During training, an SCT is grown using a typical greedy classification tree–growing strategy. The tree starts as a single node containing the most general pattern < + >. Patterns are specialized by replacing a gap with a more specific pattern. If *w* is a word covered by the gap in some example, then the gap can be replaced with *w*, *w*+, +*w*, or +*w*+. The tree-growing algorithm follows the standard practice of testing all possible single specializations to see which produces the best split of examples according to some metric of the purity of the leaves. This specialization is then implemented, and the growing algorithm is recursively applied to each of the leaves until all leaves contain sets of examples that cannot be split any further. This occurs when all the examples in the leaf belong to the same category, or the pattern at the leaf contains no gaps.

The input to CHILL is a set of training instances consisting of sentences paired with the desired queries. The output is a shift-reduce parser in Prolog that maps sentences into queries.

Applying Semantic Classification Trees to Semantic Interpretation Semantic interpretation is done by a robust matcher consisting of forests of SCTs. Determining the list of displayed attributes is straightforward; there is a forest of SCTs, one for each possible attribute. Each SCT classifies a sentence as either yes or no. A yes classification indicates that the attribute associated with this tree should be displayed for the query and, hence, included in the listing of displayed attributes. A no classification results in the attribute being left out of the list.

Creating the list of constraints is a bit more involved. In this case, SCTs are used to classify substrings within the sentence. There is a tree for each symbol (for example, CIT, TIM) that can be inserted by the local parser. This tree is responsible for classifying the role that this symbol fulfills, wherever it occurs in the sentence. To identify the role of each symbol in the sentence, the symbol being classified is specially marked. To classify the first CIT symbol in our example, the sentence "show me the TIM flights from *CIT to CIT" would be submitted to the role-classification tree for CIT. The * marks the instance of CIT that is being classified. This tree would then classify the entire sentence pattern according to the role of *CIT. Leaves of the tree would have labels such as *origin*, *destination*, or *scrap*, the last indicating that it should not produce any constraint. The constraint is then built by binding the identified role to the value for the symbol that was originally extracted by the local parser in the syntactic phase.

It is difficult to assess the effectiveness of the semantic mapping component alone because it is embedded in a larger system, including many hand-coded parts. However, some preliminary experiments with the semantic mapping portion of CHANEL indicated that it was successful in determining the correct set of displayed attributes in 91 percent of sentences held back for testing. Overall performance on the ATIS measures placed the system near the middle of the 10 systems evaluated in December 1993.

Inductive Logic Programming

Both the statistical and SCT approaches are based on representations that are basically propositional. The CHILL system (Zelle and Mooney 1996; Zelle 1995), which has also been applied to the database query problem, is based on techniques from inductive logic programming for learning relational concepts. The representations used by CHILL differ substantially from those used by CHANEL. CHILL has been demonstrated on a database task for U.S. geography, where the database is encoded as a set of facts in Prolog, and the query language is a logical form that is directly executed by a query interpreter to retrieve appropriate answers from the database.

Representations The input to CHILL is a set of training instances consisting of sentences paired with the desired queries. The output is a shift-reduce parser in Prolog that maps sentences into queries. CHILL treats parser induction as a problem of learning rules to control the actions of a shift-reduce parser expressed as a Prolog program. Control rules are expressed as definite-clause (Prolog) concept definitions. These rules are induced using a general concept learning system employing techniques from *inductive logic programming*, a subfield of machine learning that addresses the problem of learning definite-clause logic descriptions (Prolog programs) (Lavrač and Džeroski 1994; Muggleton 1992) from examples.

As mentioned previously, CHILL produces parsers that turn sentences directly into a logical form suitable for direct interpretation. The meaning of our example sentence might be represented as

answer(*F*, *time*(*F*,*T*), *flight*(*F*), *morning*(*T*), *origin*(*F*, *O*), *equal*(*O*, *city*(*boston*)), *destination*(*F*, *D*), *equal*(*D*, *city*(*denver*))) .

The central insight in CHILL is that the general operators required for a shift-reduce parser to produce a given set of sentence analyses are directly inferable from the representations themselves. For example, building the previous representation requires operators to introduce logical terms such as time(_,_), operators

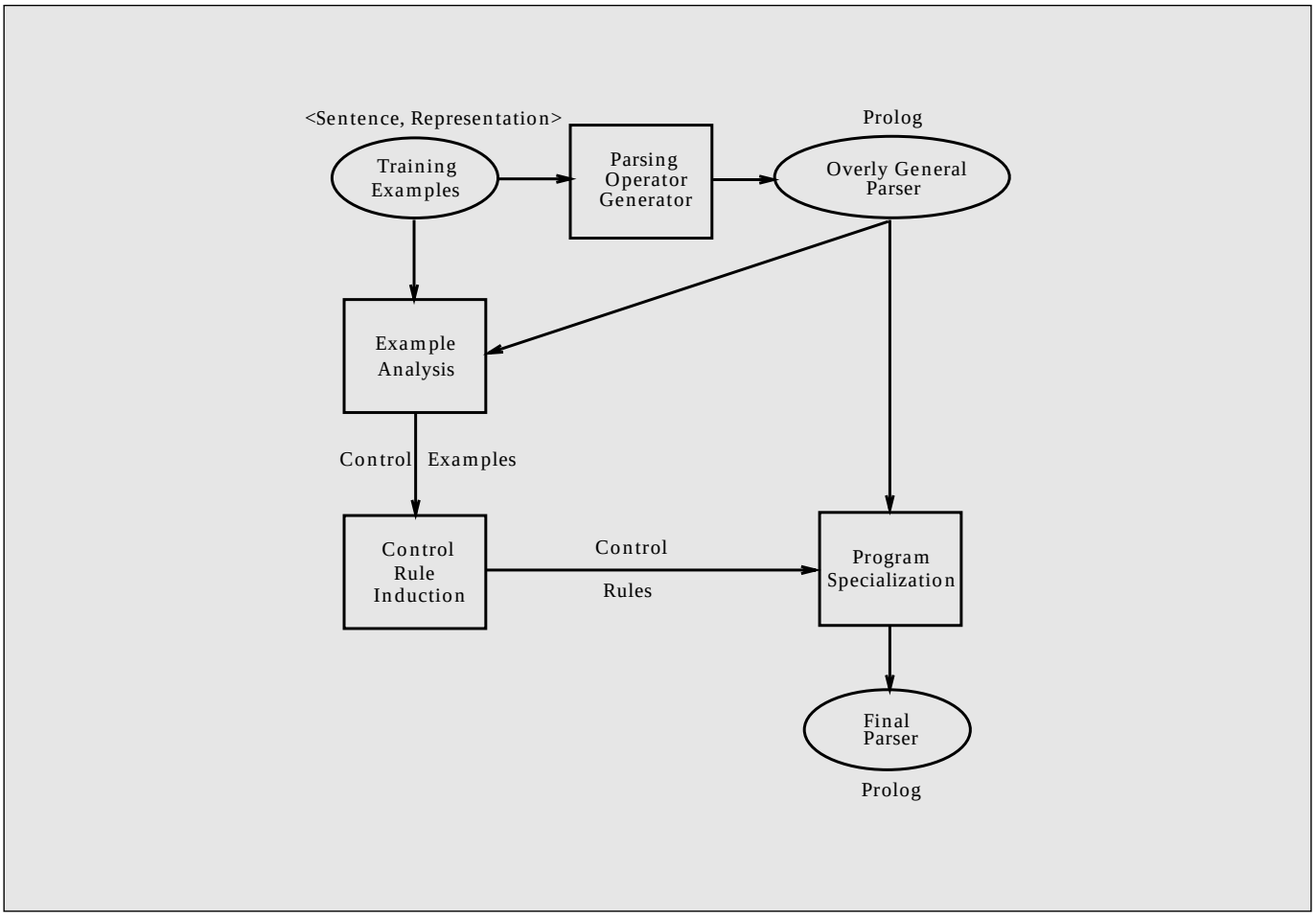

Figure 10. The CHILL Architecture.

to coreference logical variables (for example, an operator that binds first argument of *time* with the argument of *flight)*, and operators to embed terms inside of other terms (for example, to place *time, flight*, and so on, into *answer.* However, just inferring an appropriate set of operators does not produce a correct parser because more knowledge is required to apply operators accurately during the course of parsing an example.

The current context of a parse is contained in the contents of the stack and the remaining input buffer. CHILL uses parses of the training examples to figure out the contexts in which each of the inferred operators is and is not applicable. These contexts are then given to a general induction algorithm that learns rules to classify the contexts in which each operator should be used. Because the contexts are arbitrarily complex parser states involving nested (partial) constituents, CHILL uses a learning algorithm that can deal with structured input and produce relational concept descriptions, which is exactly the problem addressed by inductive logic programming research.

Learning a Shift-Reduce Parser Figure 10 shows the basic components of CHILL. During parser operator generation, the training examples are analyzed to formulate an overly general shift-reduce parser that is capable of producing parses from sentences. The initial parser simply consists of all the parsing operators that can be inferred from the meaning representations in the training examples. In an initial parser, an operator can be applied at any point in a parse, which makes it wildly overly general in that it can produce a great many spurious analyses for any given input sentence. In example analysis, the training examples are parsed using the overly general parser to extract contexts in which the various parsing operators should and should not be employed. The parsing of the training examples is guided by the form of the output (the database query) to determine the correct sequence of operators yielding the desired output. Control-rule induction then uses a general inductive

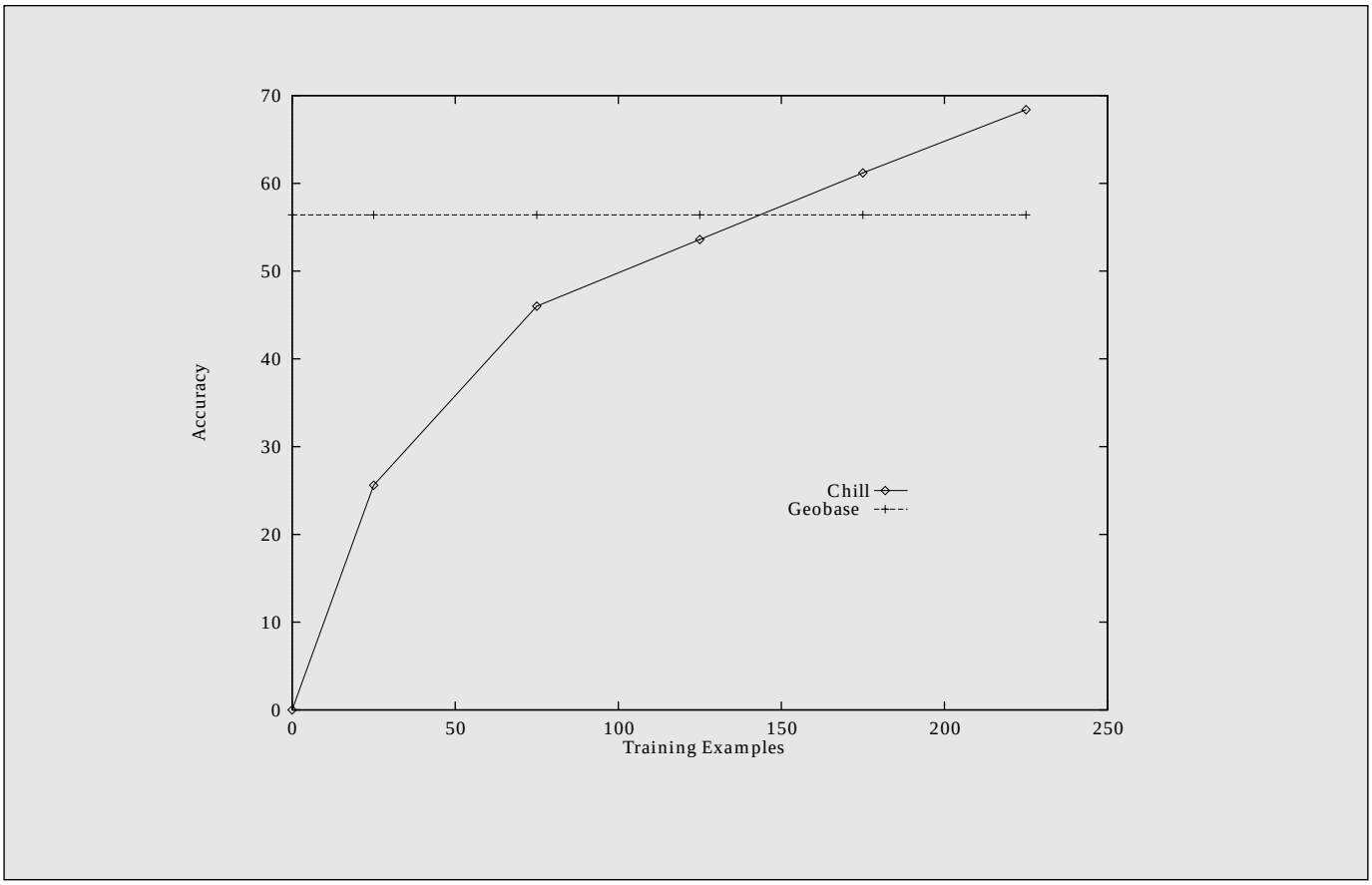

Figure 11. Geoquery: Accuracy.

logic programming algorithm to learn rules that characterize these contexts. The inductive logic programming system analyzes a set of positive and negative examples of a concept and formulates a set of Prolog rules that succeed for the positive examples and fail for the negative examples. For each parsing operator, the induction algorithm is called to learn a definition of parsing contexts in which this operator should be used. Finally, program specialization folds the learned control rules back into the overly general parser to produce the final parser, a shift-reduce parser that only applies operators to parsing states for which they have been deemed appropriate.

As mentioned previously, the CHILL system was tested on a database task for U.S. geography. Sample questions in English were obtained by distributing a questionnaire to 50 uninformed subjects. The questionnaire provided the basic information from the online tutorial supplied for an existing natural language database application, including a verbatim list of the type of information in the database and sample questions that the system could answer. The result was a corpus of 250 sentences that were then annotated with the appropriate database queries. This set was then split into a training set of 225 examples, with 25 held out for testing.

Tests were run to determine whether the final application produced the correct answer to previously unseen questions. Each test sentence was parsed to produce a query. This query was then executed to extract an answer from the database. The extracted answer was then compared to the answer produced by the correct query associated with the test sentence. Identical answers were scored as a correct parsing; any discrepancy was scored as a failure. Figure 11 shows the accuracy of CHILL's parsers over a 10-trial average. The line labeled Geobase shows the average accuracy of the preexisting hand-coded interface on these 10 testing sets of 25 sentences. The curves show that CHILL outperforms the hand-coded system when trained on 175 or more examples. In the best trial, CHILL's induced parser, comprising 1100 lines of Prolog code, achieved 84-percent accuracy in answering novel queries.

Discussion and Future Directions

Unfortunately, it is difficult to directly compare the effectiveness of these three approaches to the semantic parsing problem. They have not been tested head to head on identical tasks, and given the various differences, including how they are embedded in larger systems and the differing form of input and output, any such comparison would be difficult. What can be said is that each has been successful in the sense that it has been demonstrated as a useful component of an NLP system. In the case of systems that compete in ATIS evaluations, we can assume that designers of systems with empirically based semantic components have chosen to use these approaches because they offered leverage over hand-coded alternatives. There is no requirement that these systems be corpus-based learning approaches. The previous CHILL example shows that an empirical approach outperformed a preexisting hand-coded system at the level of a complete natural language application. Of course, none of these systems is a complete answer, and there is much room for improvement in accuracy. However, the overall performance of these three approaches suggests that corpus-based techniques for semantic parsing do hold promise.

Another dimension for consideration is the flexibility offered by these approaches. Particularly interesting is the question of how easily they might be applied to new domains. All three approaches entail a combination of hand-coded and automatically acquired components.

In the case of a purely statistical approach, the effort of crafting a statistical model should carry over to new domains, provided the model really is sufficient to capture the decisions that are required. Creating statistical models involves the identification of specific contexts over which decisions are made (for example, deciding which parts of a parse tree context are relevant for a decision about slot filling). If appropriate, tractable contexts are found for a particular combination of input language and output representation, most of the effort in porting to a new database would then fall on the annotation process. Providing a large corpus of examples with detailed syntactic-semantic parse trees and frame-building operations would still seem to require a fair amount of effort and linguistic expertise.

In the case of SCTs, the annotation task seems straightforward; in fact, the necessary information can probably be extracted directly from SQL queries. In this case, it seems that most of the effort would be in designing the local (syntactic) parser to find, tag, and correctly translate the form of the constraints. Perhaps future research would allow the automation of this component so that the system could learn directly from sentences paired with SQL queries.

The inductive logic programming approach is perhaps the most flexible in that it learns directly from sentences paired with executable queries. The effort in porting to a new domain would be mainly in constructing a domain-specific meaning-representation language to express the types of query that the system should handle. Future research into constructing SQL-type queries within a shift-reduce parsing framework might alleviate some of this burden.

The CHILL system also provides more flexibility in the type of representation produced. CHILL builds structures with arbitrary embeddings, allowing queries with recursive nestings. For example, it can handle sentences such as, What state borders the most states? It is difficult to see how systems relying on a flat listing of possible attributes or slot fillers could be modified to handle more sophisticated queries, where arbitrary queries might be embedded inside the top-level query.

Another consideration is the efficiency of these learning approaches. The simpler learning mechanisms such as statistical and decision tree methods can be implemented efficiently and, hence, can be used with large corpora of training examples. Although training times are not reported for these approaches, they are probably somewhat faster than for inductive logic programming learning algorithms. Inductive logic programming algorithms are probably still too slow to be used practically on corpora of more than a few thousand examples. The flip side of efficiency is how many training examples are required to actually produce an effective interface. The reported experiments with CHILL suggest that (at least for some domains) a huge number of examples might not be required. The training time for the geography database experiments was modest (on the order of 15 minutes of central processing unit time on a midlevel workstation). In general, much more research needs to be done investigating the learning efficiency of these various approaches.

Finally, it is worth speculating on the overall utility of learning approaches to constructing natural language database interfaces. Some might argue that empirical approaches simply replace hand-built rule sets with hand-built corpora annotations. Is the construction of suitable corpora even feasible? We believe that corpus-based approaches will continue to

make significant gains. Even in the traditional approach, the construction of a suitable corpus is an important component in interface design. From this corpus, a set of rules is hypothesized and tested. The systems are improved by examining performance on novel examples and trying to fix the rules so that they can handle previously incorrect examples. Of course, changes that fix some input might have deleterious effects on sentences that were previously handled correctly. The result is a sort of grammar-tweaking–regression-testing cycle to ensure that overall progress is being made. Empirical approaches follow a similar model where the burden of creating a set of rules that is consistent with the examples is placed squarely on an automated learning component rather than on human analysts. The use of an automated learning component frees the system designer to spend more time on collecting and analyzing a larger body of examples, thus improving the overall coverage and quality of the interface. Another possibility is that techniques of active learning might be used to automatically identify or construct the examples that the learning system considers most informative. Active learning might significantly reduce the size of the corpus required to achieve good interfaces. Indeed, extending corpus-based approaches to the entire semantic mapping task, for example, where training samples might consist of English sentences paired with SQL queries, offers intriguing possibilities for the rapid development of database interfaces. Formal database queries could be collected during the normal day-to-day operations of the database. After some period of use, the collected queries could be glossed with natural language questions and the paired examples fed to a learning system that produces a natural language interface. With a natural language interface in place, examples could continue to be collected where input that are not correctly translated would be glossed with the correct SQL query and fed back into the learning component. In this fashion, the construction and refinement of the interface might be accomplished as a side-effect of normal use without the investment of significant time or linguistic expertise.

Conclusion

The field of NLP has witnessed an unprecedented surge of interest in empirical, corpus-based learning approaches. Inspired by their successes in speech-recognition research, and their subsequent successful application to part-of-speech tagging and syntactic parsing, many NLP researchers are now turning to empirical techniques in their attempts to solve the long-standing problem of semantic interpretation of natural languages. The initial results of corpus-based WSD look promising, and it has the potential to significantly improve the accuracy of information-retrieval and machine-translation applications. Similarly, the construction of a complete natural language query system to databases using corpus-based learning techniques is exciting. In summary, given the current level of enthusiasm and interest, it seems certain that empirical approaches to semantic interpretation will remain an active and fruitful research topic in the coming years.

References

Abramson, H., and Dahl, V. 1989. *Logic Grammars.* New York: Springer-Verlag.

Agirre, E., and Rigau, G. 1996. Word-Sense Disambiguation Using Conceptual Density. In Proceedings of the Sixteenth International Conference on Computational Linguistics. Somerset, N.J.: Association for Computational Linguistics.

Anderson, J. R. 1977. Induction of Augmented Transition Networks. *Cognitive Science* 1:125–157.

Brown, J. S., and Burton, R. R. 1975. Multiple Representations of Knowledge for Tutorial Reasoning. In *Representation and Understanding,* eds. D. G. Bobrow and A. Collins. San Diego, Calif.: Academic.

Brown, P. F.; Della Pietra, S. A.; Della Pietra, V. J.; and Mercer, R. L. 1991. Word-Sense Disambiguation Using Statistical Methods. In Proceedings of the Twenty-Ninth Annual Meeting of the Association for Computational Linguistics, 264–270. Somerset, N.J.: Association for Computational Linguistics.

Bruce, R., and Wiebe, J. 1994. Word-Sense Disambiguation Using Decomposable Models. In Proceedings of the Thirty-Second Annual Meeting of the Association for Computational Linguistics, 139–146. Somerset, N.J.: Association for Computational Linguistics.

Cardie, C. 1993. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In Proceedings of the Eleventh National Conference on Artificial Intelligence, 798–803. Menlo Park, Calif.: American Association for Artificial Intelligence.

Choueka, Y., and Lusignan, S. 1985. Disambiguation by Short Contexts. *Computers and the Humanities* 19:147–157.

Cost, S., and Salzberg, S. 1993. A Weighted Nearest-Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning* 10(1): 57–78.

Cover, T. M., and Hart, P. 1967. Nearest-Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13(1): 21–27.

Dagan, I., and Itai, A. 1994. Word-Sense Disambiguation Using a Second-Language Monolingual Corpus. *Computational Linguistics* 20(4): 563–596.

Duda, R., and Hart, P. 1973. *Pattern Classification and Scene Analysis.* New York: Wiley.

Gale, W.; Church, K. W.; and Yarowsky, D. 1992a. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities* 26:415–439.

Gale, W.; Church, K. W.; and Yarowsky, D. 1992b. Estimating Upper and Lower Bounds on the Performance of Word-Sense Disambiguation Programs. In Proceedings of the Thirtieth Annual Meeting of the Association for Computational Linguistics, 249–256. Somerset, N.J.: Association for Computational Linguistics.

Hendrix, G. G.; Sacerdoti, E.; Sagalowicz, D.; and Slocum, J. 1978. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2): 105–147.

Hirst, G. 1987. *Semantic Interpretation and the Resolution of Ambiguity.* Cambridge, U.K.: Cambridge University Press.

Kelly, E., and Stone, P. 1975. *Computer Recognition of English Word Senses.* Amsterdam: North-Holland.

Klein, S., and Kuppin, M. A. 1970. An Interactive, Heuristic Program for Learning Transformational Grammars, Technical Report, TR-97, Computer Sciences Department, University of Wisconsin.

Kuhn, R., and De Mori, R. 1995. The Application of Semantic Classification Trees to Natural Language Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(5): 449–460.

Langley, P. 1982. Language Acquisition through Error Recovery. *Cognition and Brain Theory* 5.

Lavrač, N., and Džeroski, S. 1994. *Inductive Logic Programming: Techniques and Applications.* London: Ellis Horwood.

Leacock, C.; Towell, G.; and Voorhees, E. 1993. Corpus-Based Statistical Sense Resolution. In Proceedings of the ARPA Human Language Technology Workshop, 260–265. Washington, D.C.: Advanced Research Projects Agency.

Lin, D. 1997. Using Syntactic Dependency as Local Context to Resolve Word-Sense Ambiguity. In Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics. Somerset, N.J.: Association for Computational Linguistics.

Luk, A. K. 1995. Statistical Sense Disambiguation with Relatively Small Corpora Using Dictionary Definitions. In Proceedings of the Thirty-Third Annual Meeting of the Association for Computational Linguistics, 181–188. Somerset, N.J.: Association for Computational Linguistics.

Magerman, D. M. 1994. Natural Language Parsing as Statistical Pattern Recognition. Ph.D. thesis, Stanford University.

Miller, G. A., eds. 1990. WORDNET: An Online Lexical Database. *International Journal of Lexicography* 3(4): 235–312.

Miller, S.; Bobrow, R.; Ingria, R.; and Schwartz, R. 1994a. Hidden Understanding Models of Natural Language. In Proceedings of the Thirty-Second Annual Meeting of the Association for Computational Linguistics, 25–32. Somerset, N.J.: Association for Computational Linguistics.

Miller, G. A.; Chodorow, M.; Landes, S.; Leacock, C.; and Thomas, R. G. 1994b. Using a Semantic Concordance for Sense Identification. In Proceedings of the ARPA Human Language Technology Workshop, 240–243. Washington, D.C.: Advanced Research Projects Agency.

Miller, S.; Stallard, D.; Bobrow, R.; and Schwartz, R. 1996. A Fully Statistical Approach to Natural Language Interfaces. In Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, 55–61. Somerset, N.J.: Association for Computational Linguistics.

Mooney, R. J. 1996. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 82–91. Somerset, N.J.: Association for Computational Linguistics.

Mooney, R. J. 1995. Encouraging Experimental Results on Learning CNF. *Machine Learning* 19(1): 79–92.

Muggleton, S. H., ed. 1992. *Inductive Logic Programming.* San Diego, Calif.: Academic.

Ng, H. T. 1997a. Exemplar-Based Word-Sense Disambiguation: Some Recent Improvements. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, 208–213. Somerset, N.J.: Association for Computational Linguistics.

Ng, H. T. 1997b. Getting Serious about Word-Sense Disambiguation. In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How? 1–7. Somerset, N.J.: Association for Computational Linguistics.

Ng, H. T., and Lee, H. B. 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach. In Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, 40–47. Somerset, N.J.: Association for Computational Linguistics.

Pedersen, T., and Bruce, R. 1997a. A New Supervised Learning Algorithm for Word-Sense Disambiguation. In Proceedings of the Fourteenth National Conference on Artificial Intelligence. Menlo Park, Calif.: American Association for Artificial Intelligence.

Pedersen, T., and Bruce, R. 1997b. Distinguishing Word Senses in Untagged Text. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, 197–207. Somerset, N.J.: Association for Computational Linguistics.

Pedersen, T.; Bruce, R.; and Wiebe, J. 1997. Sequential Model Selection for Word-Sense Disambiguation. In Proceedings of the Fifth Conference on Applied Natural Language Processing, 388–395. Somerset, N.J.: Association for Computational Linguistics.

Peh, L. S., and Ng, H. T. 1997. Domain-Specific Semantic-Class Disambiguation Using WORDNET. In Proceedings of the Fifth Workshop on Very Large Corpora, 56–64. Somerset, N.J.: Association for Computational Linguistics.

Pieraccini, R.; Levin, E.; and Lee, C. H. 1991. Stochastic Representation of Conceptual Structure in the ATIS Task. In Proceedings of the 1991 DARPA Speech and Natural Language Workshop, 121–124. San Francis-

co, Calif.: Morgan Kaufmann.

Procter, P. 1978. *Longman Dictionary of Contemporary English.* London: Longman.

Quinlan, J. R. 1993. C4.5: *Programs for Machine Learning.* San Francisco, Calif.: Morgan Kaufmann.

Reeker, L. H. 1976. The Computational Study of Language Acquisition. In *Advances in Computers, Volume 15,* eds. M. Yovits and M. Rubinoff. San Diego, Calif.: Academic.

Resnik, P. 1997. Selectional Preference and Sense Disambiguation. In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How? 52–57. Somerset, N.J.: Association for Computational Linguistics.

Resnik, P., and Yarowsky, D. 1997. A Perspective on Word-Sense Disambiguation Methods and Their Evaluation. In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How? 79–86. Somerset, N.J.: Association for Computational Linguistics.

Rivest, R. L. 1987. Learning Decision Lists. *Machine Learning* 2(3): 229–246.

Rosenblatt, F. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65:386–408.

Schütze, H. 1992. Dimensions of Meaning. In Proceedings of Supercomputing, 787–796. Washington, D.C.: IEEE Computer Society Press.

Schütze, H., and Pedersen, J. 1995. Information Retrieval Based on Word Senses. In Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval, 161–175. Las Vegas, Nev.: University of Nevada at Las Vegas.

Selfridge, M. 1981. A Computer Model of Child Language Acquisition. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 106–108. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Sembugamoorthy, V. 1981. A Paradigmatic Language-Acquisition System. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 106–108. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Siklossy, L. 1972. Natural Language Learning by Computer. In *Representation and Meaning: Experiments with Information Processing Systems,* eds. H. A. Simon and L. Siklossy. Englewood Cliffs, N.J.: Prentice Hall.

Stanfill, C., and Waltz, D. 1986. Toward Memory-Based Reasoning. *Communications of the Association for Computing Machinery* 29(12): 1213–1228.

Warren, D. H. D., and Pereira, F. C. N. 1982. An Efficient Easily Adaptable System for Interpreting Natural Language Queries. *American Journal of Computational Linguistics* 8(3–4): 110–122.

Wilks, Y.; Fass, D.; Guo, C.; McDonald, J. E.; Plate, T.; and Slator, B. M. 1990. Providing Machine-Tractable Dictionary Tools. *Machine Translation* 5(2): 99–154.

Woods, W. A. 1970. Transition Network Grammars for Natural Language Analysis. *Communications of the Association for Computing Machinery* 13:591–606.

Yarowsky, D. 1995. Unsupervised Word-Sense Disambiguation Rivaling Supervised Methods. In Proceedings of the Thirty-Third Annual Meeting of the Association for Computational Linguistics, 189–196. Somerset, N.J.: Association for Computational Linguistics.

Yarowsky, D. 1994. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In Proceedings of the Thirty-Second Annual Meeting of the Association for Computational Linguistics, 88–95. Somerset, N.J.: Association for Computational Linguistics.

Yarowsky, D. 1993. One Sense per Collocation. In Proceedings of the ARPA Human-Language Technology Workshop, 266–271. Washington, D.C.: Advanced Research Projects Agency.

Yarowsky, D. 1992. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In Proceedings of the Fourteenth International Conference on Computational Linguistics, 454–460. Somerset, N.J.: Association for Computational Linguistics.

Zelle, J. M. 1995. Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers. Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin.

Zelle, J. M., and Mooney, R. J. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1050–1055. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hwee Tou Ng is a group head at DSO National Laboratories, Singapore. He received a B.Sc. in computer science for data management (with high distinction) from the University of Toronto, an M.S. in computer science from Stanford University, and a Ph.D. in computer science from the University of Texas at Austin. His current research interests include natural language processing, information retrieval, and machine learning. He received the Best Nonstudent Paper Award at the 1997 ACM SIGIR conference. His e-mail address is nhweetou@dso.org.sg.

John Zelle is an assistant professor of computer science at Drake University where he has been since 1995. He holds a B.S. and an M.S. in computer science from Iowa State University and a Ph.D. in computer sciences from the University of Texas at Austin. He developed the CHILL system as part of his dissertation work and continues to pursue an interest in natural language applications of machine learning. His e-mail address is zelle@zelle.drake.edu.