INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Determining Similarity and Inferring Relations in a Lexical Knowledge Base

by

Stephen D. Richardson

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1997

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/16/96 — Date

Virginia Teller — Chair of Examining Committee

9/16/96 — Date

Executive Officer

Martin S. Chodorow

George E. Heidorn

John A. Moyne

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

DETERMINING SIMILARITY AND INFERRING RELATIONS IN A LEXICAL KNOWLEDGE BASE

by

Stephen D. Richardson

Advisor: Professor Virginia Teller

This dissertation describes the creation of a large-scale, richly structured lexical knowledge base (LKB) from complex structures of labeled semantic relations. These structures were automatically extracted using a natural language parser from the definitions and example sentences contained in two machine readable dictionaries. The structures were then completely inverted and propagated across all of the relevant headwords in the dictionaries to create the LKB.

A method is described for efficiently accessing salient paths of semantic relations between words in the LKB using weights assigned to those paths. The weights are based on a unique computation called averaged vertex probability. Extended paths, created by joining sub-paths from two different semantic relation structures, are allowed in order to increase the coverage of the information in the LKB.

A novel procedure is used to determine the similarity between words in the LKB based on the patterns of the semantic relation paths connecting those words. The patterns

were obtained by extensive training using word pairs from an online thesaurus and a specially created anti-thesaurus.

The similarity procedure and the path accessing mechanism are used in a procedure to infer semantic relations that are not explicitly stored in the LKB. In particular, the utility of such inferences is discussed in the context of disambiguating phrasal attachments in a natural language understanding system.

Quantitative results indicate that the size and coverage of the LKB created in this research and the effectiveness of the methods for accessing explicit and implicit information contained therein represent significant progress toward the development of a truly broad-coverage semantic component for natural language processing.

Acknowledgments

I would like to give many thanks to my colleagues at Microsoft Research who have participated in and supported this research over many years. Included in this group are George Heidorn, Karen Jensen, Lucy Vanderwende, Bill Dolan, and Joseph Pentheroudakis. There is no finer group of people to work with anywhere. In particular, George and Karen have provided encouragement and guidance in my professional and academic career for well over a decade, not to mention their constant and dedicated friendship.

Virginia Teller, my advisor for this dissertation, has done an outstanding job of encouraging and supporting me throughout my research. She has provided just the right mix of guidance, feedback, patience, and motivation, and for this I am most grateful. I also thank the other members of my reviewing committee, Martin Chodorow, George Heidorn, and John Moyne for their valuable feedback, which has significantly contributed to the quality of the finished work.

Because of this dissertation, my children have sacrificed having a father at many of their school and extra-curricular activities. Nevertheless, they have encouraged me, supported me, and even prayed for me during the vast majority of their growing up years that this work has spanned. I love them and am grateful for them.

Above all, my deepest gratitude and love go to my wife Marianna, without whom this dissertation and the research behind it would never have been realized. No one has sacrificed more, supported more, encouraged more, or complained less. In a very literal sense, this is as much her accomplishment as it is mine.

Table of Contents

List of Figures

List of Tables

List of Equations

1. Introduction

The goal of this research is to create a large-scale lexical knowledge base (LKB) from online dictionaries and to develop efficient and effective methods to access information stored explicitly in the LKB as well as to infer information that is only implicit. Central to the task of inferring information from the LKB is a procedure for determining similarity between words. The information in the LKB is in the form of semantic relations between words, and it is intended to be used in natural language processing (NLP) systems principally to perform both structural and lexical disambiguation.

Brought to bear in this research are a combination of methodologies from other dictionary-based (DB), example-based (EB), and corpus-based (CB) research. DB work has focused mainly on the creation of LKBs, specifically taxonomies, from machine-readable dictionaries (MRDs). The aim of most EB research has been to create large example bases, or collections of example relations or phrases, and to develop methods for both the exact and fuzzy matching of incoming text to the stored examples. CB efforts have used quantitative analyses of text corpora to develop statistical models of the relationships between words, including simple co-occurrence as well as deeper similarity relationships. Techniques from each of these three areas of research have been applied to the identification of the precise meaning of words (word sense disambiguation, or WSD) and to the resolution of structural ambiguities (e.g., prepositional phrase attachment) in a variety of natural language (NL) applications, from general parsing to machine translation to speech recognition. Some references will be made to other work in these areas

throughout the chapters describing the specifics of this research, and detailed comparisons are provided in Chapter 5.

This research has been conducted in the context of the Natural Language Processing group at Microsoft Research. The mission of this group is given in the following statement:

> *The goal of our NLP group is to design and build a computer system that will analyze, understand, and generate <text in> natural languages. Our system takes input text, and moves through various stages of linguistic processing, from lexical/morphological analysis through syntax, semantics, and eventually pragmatics and discourse. Our approach makes heavy use of the information available in online dictionaries and other written works, from which we are able to extract a rich knowledge base, which can then be used to bootstrap into increasingly more advanced stages of machine understanding.* (Microsoft 1996)

Various members of the NLP group have participated in building the system that provides the framework for this research, and the work of the main participants over the past five years is described briefly here. George Heidorn, assisted by the author, designed and implemented the basic system, including a linguistic rule writing language ("G"), parsing engine, various system functions (e.g., for data structure manipulation and storage management), and user interface. Karen Jensen wrote the Microsoft English Grammar (MEG), consisting of "G" rules for parsing English text and producing both syntactic and deeper, logical structures. Lucy Vanderwende developed the processes for extracting and

structuring semantic information contained in the definitions and example sentences of online dictionaries and using it to determine correct phrasal attachment during parsing. Joseph Pentheroudakis created and structured the content of the Microsoft Natural Language Dictionary (MIND), the basis for the LKB described in this research, from the data in two different MRDs, as well as implementing the morphological component supporting MEG. William Dolan assisted in creating various versions of MIND and developed a component that sense disambiguates both the words in the semantic structures contained in the LKB as well as the words in text being parsed.

In addition to being a primary developer of the basic NLP system, the author of this dissertation designed and implemented the storage and access software for MIND and for the resulting LKB. The author also enabled the extraction and structuring of semantic information across the entire dictionary, thereby creating the initial version of the LKB, which has recently come to be called MINDNET.

The process of extracting this information, represented as semantic relations (or *semrels*), from the definitions of MIND and then storing it in the LKB is described in the first half of Chapter 2. This is the only portion of this thesis describing work that was performed principally by other members of the Microsoft NLP group (especially Lucy Vanderwende). This process relies on the use of the MEG parser, which is capable of producing syntactic parse trees for the definitions as well as deeper representations known as logical forms. Structural patterns are applied to the output of the parser, and various semrels, such as **Hypernym** (Is-A) and **Purpose** are extracted (e.g., ***car***—**Hypernym**→***vehicle***, ***stove***—**Purpose**→***cook***). The semrels themselves are

connected in semantic relation structures (or *semrel structures*), each one representing the semrels extracted from a single definition.

The method for inverting semrel structures, which is the beginning of work that is unique to the research of this thesis, is described in the second half of Chapter 2. The inversion process consists of taking each of the semrel structures extracted from the definitions, inverting it so that all of the relations are stored relative to a particular node in the structure, and then storing the inverted structure with the word that is represented by that node. The process is repeated for each node in each semrel structure, effectively creating a massively connected network of semrel structures that can be easily accessed from any word in the LKB. In the discussion, the benefits of this architecture, and the linguistic coverage it provides, are compared favorably against the inherent redundancy and size of the enhanced version of the LKB that is created.

Chapter 3 begins by discussing the need to manage the complexity and volume of information created by the inversion process. The proposed solution is to implement a scheme for assigning weights to each semrel and each combination of semrels contained in semrel structures. After experimentation with weights based on an analogy with the term and inverse document frequency weights used in many information retrieval systems, and further experimentation with weights based on mutual information scores, a novel weighting scheme based on *averaged vertex probabilities* is implemented.

In the last section of Chapter 3, formulas for assigning weights to entire *semrel paths* are provided. A semrel path is simply a linear sequence of semrels contained in a single semrel structure. The notion of an *extended* semrel path is also introduced, which

is a complex path formed by the joining of two paths from separate semrel structures. An algorithm is described for obtaining the highest weighted extended and non-extended paths between two words. When adequately constrained and appropriated weighted, extended semrel paths provide yet another significant increase in the coverage of the information stored in the LKB.

Given the ability to obtain the highest weighted semrel paths between any two words in the LKB, Chapter 4 details a novel procedure for determining the similarity of two words. The procedure relies on the observation that certain semrel path *patterns* (i.e., sequences of relations, without the specific words contained in a path) between words are strong indicators of similarity. For example, when two words share a common hypernym (**Hyp**), as in the path ***pen*—Hyp→*instrument*←Hyp—*pencil***, the words are often strongly similar. Paths denoting similarity are often symmetrical, as in the foregoing example, but not necessarily so. A number of experiments are performed in which frequencies of similarity path patterns are first obtained by querying the LKB for the paths between many thousands of word pairs from an online thesaurus. The most frequently occurring path patterns are then used to determine the similarity of other word pairs from the thesaurus which were not in the training set. The ability of the procedure to identify dissimilarity is also measured using a specially constructed *anti-thesaurus*. Through successive enhancements to the similarity procedure, and most significantly, through the scaling up of the LKB to include semrel structures extracted from an additional MRD, the procedure is shown to be nearly as effective as humans in

determining similarity, based on an experiment in which the performance of a small group of humans is compared to that of the similarity procedure.

The final section of Chapter 4 is devoted to describing the use of the similarity procedure to infer semrels that are not explicitly stored in the LKB, which was a principal goal of this research. If a particular semrel is not contained in the LKB, then the inference procedure attempts to infer it by looking for a stored semrel in which one of the words is different from, but similar to, the corresponding word in the semrel being sought. After experimenting with different constraints on the inference procedure, it was found that the best results are achieved by limiting the set of similar words allowed to participate in the inference to the words contained in the paths connecting the two words in the sought-after semrel. If the desired relation is found at either end of one of the paths connecting the two words, then the word participating in that relation with one of the two words is checked for similarity with the other of the two words. For example, the path ***observe***←**Hyp**—***watch***—**Hyp**→***look***—**Means**→***binoculars*** allows the inference of the semrel ***observe***—**Means**→***binoculars*** since ***look*** and ***observe*** are designated as strongly similar by the similarity procedure. Experiments were performed that show the inference procedure to produce reasonable results, and the use of the procedure is discussed in the context of a prepositional phrase attachment task in a NL analysis system.

Chapter 5 provides a detailed comparison of the methods used in this research with those found in other DB, EB, and CB work. The comparison is organized according to specific issues that distinguish this research, such as focusing on taxonomies vs. networks, the use of labeled semantic relations vs. simple co-occurrence relations, and the

determination of general relatedness vs. substitutional similarity. The process of comparison also provides insights into the similarity of certain aspects of the general DB, EB, and CB methodologies.

The final chapter concludes that the combination of the methods exploited in this work, both those in common with previous work and those that are unique, constitute a significant step forward in support of broad-coverage semantic processing in NL systems. Specifically, the unique contributions of this research include:

1. The creation of a richly structured LKB through the complete inversion and propagation of automatically extracted, complex structures of labeled semantic relations representing the information contained in the definitions and example sentences of two machine readable dictionaries.

2. The formulation and efficient accessing of salient *extended* paths of semantic relations between words in the LKB, based on the computation and assignment of *averaged vertex probability* weights to those paths.

3. The development of a novel procedure for determining similarity between words based on the patterns of the semrel paths connecting those words, the patterns being obtained by extensive training based on word pairs from an online thesaurus.

4. The implementation of a procedure to infer semantic relations that are not explicitly stored in the LKB, based on the use of the above-mentioned extended semrel paths and similarity procedure.

Areas for further research are identified at the end of the concluding chapter. These include refinements to increase the accuracy of automatic semrel extraction and extended path formation, as well as extensions to increase the coverage of the semrels in the LKB. Such improvements would, of course, positively affect the coverage and accuracy of the similarity and inference procedures as well.

2. Extracting and inverting semantic relation structures

This chapter describes the methods used to create the lexical knowledge base (LKB) that is central to the topic of this research. Initially, the LKB was created through automatically extracting and structuring information contained in the Longman Dictionary of Contemporary English (Longman 1978), hereafter referred to as LDOCE. The version of LDOCE used as the basis for the LKB was the machine-readable, *lispified* version described by Alshawi et al. (1989), in which dictionary information is represented in the form of Lisp S-expressions.

In 1995, information contained in the *American Heritage Dictionary, 3rd Edition* (Houghton Mifflin Company 1992) was also added to the LKB. This addition will be described as it pertains to the results of this work later in Chapter 4. For now, it should be noted that all definitions cited in this chapter and in Chapter 3 are taken from LDOCE, with possible minor typographic variations.

There has been much research on the extraction of information from machine-readable dictionaries, including that of Byrd et al. (1987), Klavans et al. (1990), Markowitz et al. (1986), Vossen (1991), and Wilks et al. (1989, 1992). Work from these and other references is described in greater detail in Chapter 5. This research is based on previous work in the same area begun by Jensen and Binot (1987), continued by Montemagni and Vanderwende (1992), and further refined and extended by Vanderwende (1996).

2.1 Representing dictionary entries

In the context of the Microsoft Natural Language Processing (NLP) system in which this research has been performed, linguistic information is represented in data structures known as *records*. A record is an object that contains one or more attributes, each of which has one or more values (Jensen et al. 1993). This representation is thus of the straightforward entity-attribute-value variety commonplace in present-day relational data base systems.

The author designed and implemented a file access method having an architecture capable of storing, modifying, and retrieving dictionary data represented in this record format. This access method was integrated seamlessly into the Microsoft NLP system so that dictionary information could be accessed transparently, as if it were resident with other linguistic data (represented in records) in the system, for purposes such as syntactic parsing and semantic disambiguation. The lispified version of LDOCE was then converted to the required input format and loaded into the NLP system, thus becoming the basis for the MIcrosoft Natural language Dictionary (MIND). Since 1992, the data in MIND has been extended and enhanced (including through the addition of data from AHD3) to support inflectional and derivational morphological processing, syntactic parsing, structural attachment resolution, and word sense disambiguation. The latter two processes, in particular, make heavy use of the information described in this chapter, which has been automatically extracted from the dictionary definitions.

Part of the dictionary entry in MIND for the word ***fruit*** is shown in Figure 2.1. Braces { } define the beginning and end of a record, with the names of attributes

appearing in the left-hand column in the record and the corresponding values of those attributes in the right-hand column. The value of an attribute may itself be one or more records, hence there is a nesting effect. In the figure, the topmost record has four attributes, the first of which is **Word**, which has the value "fruit". The next two attributes, **Noun** and **Verb**, each have a single record as their value. These two records (called *part-of-speech* records) contain information for each possible part-of-speech of ***fruit*** that is required for syntactic parsing, that information being gleaned and combined from the records contained in the fourth attribute, **Senses**. Each record contained in the **Senses** attribute (called *sense* records) contains the information from one word sense as given in LDOCE. In the figure, only the single most common noun sense record and verb sense record are shown. Some of the attributes included on the nested records shown include: **Bits**, linguistic binary features (including verb subcategorization features from LDOCE); **Infl**, the inflectional paradigm of the word; **Ldoce**, a reference to the specific LDOCE word sense; **Defin**, the definition text for the sense; **Exs**, example sentences for the sense; **Boxc**, LDOCE box codes (inherent semantic features); and **Subjcl**, LDOCE subject codes (indicating domain or topic).

```
(Word        "fruit"
 Noun
        (Segtype     NOUN
         Lex         "fruit"
         Lemma       "fruit"
         Bits        Pers3 Sing Plur Wn1
                     Wn2 Count Mass Food
         Infl        Noun-default
         InflForms   (fruits) )
 Verb
        (Segtype     VERB
         Lex         "fruit"
         Lemma       "fruit"
         Bits        Inf Plur Pres I0
         Infl        Verb-default )
 Senses
        (Bits        Count
         Ldoce       1001
         SenseNo     103
         Cat         Noun
         Infl        Noun-default
         Defin       "an object that grows on a tree or bush,
                      contains seeds, is used for food, but is not
                      usu. eaten with meat or with salt"
         Boxc        "----J"
         Subjcl      (FO) )
          .
          .
          .
        (Bits        I0
         Ldoce       2001
         SenseNo     109
         Cat         Verb
         Infl        Verb-default
         Defin       "(of a tree, bush, etc.) to bear fruit"
         Exs         "The apple trees are fruiting early this year"
         Boxc        "----P"
         Subjcl      (AG) ) )
```

Figure 2.1. Part of a dictionary entry for the word **fruit**

2.2 Extracting semantic relations

Once the information from LDOCE has been loaded into the system-tractable representation of MIND, it becomes possible to extract semantic information from the LDOCE definitions and augment MIND with structured representations of this information. Although this process is the work of Vanderwende (1996), its description is

provided here as background material helpful to understanding the unique contributions of the research for this dissertation.

The first step in the extraction process is to parse the LDOCE definitions in MIND with a broad-coverage grammar of English; the Microsoft English Grammar (MEG) is used for this purpose. MEG produces approximate syntactic parses of general English text segments (sentences as well as fragments) by adhering to principles and organization similar to those outlined by Jensen (1993a), the author of MEG.

MEG consists of a comprehensive set of augmented phrase structure grammar (APSG) rules, which are applied to input text by a bottom-up parsing engine similar to the chart-like parser described by Jensen et al. (1993). At the beginning of parsing, each word in the input text is looked up in MIND, with morphological processing being performed as necessary to handle inflectional variations, and the information from the part-of-speech records for that word is placed in the chart. The APSG rules are then applied until a parse structure is obtained for the input text. A soft failure mechanism allows for "fitted" structures to be obtained for fragments or other ungrammatical text. By making use of the syntactic information stored in the very dictionary to which it will add structured semantic information, the extraction process, through its use of MEG, exhibits a *bootstrapping* behavior. MEG itself also becomes a beneficiary of this bootstrapping, as the information thus obtained is later used to enhance semantically the parse structures MEG produces.

A parse structure produced by MEG for one of the noun definitions of ***fruit*** is shown in Figure 2.2. The structure is similar to a *dependency* tree, in that there is a *head*

node at each level in the structure, indicated by an asterisk (*), with *pre-modifiers* before it and *post-modifiers* following it. Each node in the structure is labeled by a node type followed by a number that distinguishes it from other nodes of the same type in the same structure. The following brief description of some of the node types in the structure may be helpful in its interpretation: an NP is a noun phrase; a RELCL is a relative clause; a VP is a verb phrase (often an entire predicate); and a PP is a prepositional phrase. Thus, the definition represented by this structure may be characterized as one large noun phrase, with its head noun ***object*** being modified by a relative clause containing four conjoined verb phrases, their head verbs being ***grows***, ***contains***, ***used***, and ***eaten***.

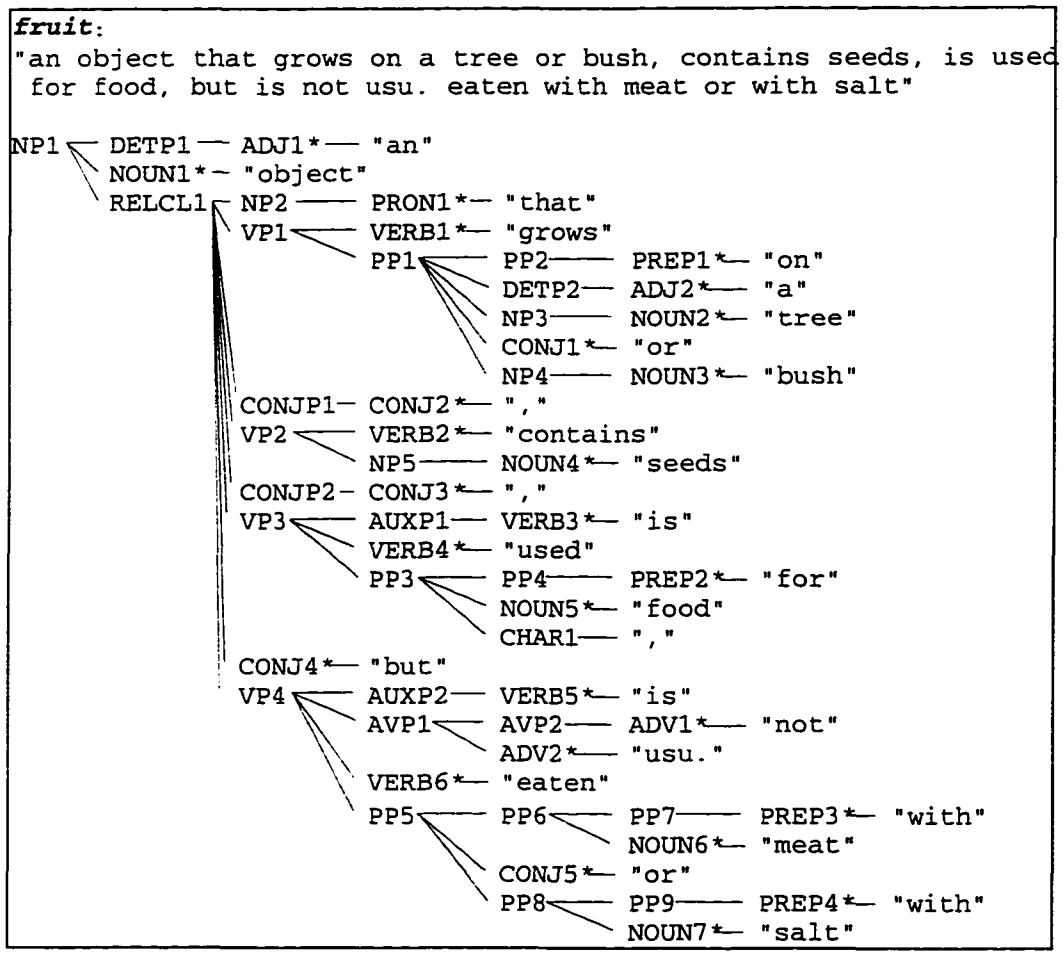

Figure 2.2. Parse structure for a noun definition of **fruit**

Vanderwende (1996) has argued convincingly that using parse structures such as those produced by MEG provides a framework for extracting information from on-line dictionaries that is superior to other methods. The reader is referred to her work for an extended treatment of this issue, as it is sufficient for the research at hand to state that the use of such structures enables the extraction of a rich set of *semantic relations* from dictionary definitions.

A semantic relation consists of a unidirectional link, labeled by a semantic relation type, between two words. For example, one might say that in the definition of ***fruit*** shown in Figure 2.2, the verb phrase ***is used for food*** indicates a **Purpose** relation between ***fruit*** and ***food***, i.e., ***food*** is a **Purpose** of ***fruit***. In this example, **Purpose** is the semantic relation type, and the relation is unidirectional; the converse (***fruit*** being a **Purpose** of ***food***) is not true.

The **Purpose** relation type is in fact one of the semantic relation types obtained by the extraction process. Each semantic relation has associated with it one or more structural patterns that are matched against the parse structure produced by MEG for each definition during extraction. The pattern for the **Purpose** relation type may be described (in part) as follows:

> *If the head verb of a verb phrase is* **used** *and it is post-modified by a prepositional phrase containing the preposition* **for**, *then create a* ***Purpose*** *relation between the word being defined and the object(s) of* **for**. (adapted from Vanderwende 1996)

By applying the above pattern to the parse structure for the definition of ***fruit***, a **Purpose** relation is indeed created between ***fruit*** and ***food***. This relation is then stored on the sense record containing the definition in MIND. In like manner, structural patterns for the other semantic relations are also applied to the parse structure for the definition, and any resulting semantic relations are stored on the sense record as well.

The different types of semantic relations currently created by the application of structural patterns during the extraction process are enumerated in Table 2.1. The table contains a (possibly) shortened relation type name in the first column that is used when a relation is stored on a sense record. An extended name and/or brief description (in parentheses) appears in the second column if needed for clarification.

Relation Type	**Extended name** (description)
Cause	
Domain	(subject area)
Hyp	**Hypernym** ("is a")
Locn	**Location**
Manner	
Matr	**Material** (ingredient)
Means	
Part	(component)
Possr	**Possessor**
Purp	**Purpose**
Quesp	**Quespernym** (quasi-hypernym derived from "used as" or "such as")
Syn	**Synonym**
Time	
Tobj	**TypicalObject** (of a verb)
Tsub	**TypicalSubject** (of a verb)
User	

Table 2.1. Semantic relation types

When the process of applying structural patterns and creating and storing semantic relations on all of the sense records for a given word is complete, the dictionary entry for that word is updated and the extraction process moves on to the next word in MIND. In the case of the ***fruit*** example, the updated sense record for the definition in Figure 2.2 is shown in Figure 2.3. In this sense record, semantic relation types are represented as attributes, and the values of these attributes are records that have a **Lemma** attribute containing the second word in the relation. The implicit first word in each case is ***fruit***, being the headword of the dictionary entry that contains this sense record. By default, the direction of each relation is from ***fruit*** to the words in the records that are values of the semantic relation attributes. A reversed direction is indicated by the addition of **Of** to the attribute name. Thus, while ***food*** is the **Purpose** of ***fruit***, ***fruit*** is the **TypicalSubjectOf** ***grow***. One additional point of interest is the **Location** relation that was extracted between ***grow*** and ***tree*** (and ***bush***). The usefulness of such semantic relations that do not directly involve the headword of the dictionary entry will be discussed in the next section.

```
(Bits       Count
 Ldoce      1001
 SenseNo    103
 Cat        Noun
 Infl       Noun-default
 Defin      "an object that grows on a tree or bush,
             contains seeds, is used for food, but is
             not usu. eaten with meat or with salt"
 Boxc       "----J"
 Subjcl     (FO)
 Hyp
       (Lemma      "object" )
 Part
       (Lemma      "seed" )
 Purp
       (Lemma      "food" )
 TsubOf
       (Lemma      "grow"
        Locn
              (Lemma      "tree" )
              (Lemma      "bush" ) )
 TobjOf
       (Lemma      "eat" ) )
```

Figure 2.3. Sense record for **fruit** *containing semantic relations extracted from the definition*

Another point needs to be made before leaving the topic of semantic relation extraction. It is that the parse structures generated by MEG contain a great deal more information than that which is visible in Figure 2.2. Each node in the parse structure is represented by a record containing many attributes with their values. Among these are functional attributes, such as **Subject** and **Object**, that indicate the node in the parse structure representing the words or phrases that function in these roles. Attributes such as these may be accessed by structural patterns to create **TypicalSubject** and **TypicalObject** relations, however, they do not always provide the desired reference.

In the definition of ***fruit*** examined above, the passive verb ***eaten*** does not have a grammatical object in the traditional sense. At a deeper level, however, one may claim that its object is really ***fruit*** (indirectly through the reference of ***object***). This can be seen

plainly if the last verb phrase in the definition is rephrased into the active voice: ***but one does not usu. eat fruit with meat or with salt***. It is exactly this *deep object* identification that is provided by a processing step that follows syntactic parsing by MEG. This step is analogous to a similar process described by Jensen (1993b) and results in the creation of an additional structural representation of the input text called the *logical form*. The logical form for the definition of ***fruit*** is given below in Figure 2.4. The processing that creates the logical form identifies deep subjects and deep objects of passive verbs and of verbs in infinitive phrases or in conjoined phrases where an argument has been elided. It also resolves anaphoric references of pronouns. By accessing the information contained in the logical form for a definition, structural patterns gain additional power and are able to create semantic relations that would be overlooked by other less sophisticated methods. It is because of the information contained in the logical form, specifically the *eat1*—**Dobj**—*object1* relationship (see Figure 2.4), that the **TypicalObject** relation between ***eat*** and ***fruit*** in Figure 2.3 was able to be identified and created.

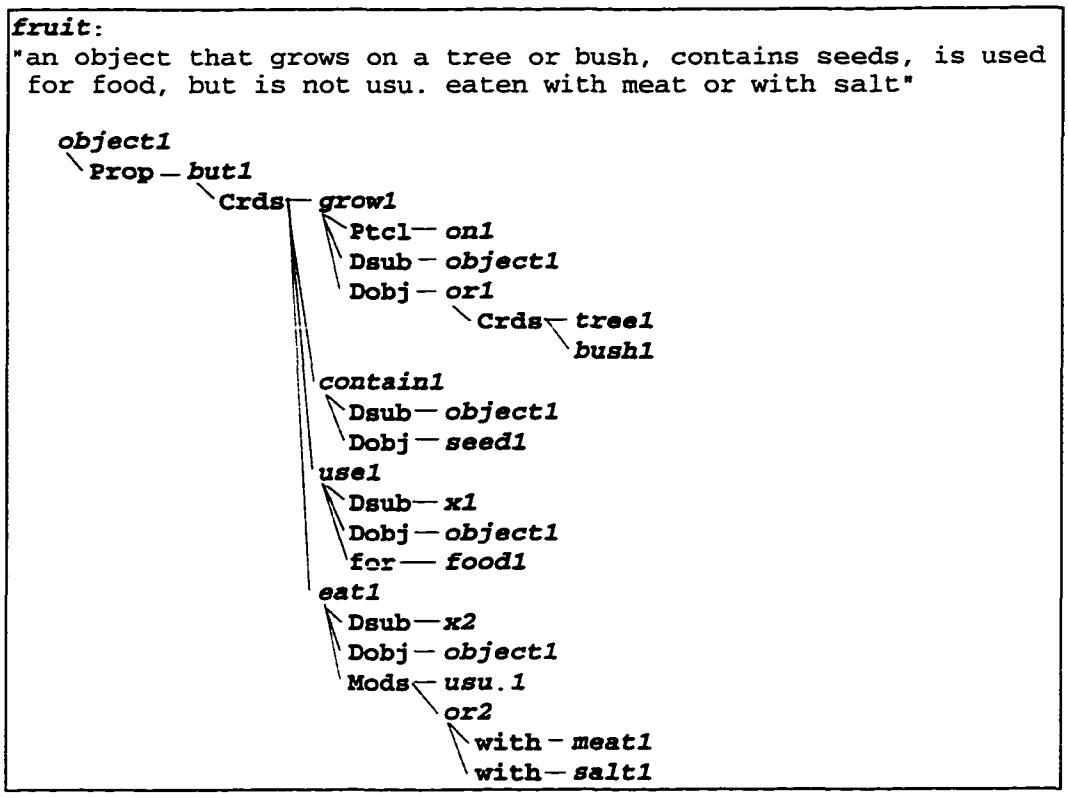

Figure 2.4 Logical form for a definition of **fruit**

Through the application of structural patterns to the parse structures and logical forms of definitions, vast numbers of semantic relations are able to be extracted automatically across the entire dictionary and added to each entry. Already, this extraction process provides a rich set of structured information about each word in the dictionary. However, the information in the form shown in Figure 2.3 does not yet provide the kind of linking and access that is required for the powerful similarity and inference functions developed for this research. This issue will be addressed in the next section.

2.3 Inverting semantic relation structures

Although it may not have been obvious in the sense record representation shown in Figure 2.3, the cluster of semantic relations extracted from a given definition actually

forms a directed acyclical graph structure, referred to hereafter as a *semantic relation structure*. This structure has as its root node the headword of the entry containing the definition. The semantic relation structure for the definition of ***fruit*** discussed above is shown in Figure 2.5. The structural representation shows succinctly the individual semantic relations that were extracted and their relationship with one another. Directionality of the relations, indicated by arrow-like brackets <> on either side of the relation type, is visually more intuitive, but the relations may still be read by inserting the words ***has a*** before a left-to-right relation and the words ***is a*** and ***of*** on either side of a right to left relation. Thus, in Figure 2.5, ***fruit*** has a **Hypernym** ***object***, and ***fruit*** is a **TypicalSubject** of ***grow***.

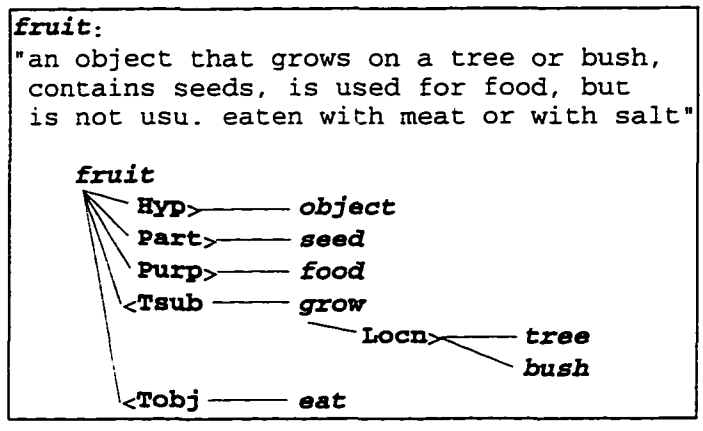

Figure 2.5. Semantic relation structure for the definition of **fruit**

Although the definition and corresponding semantic relation structure in Figure 2.5 provide a useful set of information about ***fruit***, it would by no means be considered a complete or even near complete set. Definitions often fail to express many basic facts about word meanings, facts that should obviously be included in an LKB that would be useful for broad-coverage natural language understanding. Another example in LDOCE that illustrates this point is shown for the word ***car*** in Figure 2.6. Although the

information in this example is indeed useful, it mentions only one of potentially dozens or even hundreds of generally recognized ***car*** parts, and only one action performed with a ***car*** and one purpose of a ***car***.

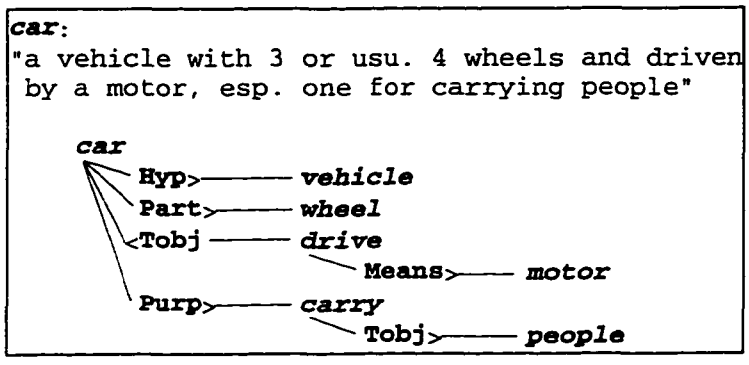

Figure 2.6. Semantic relation structure for a definition of **car**

In order to overcome this ostensible limitation on the information that can be extracted, one might access the semantic relation structures associated with the words in the structure in Figure 2.6 (e.g., ***vehicle***, ***wheel***) in a way that is similar to the forward spreading activation in the networks of Veronis and Ide (1990) or Kozima and Furugori (1993). In this case, however, such a strategy is not especially productive, yielding some information already given by the definition of ***car***, as well as more general information (about ***vehicles***, ***wheels***, etc.), but not many new details specific to ***cars***.

Another more effective solution is to access the semantic relation structures associated with words that are related to car by those structures. This is the same as looking up words in the dictionary whose definitions mention the word car. This research has verified that there is a great deal of additional information about a given word's meaning that is typically stored not in the entry for that word itself, but rather in the entries for other words that mention that word. For instance, it is relatively unusual to

find the words which describe the parts of some object in the dictionary entry for that object; instead, the relationship between the words for these parts and the larger object is defined generally in the dictionary definitions for the parts themselves. For example, in the semantic relation structure extracted from the definition of car, only wheels are shown to be Part of a car. In Figure 2.7, however, the structures from the definitions of fender and trunk also show them in a **Part** relation with ***car***.

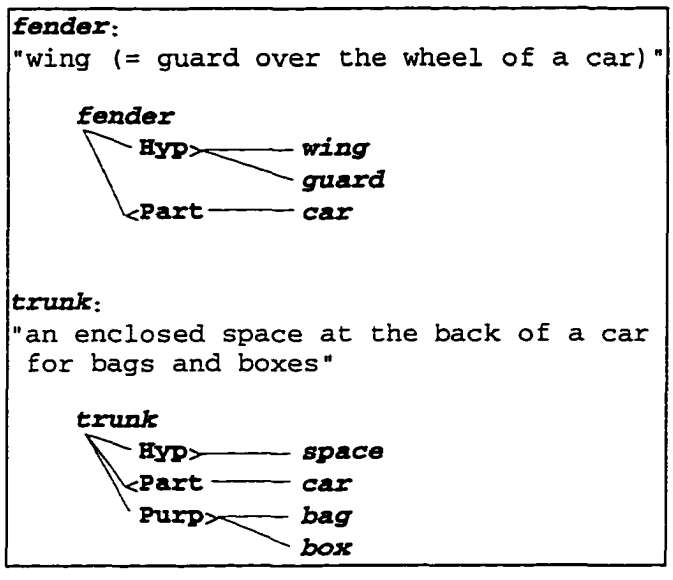

Figure 2.7. Semantic relation structures for definitions of **fender** *and* **trunk**

Additional **Part** relations for ***car*** may be found in the entries of yet other words that may or may not be ***car*** parts. This is illustrated by the structures from the definitions for ***hood***, ***demist***, and ***peep*** in Figure 2.8. These three structures show that ***engine***, ***window***, and ***horn***, respectively, are all **Part** of a ***car***. As stated above, any definition that mentions ***car*** may result in the creation of any semantic relation, although certain relations, such as **Hypernym**, usually involve the headword of a dictionary entry directly.

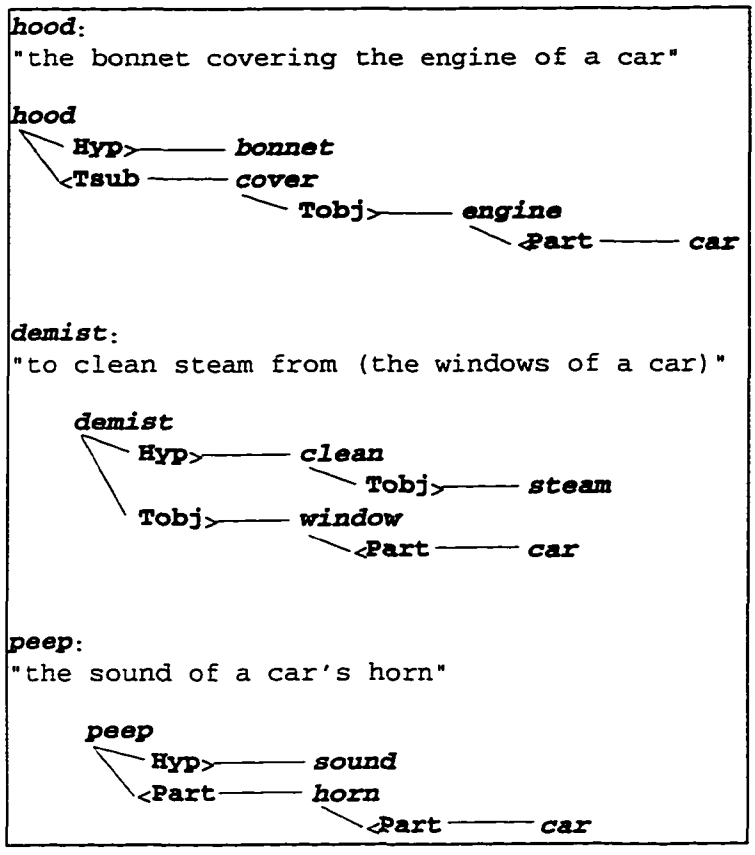

Figure 2.8. Semantic relation structures for definitions of **hood**, **demist**, *and* **peep**

The explicit referencing of semantic relations for a word contained in the semantic relation structures associated with other words has been called *backlinking* by Dolan et al. (1993) and Richardson et al. (1993). For each word in MIND, forward-linked semantic relations are those in the semantic relation structures derived from the definitions of that word. Backward-linking semantic relations are those in structures derived from definitions of other words. It is these backward-linking semantic relations, and other relations in their associated structures, that serve to dramatically increase the amount of information accessible for a given word. By making these backward-linked relations explicit in MIND together with the already explicit forward-linked relations, the dictionary thus becomes a highly interconnected network of semantic information. This network constitutes the essence of the LKB used in this research.

There is one unfortunate aspect of backlinking that is specific to the use of LDOCE. Although the limited defining vocabulary of LDOCE has been lauded as a simplifying factor by some computational linguists, it creates a situation in which backward-linking semantic relations are limited to words in the defining vocabulary. This does provide a substantial amount of information for the roughly 2,000 words in that vocabulary, but it also severely restricts the amount of information available for words outside the vocabulary. The only solution to this problem is to incorporate semantic relation structures derived from the example sentences in LDOCE and from definitions and example sentences contained in other unrestricted dictionaries. Such an augmentation was actually performed for a later stage of this research, and it is described in detail in Chapter 4. However, for the present discussion, and for the development of the basic methods of this research, information and examples from LDOCE are entirely sufficient.

The explicit backward-linking of semantic relations in MIND is accomplished by *inverting* the semantic relation structures for the definitions in a dictionary entry and then propagating those inverted structures onto the entries of all the words they refer to. The inversion of semantic relation structures is the first of the unique contributions of this research. Work described to this point, including the basic NL system architecture, semrel extraction, and the notion of backlinking, has involved, in addition to the author, other members of the Microsoft NLP group. Henceforth, the ideas and implementations described will be the unique contributions of the author, except where the participation of others is noted explicitly.

The process of inversion consists of a recursive traversal of each semantic relation structure, and at each node visited in the structure, beginning another traversal which outputs a copy of the structure to a file, the copied structure being reoriented so that the visited node is the root node. The file of inverted structures is sorted according to the words on the new root nodes, and the structures are then stored in the dictionary entries for these words. One important aspect of this process is the maintenance of the correct directionality for each semantic relation as a structure is inverted, which often must be reversed as nodes are brought into the root position in the structure. Examples of inverted semantic relation structures for ***hood***, ***demist***, and ***peep*** that bring ***car*** into the root node position, thereby making explicit the **Part** relations discussed earlier, are shown in Figure 2.9.

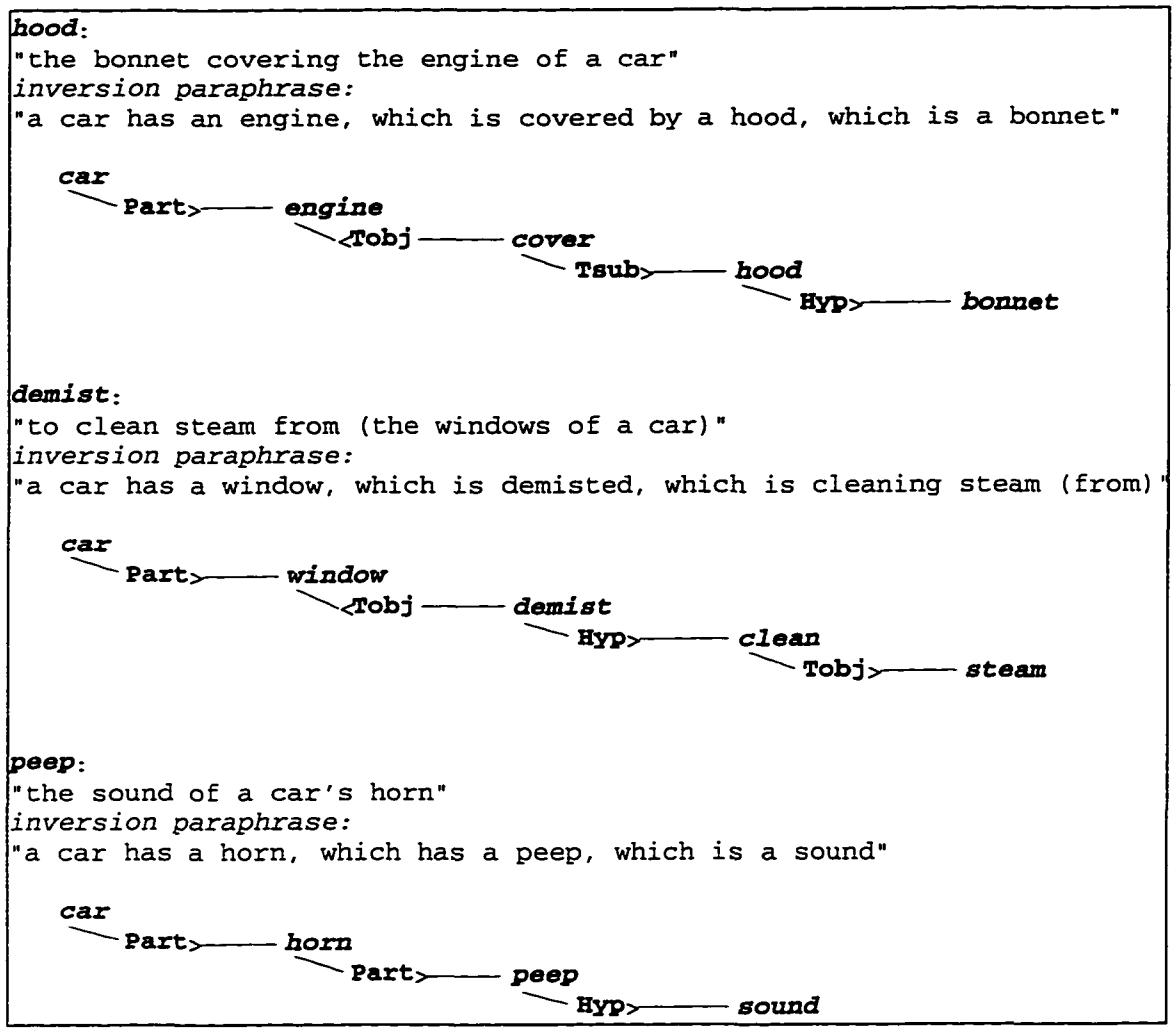

Figure 2.9. Inversions of semantic relation structures shown in Figure 2.8

A metaphor that is helpful in understanding inversion is a mobile. If a semantic relation structure were a mobile hanging by its root node, different inversions of the structure could be obtained by successively picking up the mobile by a different node, each time having the rest of the structure hang neatly (this is where the metaphor stretches) down from that node.

The process of inversion provides direct access from a word to every semantic relation involving that word, no matter what semantic relation structure contained it anywhere in the dictionary. Figure 2.10 shows examples of semantic relations, other than

the **Part** relations that have been discussed, that are made accessible for ***car*** through inversion. Paraphrasing, these contain the information that ***motorists drive cars***, ***cars pass other cars***, and ***convertibles are cars***. Every semantic relation that is extracted, even and sometimes especially those that do not involve the headword of a dictionary entry (e.g., the **Location** relation between ***grow*** and ***tree*** mentioned back in the discussion of Figure 2.3), can provide significant information about a word in the resulting LKB. The inverted semantic relation structures also identify linear sequences of relations between words, the importance of which will be discussed in the next chapter.

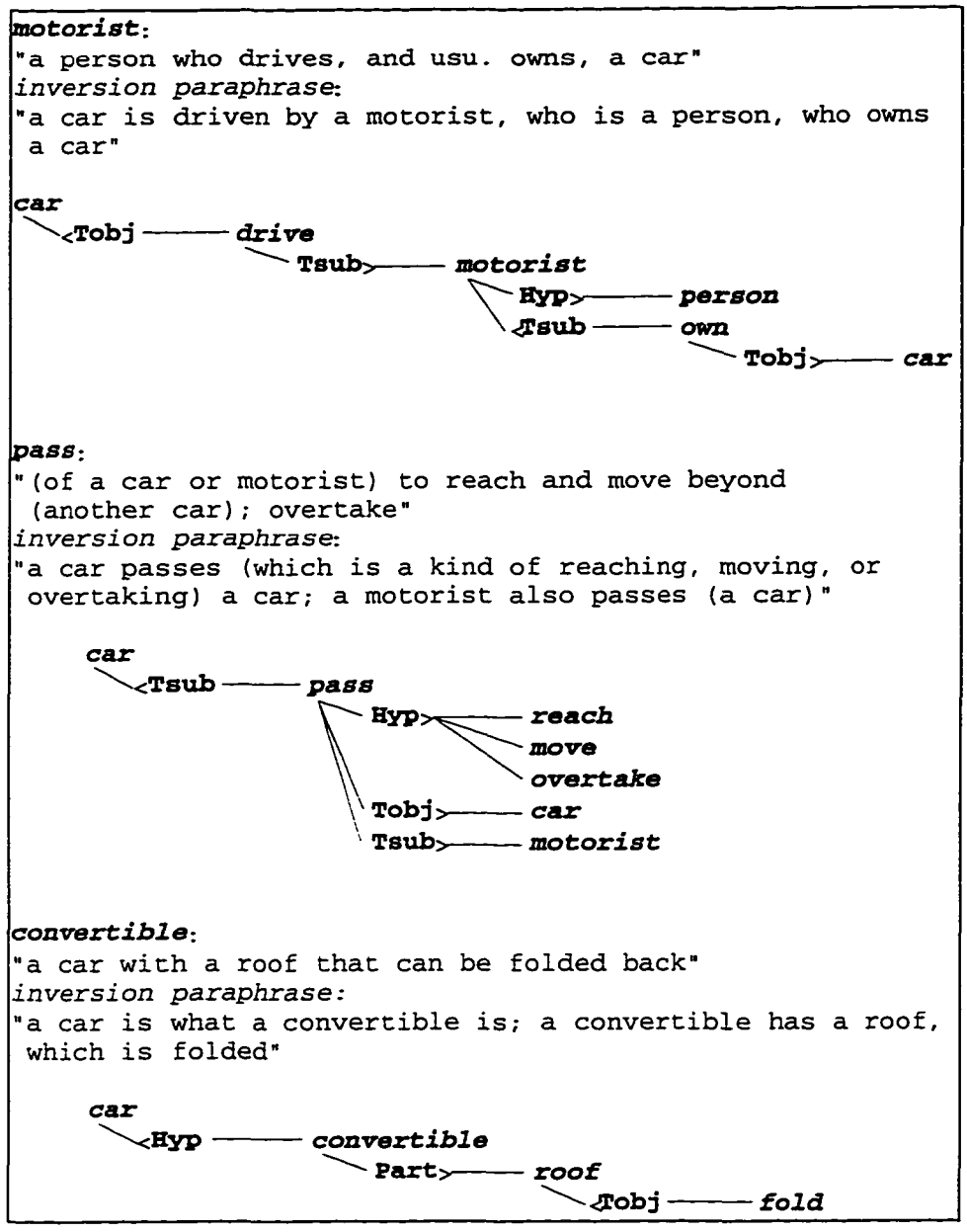

Figure 2.10. Inverted semantic relation structures with other than ***Part*** *relations for* **car**

By inverting some of these semantic relation structures, certain deficiencies in the structures themselves and in the extraction process that created them become more evident. In the structure for ***motorist***, the fact that the same car is being driven and owned is not maintained in the semantic relation structure, although it was originally present in the logical form generated for the definition. In addition, information from adverbs and

adjectives is not currently represented. Differing senses of the words in inverted structures are problematic when trying to combine the information contained in those structures for a given word. Some of these concerns, such as the word sense problem, will be discussed in later chapters, while others, such as modifier representation, must await refinements in the extraction process. As always, there is room for improvement in all phases of this work, but as they now stand, the inverted semantic relation structures described herein have proven to be immensely useful in achieving the goals of this research.

A comment should be made about the storage of the inverted semantic relation structures. It certainly would have been a more efficient use of space to store only one copy of each structure and then index that structure to each word it referenced. The typical space-time tradeoff comes into play, however, as this would make accessing the structures from the perspective of any particular word (node) more expensive computationally. Since the semantic relations are directional in nature and need to be oriented according to the node at which the structure is *entered*, this would necessitate a sort of mini-inversion process each time a structure was accessed.. The next chapter also discusses information (weights) which is added to these structures, and once again, it appears to be more important to pre-compute and store this information relative to the entry node (root node in the inverted structure) than to have to compute it during each access. Hence, the desirability of storing redundant inverted structures, even though the size of the LDOCE-based LKB is almost doubled (from 15MB to 28MB) by so doing.

To provide an idea of the quantity and quality of information contained in the LDOCE-based LKB version of MIND (containing inverted semantic relation structures), the following information, rounding numbers to the nearest thousand, is supplied:

- Semantic relation extraction (including parsing by MEG) processes 33,000 single word noun definitions and 12,000 single word verb definitions (a total of 45,000 definitions out of the 75,000 definitions in MIND) in about 4 hours on a Pentium 120 PC. Definitions of adjectives, adverbs, and phrasal entries are currently not processed. As a result of extraction processing, over 147,000 semantic relations contained in close to 45,000 semantic relation structures are created.

- Semantic relation structure inversion processes the 45,000 structures created during extraction to create almost 180,000 inverted structures, stored on 23,000 of the 39,000 headwords in MIND. A sampling of the number of inverted semantic relation structures produced for various words is given in Table 2.2. Approximately 120 words have more than 200 inverted structures each. A maximum of 800 structures are currently stored per word because of data base limitations, but this affects only a handful of the most common words.

Word	Number of inverted structures
person	3130
place	1149
animal	733
play	578
water	474
grow	339
book	280
fruit	211
car	119
street	47

Table 2.2. Numbers of inverted semantic relation structures generated for various words

- In related work published earlier (Richardson 1993), the author reported on a hand-checked random sample of 250 semantic relations from across the dictionary and found them to be 78% correct overall. Using common statistical techniques it was estimated that this accuracy rate is representative of all the relations in the dictionary with a margin of error of +/- 5%. Of note is that just about half of the relations in the sample were **Hypernym** relations, which were accurate 87% of the time. Other relations were not so accurate, accounting for the overall rate of 78%. A number of improvements have been made to the extraction process since that report.

The LDOCE-based LKB created by the processing described above represents one of the largest and most deeply processed knowledge bases ever produced for an NLP system by automatic means. Furthermore, the addition of information from another dictionary, as described in Chapter 4, more than triples the amount of information in the LKB, making it a truly unique and powerful resource for natural language understanding systems.

3. Assigning weights to semantic relation paths

In the last chapter, the creation of a highly connected lexical knowledge base (LKB) from the definitions of a machine-readable dictionary (LDOCE) was described. Semantic relations (or *semrels*) are extracted by first parsing the definitions with a NL parser, and then by matching patterns to the resulting parse structures. A cluster of semrels extracted from a definition is represented in a semantic relation structure (or *semrel structure*). When these semrel structures are fully inverted and propagated across all of the headword entries in the dictionary, a richly-connected network of information results.

For example, Figure 2.9 in the last chapter showed some of the **Part** relations accessible for ***car*** through the inversion of three semrel structures. Table 3.1 below lists the 15 different **Part** relations in the inverted semrel structures from 21 definitions in LDOCE. It is interesting to note that of the 21 definitions, only one is actually from the headword ***car*** in the dictionary. This is further evidence of the utility of semrel structure inversion.

By having all of the semrel structures that refer to a word directly stored with that word, it is also possible to query the LKB efficiently with a headword and a specific relation type, and quickly obtain a list of all of the semrels associated with that headword which contain the specified relation type, exactly as shown in Table 3.1. The author has implemented this function, and it is an integral part of the processing used by the Microsoft NLP system to disambiguate structures and words in NL text. This processing will be described in more detail in Chapter 4.

Semantic relations	Definition headwords
car—Part→body	body
car—Part→boot	boot
car—Part→clutch	declutch
car—Part→cockpit	cockpit
car—Part→dicky	dicky
car—Part→engine	petrol, hood, dipstick
car—Part→fender	fender
car—Part→front	windscreen
car—Part→horn	peep
car—Part→scoop	scoop
car—Part→top	sunroof, grid
car—Part→trunk	trunk
car—Part→underside	pit
car—Part→wheel	wheel, suspension, car
car—Part→window	demister, demist

Table 3.1. ***Part*** *relations obtained from inverted semrel structures for* **car**

Another function implemented by the author, equally important in disambiguation processing, allows the LKB to be queried by specifying a headword and one or more additional words to be searched for among the inverted semrel structures associated with the headword. One form of output from this function is exemplified in Figure 3.1, which consists of 4 inverted semrel structures for the headword ***car*** that also contain the words ***people*** or ***person***. Direct or indirect relationships between ***car*** and ***people*** or ***person*** may be obtained by inspection of these structures. These relationships, consisting of one or more semrels connected together, constitute a *semantic relation path* (or *semrel path*) between two words. The semrel paths contained in the LKB are the basic units of meaning matched against structures in NL text during lexical and structural disambiguation in the Microsoft NLP system, and they are also used to determine similarity between words, as described in the next chapter.

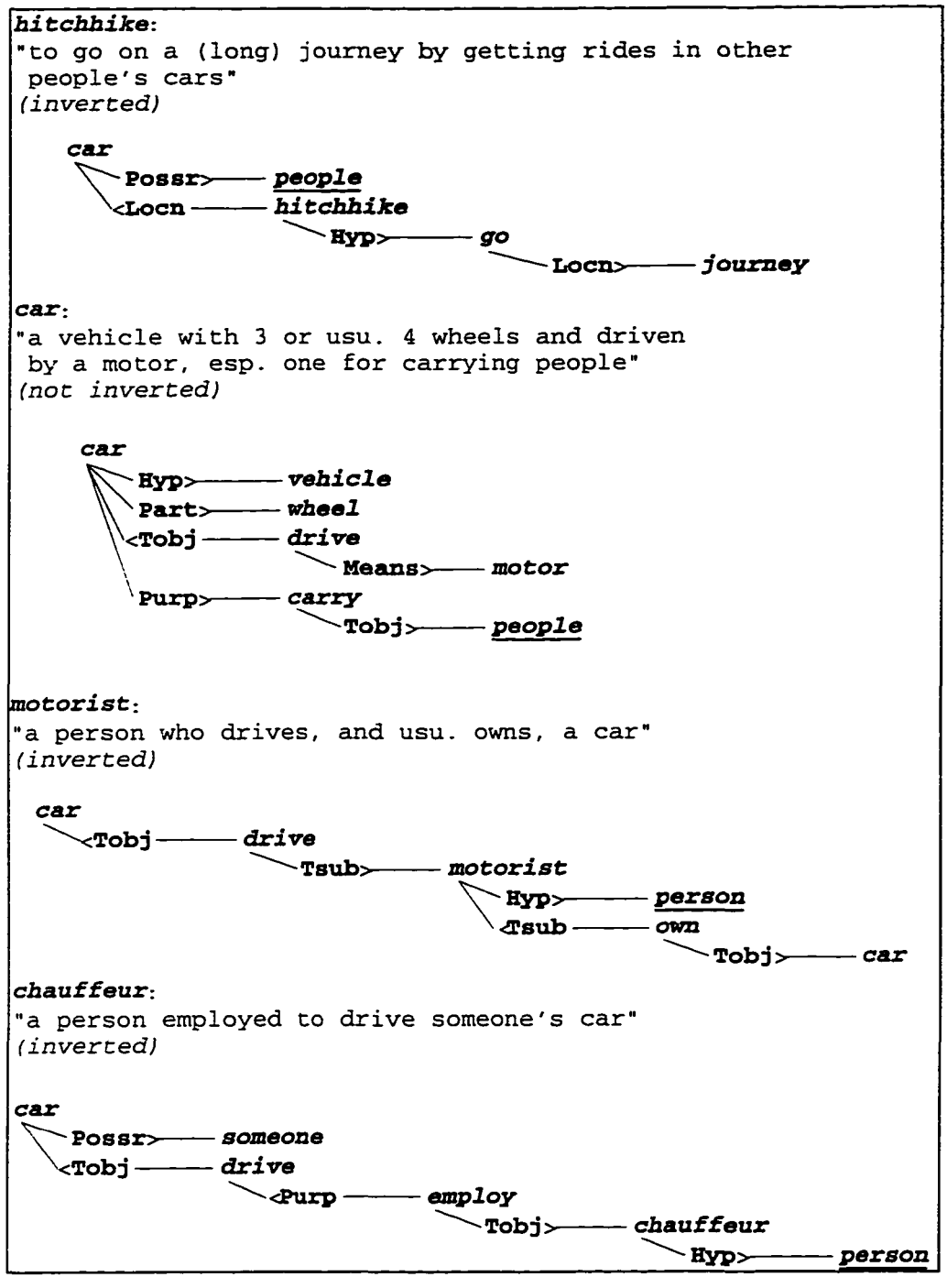

Figure 3.1. Some inverted semrel structures associated with **car** *that also contain* **people** *or* **person**

The semrel paths between ***car*** and ***people***, or ***person***, from the inverted semrel structures in Figure 3.1 are given in Figure 3.2 below. The paths may be as simple as a single semrel or they may consist of multiple semrels, as shown in Figure 3.2. The

directionality of the individual semrels in a semrel path does not imply a directionality for the entire path. The path is simply a bi-directional linkage between two words, and the directionality of an individual semrel only pertains to the meaning of the relation type of that semrel. Hence, one may read the third semrel path in Figure 3.2 starting at either end of the path: ***a car is driven by a motorist, who is a person***; or ***a type of person, a motorist, drives cars***. Both readings are paraphrases of one another and both indicate the same essential relationship between ***car*** and ***person***. This also means that the semrel paths between ***car*** and ***person*** in the inverted semrel structures stored with ***car*** are identical to those in the inverted semrel structures stored with ***person***, except that the semrels constituting the paths are stored in reverse order.

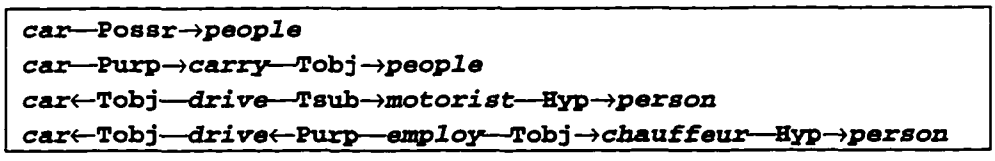

Figure 3.2. Semrel paths from **car** *to* **people** *or to* **person**, *taken from the inverted semrel structures in Figure 3.1*

Semrel paths, as described thus far, are taken only from single semrel structures produced from individual definitions. This is certainly true of the paths in Figure 3.2. Semrel paths could also be generated by linking individual semrels taken from multiple semrel structures, but this creates a number of potential problems that then must be dealt with. A combinatorial explosion of semrel paths would result, and methods would be needed for constraining them, not only to identify the most accurate and relevant ones, but also to achieve a reasonable degree of processing efficiency in generating and searching them.

In particular, paths derived from single definitions are linked through the same instance of a given word, thus insuring that the same sense of the word applies in each of the semrels it participates in. This is not the case if semrels are taken from the semrel structures of different definitions--the part-of-speech of the word may even be different--and there would need to be some way of insuring sense or part-of-speech matches, or at least compatibility. Also, given the possibility of inaccuracy for any semrel structure, as semrels from more structures are involved in the formation of a path, the likelihood that the path is correct diminishes. For these reasons, this work limits to two the number of semrel structures that can be used in the formation of a path, as well as constrains their use as described in a later section of this chapter, and does not allow the construction of semrel paths indiscriminately by piecing together individual semrels derived from several different semrel structures.

3.1 Semrel weights

In querying the information contained in the LKB, whether to obtain semrels containing a word and a particular relation type (in effect, a subset of the semrel paths of length 1 starting from the word) or to obtain all the semrel paths between two words, it is highly desirable to rank the results of a query based on potential usefulness. This is true not only for the needs of similarity determination and matching for lexical and structural disambiguation, but also for the sake of efficient processing. To this end, a method for assigning weights to semrel paths, based on weights assigned to individual semrels in the paths, has been developed. These weights are meant to be indicators of the usefulness of

the information contained in the semrel paths, and as such, they attempt to measure both the accuracy and the relevance of that information.

The use of automatic, rather than manual, methods to create the LKB in this work makes a weighting scheme particularly important. In certain cases, because of semrel inversion, there may be dozens, hundreds, or even thousands of semrel path connections between specific words. While the first goal of this work was to create a very richly connected LKB in order to achieve as wide a coverage of the language as possible, the increased numbers of connections for more common words actually began to interfere with efficient and useful processing. This is especially true in the context of LDOCE definitions, which exploit a limited defining vocabulary and therefore relate the words in that vocabulary to the headwords of the dictionary over and over again.

The ideal situation would be to have an even distribution of semrels across all of the words in the LKB, where each word participated in a sufficient number of semantic relations to provide an adequate semantic context for that word, but not so many that the information contained therein became overwhelming, irrelevant, and a hindrance to efficient processing. An ideal distribution of information in an LKB might be achieved if the semantic information were hand-coded, as it has been in the Cyc project (Guha and Lenat, 1990). However, hand-coding has its own problems, not the least of which is the cost of data entry by humans, resulting in a knowledge acquisition bottleneck. In contrast, the automatic approach in this work acquires information directly from natural language text (i.e., the dictionary definitions and example sentences), which by its very nature employs a broad spectrum of word frequencies, and a correspondingly wide range

of semrel frequencies. The management of this skewed distribution of information, particularly in cases where there is an over-abundance for certain high-frequency words, is an important purpose for assigning of weights to semrels and semrel paths. Another purpose is to provide a natural means for obtaining the most salient paths between words, which are required for determining similarity as described in the next chapter.

By way of comparison, other dictionary-based, knowledge-based, and example-based research has exploited weighting strategies to deal with identifying relevant information and ranking similarity. Braden-Harder (1993) used an adaptation of term weighting developed by Salton and McGill (1983) to weight the relevance of various pieces of information found in LDOCE entries for use in word sense disambiguation. Wilks et al. (1992) experimented with a number of statistical *relatedness* functions, based on co-occurrence of words in LDOCE definitions, to weight the strength of (unlabelled) relationships between words. Similar to Wilks, Bookman (1994) employed statistical methods, principally a modification of the averaged mutual information measure, to assign weights to the links in the relational tier of his knowledge base. Sumita and Iida (1991) relied on measurements of distance in a thesaurus hierarchy to weight the relational closeness of words during example-based matching, and Sussna (1993) used weights on relations in the hierarchy in WordNet to determine similarity in the context of information retrieval. Methods similar to those of Braden-Harder and Bookman were experimented with and examined closely as candidates for the weighting scheme used in this work before arriving at the preferred method. They are described in detail in the following section.

3.2 Preliminary weighting methods

In the discussion of weighting methods that follows, a semrel, informally a relation between two words, is represented more formally as a triple, $w_1\mathbf{R}w_2$, where w_1 and w_2 are words and **R** is the relation type of the relation between them. The computations for weighting semrels described below also require the consideration of the individual parts of semrels. The parts of a semrel $w_1\mathbf{R}w_2$, are $w_1\mathbf{R}$ (the first semrel-part) and $\mathbf{R}w_2$ (the second semrel-part). These parts are referred to hereafter as *semrel-parts*.

With semrels, as well as semrel-parts, it is important to note that the relation type **R** includes a direction, either to the right or to the left. For semrel-parts, this means that a second semrel-part, $\mathbf{R}w_2$, is equivalent to a corresponding first semrel-part with the relation type reversed, and vice-versa. For example, the second semrel-part **Purp**→*carry* in the semrel *car*—**Purp**→*carry* is equivalent to the first semrel-part *carry*←**Purp** in the equivalent semrel *carry*←**Purp**—*car*. Therefore, the form of a particular semrel in the computations below is not important, as long as the directionality of the relation type remains the same relative to the words in the semrel.

A more formal representation of a semrel path (e.g., the second one given in Figure 3.2) consisting of two semrels $w_1\mathbf{R}_1w_2$ (*car*—**Purp**→*carry*) and $w_2\mathbf{R}_2w_3$ (*carry*—**Tobj**→*people*), joined at word w_2 (*carry*), is given by the tuple $w_1\mathbf{R}_1w_2\mathbf{R}_2w_3$ (*car*—**Purp**→*carry*—**Tobj**→*people*). Generalized for semrel paths consisting of n semrels, this representation becomes $w_1\mathbf{R}_1w_2 \ldots w_n\mathbf{R}_{n+1}w_{n+1}$.

3.2.1 Term frequency and inverse document frequency

In the field of information retrieval (IR), where methods are studied for retrieving whole-text documents from expansive data bases using queries composed of relevant words contained in those documents, work by Salton (Salton and McGill 1983) is highly regarded and widely cited. His retrieval methods are based on measuring the similarity between queries and documents as represented by the similarity of vectors of term weights that represent those queries and documents. Terms are words or short phrases that are contained in the corresponding document (or query), and weights are assigned to these terms that are meant to indicate their salience relative to the content of the document (or query). An analogy is made between vectors of term weights and vectors of coordinates representing an angle in n-dimensional space. The similarity of two vectors of term weights may then be determined by comparing the cosines of the angles represented by each vector.

Salton determined that *term frequency* (tf_{ik} in Equation 3.1 below), or the frequency with which a given term occurs in a document, is a simple yet effective measure of its relevance to that document. Furthermore, he combined term frequency with another measure of relevance developed by Spark-Jones (1972), known as *inverse document frequency* ($\log_2(N/n_k)$ in the equation), to obtain the widely accepted term weighting formula given in Equation 3.1. Inverse document frequency is computed as the log of a ratio that represents the inverse of the pervasiveness of a term in the documents of a data base. It is based on the notion that the more common a term is throughout an entire data base of documents, the less likely it is to be highly relevant (and therefore a good discriminator) to any one particular document in the data base.

$$w_{ik} = tf_{ik} \cdot \log_2\left(\frac{N}{n_k}\right)$$

Equation 3.1. Term weight based on term frequency and inverse document frequency, adapted from Salton and McGill (1983) and Salton et al. (1994)

In the formula, as Salton et al. (1994) explained, "w_{ik} represents the weight of term T_k assigned to document D_i, tf_{ik} is the frequency of occurrence of term T_k in D_i, N is the size of the document collection, and n_k represents the number of documents in the collection with T_k.." A denominator (not shown) may also be used on the right-hand side of this formula to normalize the weight relative to the length of the document, so that long and short documents are retrieved with equal likelihood. Since the application of this formula to semrel weighting as described below does not deal with actual document retrieval, using the denominator was found to be unnecessary in the context of this work. Because the formula is composed of the product of the term frequency (*tf*) and the inverse document frequency (*idf*), the resulting weight is often referred to as a *tf*idf* weight.

An adaptation of Equation 3.1 was used by Braden-Harder (1993) to provide correlated weights for different sources of information in LDOCE, including syntactic subcategorization features, inherent semantic features, typical verb argument relations, and subject domain codes. Assigning weights to each of these items allowed Braden-Harder to combine their respective influences into a single semantic context and, using Salton's vector cosine similarity measurement, to compare it to the context of words in text to determine the correct senses of those words. Wilks et al. (1992) also used the same similarity measurement (but a different weighting scheme) to perform lexical disambiguation.

In this research, Equation 3.1 was the first formula used to assign weights to individual semrels. In order to understand how this was done, it is necessary to explain the correspondence between the terminology of information retrieval (terms, documents, etc.) and that of the LKB (semrels, headwords, etc.). The LKB itself may be likened to a data base of documents, with each headword entry in the LKB being the equivalent of an individual document. As each document contains a number of terms, with each term possibly occurring multiple times, so each headword entry contains a number of semrels, with each semrel occurring one or more times. In particular, the uniquely identifying part of each semrel in a headword entry is the second semrel-part, since the initial word of the semrel is the same in each case (i.e., the headword). In applying the term weight formula to this work, therefore, the frequency of each term in the document corresponds to the frequency of each second semrel-part (which is the same as the frequency of each semrel containing that part) in the headword entry. In order to compute the inverse document frequency part of the formula, the number of headword entries in the LKB is used instead of the number of documents in the data base, and the number of headword entries that contain a particular second semrel-part replaces the number of documents that contain a particular term. This translation allows the appropriate frequencies from the LKB to be inserted directly into Equation 3.1 in order to obtain a weight for each semrel. The factors for determining the weight of a semrel using this scheme, then, are the number of times it occurs in a headword entry and the number of headword entries in which its second part occurs throughout the LKB.

As an example of the assignment of a *tf*idf* weight to a semrel, Equation 3.2 shows the weight computation for the semrel ***car*—Part→*engine***. Note that the frequency of this semrel (or equivalently, the frequency of its second part) in the headword entry for ***car*** is 3, the number of headword entries in the entire LKB is 23230, and the number of headword entries in which the second semrel-part **Part→*engine*** occurs is 8.

$$w_{car—\text{Part}\rightarrow engine} = tf_{car—\text{Part}\rightarrow engine} \cdot \log_2\left(\frac{N}{n_{\text{Part}\rightarrow engine}}\right)$$
$$= 3 \cdot \log_2\left(\frac{23230}{8}\right)$$
$$= 34.511$$

*Equation 3.2. tf*idf weight of the semrel* **car—*Part*→engine**

Table 3.2 lists all of the semrels in the headword entry for ***car***[1] that contain the **Part** relation type, sorted according to their *tf*idf* weight. The purpose of listing the semrels in this table is to see if indeed the *tf*idf* weights perform the desired function of ranking the semrels according to their relevance. Relevance in this context refers to the strength of the cognitive association that is represented by the semrel. As an experiment to verify the relevance ranking given in Table 3.2, one might ask several people "what are the parts of a car?", record their answers, rank those answers by frequency of occurrence, and compare the ranking to that given in the table. Intuitively, a similar exercise may be performed in one's own mind, and the results compared against the table.

[1] Note that some of these semrels contain terms that are decidedly British, such as ***dicky*** and ***boot***, since LDOCE is, in fact, a learner's dictionary of British English.

Semantic relations	Semrel frequency (tf)	2nd Semrel-part headword frequency (idf denominator)	tf*idf weight
car*—Part→*engine	*3*	*8*	34.511
car*—Part→*wheel	*3*	*28*	29.089
car*—Part→*window	*2*	*4*	25.007
car*—Part→*top	*2*	*50*	17.720
car*—Part→*clutch	*1*	*1*	14.504
car*—Part→*dicky	*1*	*1*	14.504
car*—Part→*fender	*1*	*1*	14.504
car*—Part→*scoop	*1*	*1*	14.504
car*—Part→*underside	*1*	*1*	14.504
car*—Part→*boot	*1*	*3*	12.919
car*—Part→*cockpit	*1*	*3*	12.919
car*—Part→*trunk	*1*	*4*	12.504
car*—Part→*horn	*1*	*11*	11.044
car*—Part→*body	*1*	*39*	9.218
car*—Part→*front	*1*	*42*	9.111

*Table 3.2. **Part** relations for* **car**, *sorted by tf*idf weight*

Adopting this approach, it may be argued that the *tf*idf* weights do a reasonable job of ranking the semrels. In particular, the most frequently occurring semrels do seem to be the most relevant (with the possible exception of ***car*—Part→*top***), and this is consistent with Salton's assertion about the value of term frequency as a measure of relevance in IR. However, for those semrels that occur only once, the relevance distinctions resulting from the use of inverse document frequency seem somewhat counter-intuitive. Specifically, semrels that contain ***clutch***, ***dicky***, and ***scoop*** are ranked above those that contain ***trunk***, ***horn***, and ***body***. It is more likely, however, that people would mention the relations in the latter group more often and consider them more relevant than the uncommon ones in the former group. It could be argued that this anomaly is the result of having only single occurrences of these semrels, and that there is therefore inadequate information to

evaluate their relevance at all. In many of the cases, however, there are adequate numbers of individual semrel-parts, and if the frequencies of these parts are indeed significant in some way, which is a claim of this work, then it is more likely that this weakness is due to a difference in the meaning of relevance in an IR context versus an LKB context.

The goal of IR is to retrieve exactly (all and only) those documents that are relevant to a particular query. To this end, the methods employed must distinguish between documents that are and are not relevant. The use of inverse document frequency in the term weighting formula insures that terms which are unique to a particular document, i.e., those which are most distinctive for the purpose of retrieval, are assigned higher weights (representing greater relevance). Such unique terms may include proper nouns and words which are generally rare in the language. While this is a reasonable strategy in an IR context, it is not compatible with the "strength of cognitive association" definition of relevance desired in an LKB context. Infrequent terms, while possibly very distinctive for cars (e.g., ***glove compartment***), would not usually appear in a list of the most relevant car parts.

In an LKB generated from an online dictionary, there will be many unique and possibly uncommon words participating in correspondingly infrequent semrels; this results from the requirement for dictionaries to have adequate coverage of the language. Also, because of the automated process of semrel extraction, some unique or otherwise low frequency semrels are simply incorrect, being the result of bad parses for definitions or incorrect semrel pattern applications. These characteristics provide further justification

for excluding uniqueness and distinctive power from the definition of relevance in the LKB context of this work.

Given the contextual differences of the term *relevance*, the term *memorability* will be used hereafter to denote the type of relevance that is desired in this research. Semrels (and the semrel-parts of which they consist) that are more memorable will generally, but not always, have higher frequencies, consistent with the notion of a strong cognitive association. Important additional characteristics of memorability will be discussed in a later section of this chapter.

One final problem with *tf*idf* weights worth noting is that they are not symmetrical. For example, the weight for the semrel ***car*—Part→*engine*** is 34.511, while the weight of the equivalent semrel ***engine*←Part—*car*** is 31.79. This difference is due to the inverse document frequency computation, which in the first case is based on the number of headword entries that contain the semrel-part **—Part→*engine*** (8), and in the second case on the number of entries that contain the semrel-part **←Part—*car*** (15). Different weights for equivalent semrels become especially problematic when they result in different weights for equivalent semrel paths; these paths cannot then be ranked consistently with other semrel paths that result from querying the LKB.

At first it may seem that some weight difference based on the perspective of the relation is desirable, since certain relations may seem to have more strength (i.e., be more memorable) from the perspective of one word than from the other. However, the

difference that one senses may be explained in terms of *relative* versus *absolute* relational strength.

Consider as an example the equivalent semrels ***eat*—Means→*spoon*** and ***spoon*←Means—*eat***. The reason that the relational strength of the first semrel may feel a bit weaker than that of the second semrel is that ***eat*** has so many other semrels that are strongly associated with it, in particular, those involving typical objects of ***eat*** (***food***, specific kinds of foods, ***lunch***, etc.). ***Spoon***, on the other hand, is associated with many fewer semrels (some of which link it to other kinds of utensils), and its relation to ***eat*** is placed fairly high in the list. Relatively speaking, then, ***eat*—Means→*spoon*** is not as memorable among the semrels associated with ***eat*** as ***spoon*←Means—*eat*** is among the semrels associated with ***spoon***. This does not mean, however, that these equivalent semrels should not have the same absolute weight, and in fact, having the same weight allows them to be ranked consistently in relation to other semrel paths as stated above.

One possible modification to *tf*idf* weights that would make them symmetrical would be to introduce two *idf* computations into the formula--one for each semrel part--but then some other adjustment would have to be made to keep the influence of the *idf*s from overshadowing that of the *tf*. Interestingly, in the weighting schemes discussed below, there are components that correspond functionally to *idf*s for both semrel-parts and hence, those schemes do produce symmetrical semrel weights.

3.2.2 Mutual information

In an attempt to overcome some of the deficiencies of *tf*idf* weights, methods based on mutual information measurements were also considered for the weighting of semrels. Mutual information is a measurement of the associative strength between a pair of events, for example, a pair of words occurring in the same sentence in a corpus. As shown in Equation 3.3, mutual information (*MI*) is defined as the ratio of the probability of the occurrence of events *x* and *y* together over the product of their individual probabilities, with the product representing their expected occurrence together based on an assumption of independence. In the case of word pairs in a corpus, the probabilities are estimated by the observed frequencies with which the word pairs and individual words appear. The base-2 logarithmic function in the equation transforms the ratio into the number of bits required to represent the amount of information conveyed by the occurrence of the events together.

$$MI(x, y) = \log_2\left(\frac{P(x, y)}{P(x)P(y)}\right)$$

Equation 3.3. Mutual information

Mutual information calculations have been used by a number of NLP researchers, including Church and Hanks (1989), who created an *association ratio* representing the strength of the association between words in a fashion similar to that described in the above paragraph, and Magerman and Marcus (1990), who determined phrasal boundaries during syntactic parsing by identifying *distituent* words (i.e., contiguous words that are not part of the same phrase) on the basis of their negative mutual information scores.

Alshawi and Carter (1994) have experimented with mutual information scores specifically for the purpose of weighting semantic triples that are similar to the semrels employed in this work. In their research, mutual information was shown to be a useful measure for determining the associative strength of these triples. They also showed that χ^2 scores and *mean distance* scores were even better determiners, but the computation of both of these scores was based indirectly (for χ^2) and directly (for mean distance), on training scores derived from a hand-tagged corpus of parse structures (the Penn Treebank). The use of these two types of scores, therefore, cannot be considered as a viable weighting strategy in the context of this work, which depends on the fully automatic processing of an untagged corpus (LDOCE definitions).

Equation 3.4 below shows the application of the mutual information computation in Equation 3.3 to the semrel ***car*—Part→*engine*** in order to determine an associative-strength weight for that semrel. The occurrences of the semrel-parts ***car*—Part** and **Part→*engine*** are the two events having their mutual information measured. The probability of the two semrel-parts occurring together constitutes the numerator of the ratio, and is estimated by the frequency of the semrel ***car*—Part→*engine***, 3, over the total number of semrels in the LKB, 137526. In the denominator is the product of the probabilities of the individual semrel parts. The frequency of the first semrel-part ***car*—Part**, 21, is placed over the total number of semrels (137526) to compute the probability of the first semrel-part. Since the **Part** relation occurs in both semrel-parts, the probability of **Part→*engine*** is conditioned by the occurrence of **Part** in ***car*—Part**,

and therefore its frequency, 11, is placed over the number of semrels in the LKB containing the **Part** relation, 12365.

$$MI(car\text{—}\mathrm{Part} \rightarrow engine) = \log_2\left(\frac{P(car\text{—}\mathrm{Part} \rightarrow engine)}{P(car\text{—}\mathrm{Part})P(\mathrm{Part} \rightarrow engine|\mathrm{Part})}\right)$$
$$= \log_2\left(\frac{\frac{3}{137526}}{\left(\frac{21}{137526}\right)\left(\frac{11}{12365}\right)}\right)$$
$$= 7.327188$$

Equation 3.4. Mutual information weight of the semrel **car—*Part*→engine**

The mutual information weights for all of the semrels in the headword entry for ***car*** that contain the **Part** relation are listed in Table 3.3. This table may be compared to Table 3.2 to see if the mutual information weights actually provide a better ranking of the semrels according to their memorability. Unfortunately, the ranking is worse, as the mutual information weights prefer many semrels that occur only once over those that occur multiple times. In so doing, they appear to rank the semrels roughly in ascending order of the frequency of their second semrel-part (the frequency of the first semrel-part, ***car*—Part**, is a constant 21 in each case), being influenced much less by the frequency of the entire semrel in each case.

Semantic relations	Semrel frequency	2nd Semrel-part frequency	Mutual information
car—Part→clutch	1	1	9.201657
car—Part→dicky	1	1	9.201657
car—Part→fender	1	1	9.201657
car—Part→scoop	1	1	9.201657
car—Part→underside	1	1	9.201657
car—Part→window	2	5	7.879729
car—Part→boot	1	3	7.616694
car—Part→cockpit	1	3	7.616694
car—Part→engine	3	11	7.327188
car—Part→trunk	1	6	6.616694
car—Part→wheel	3	32	5.786619
car—Part→horn	1	11	5.742225
car—Part→top	2	61	4.270919
car—Part→body	1	43	3.775392
car—Part→front	1	48	3.616694

*Table 3.3. **Part** relations for* **car**, *sorted by mutual information weight*

The only positive comment to be made about simple mutual information weights is that they are symmetrical. In this sense, the probability of each semrel-part in the denominator of the calculation plays a role similar to the *idf* in the *tf*idf* weights, penalizing the overall score as the probability (in step with the frequency) of the semrel-part increases. As suggested at the end of the last section, having two *idf*-like computations (the semrel-part probabilities) provides symmetry in the weighting of equivalent semrels, but it also creates a problem by overshadowing the influence of the frequency of the entire semrel. While mutual information weights are therefore superior in their symmetry, they suffer significantly in the diminished influence of the frequencies of entire semrels.

The poor ranking exhibited by the mutual information weights is most likely the result of the relatively small frequencies involved in the computations. Church and

Hanks (1989) chose to ignore word pairs with frequencies less than 5 in their calculations, stating that the mutual information scores in such cases are unreliable as predictors of associative strength. Dunning (1993) also pointed out that many statistical measures are based on assumptions of normal distributions, whereas natural language texts are full of *rare events*, i.e., infrequently occurring words, and this is especially true in LDOCE dictionary text, which defines many of the uncommon words in the English language.

In an attempt to rectify this problem, Bookman (1994) employed variations on the basic mutual information computation to produce meaningful weights for the unlabeled links between concepts in the relational tier of LeMICON, a structured knowledge base similar to the LKB of this work. The first of these was his use of *average* mutual information, also known as the *expected value* of the mutual information function.

The computation of average mutual information ($\bar{I}$) is defined in Equation 3.5 below. It sums the mutual information scores for each of four possible events: x and y occur together, x occurs but y does not, y occurs but x does not, and neither x nor y occur. Within the summation, each of these scores is multiplied by the probability that the particular event will occur, thus resulting in an average score across all four possible events.

$$\bar{I}(x,y) = \sum_{x,y \in \{0,1\}} P(x,y) MI(x,y)$$

Equation 3.5. Average mutual information

Equation 3.6 shows the application of the average mutual information computation to the semrel ***car*—Part→*engine***. The first term in the summation is identical to the basic mutual information computation for this semrel in Equation 3.4, multiplied by the probability of the entire semrel. The second term represents the mutual information score of semrels that contain **Part→*engine***, but do not contain ***car*—Part**, multiplied by the overall probability of such semrels. The third term represents the score of semrels that contain ***car*—Part**, but do not contain **Part→*engine***, again multiplied by the corresponding probability. The last term represents the score of semrels that contain neither ***car*—Part** nor **Part→*engine***, multiplied by the corresponding probability. Note that the initial probabilities in each term total to 1, thus covering all possible combinations of the semrel-parts that comprise this semrel. Also, for many semrels, in actually computing the probabilities required for average mutual information, it is necessary to use the accepted smoothing technique of adding .5 to the frequency counts (not shown below) in order to have placeholders in the case of sparse data, thereby avoiding division by zero.

$$\overline{I}(car\text{—Part} \rightarrow engine) = \sum_{x,y \in \{0,1\}} P(car\text{—Part} \rightarrow engine)\, MI(car\text{—Part} \rightarrow engine)$$

$$= \left(\frac{3}{137526}\right)\log_2\left(\frac{\frac{3}{137526}}{\left(\frac{21}{137526}\right)\left(\frac{11}{12365}\right)}\right) +$$

$$\left(\frac{8}{137526}\right)\log_2\left(\frac{\frac{8}{137526}}{\left(\frac{12344}{137526}\right)\left(\frac{11}{12365}\right)}\right) +$$

$$\left(\frac{18}{137526}\right)\log_2\left(\frac{\frac{18}{137526}}{\left(\frac{21}{137526}\right)\left(\frac{12354}{12365}\right)}\right) +$$

$$\left(\frac{137497}{137526}\right)\log_2\left(\frac{\frac{137497}{137526}}{\left(\frac{12344}{137526}\right)\left(\frac{12354}{12365}\right)}\right)$$

$$= 3.478173$$

Equation 3.6. Average mutual information weight of the semrel **car—*Part*→engine**

The average mutual information weights for all of the semrels in the headword entry for ***car*** that contain the **Part** relation are listed in Table 3.4. This table may be compared with both Table 3.2 and Table 3.3 to evaluate the desirability of the average mutual information weights. A significant shift for the better has occurred with these weights in that the semrels that occur only once are now at the bottom of the list and the higher frequency semrels are placed in the top half of the ranking. However, as with the basic mutual information weights, there is still too much overshadowing of the semrel frequencies by the (descending) frequency of the second semrel-part (the first semrel-part frequency is still constant here).

Semantic relations	Semrel frequency	2nd Semrel-part frequency	Average mutual information
car—Part→*top*	2	61	3.482114
car—Part→*front*	1	48	3.481005
car—Part→*body*	1	43	3.480600
car—Part→*wheel*	3	32	3.479839
car—Part→*engine*	3	11	3.478173
car—Part→*horn*	1	11	3.478019
car—Part→*window*	2	5	3.477626
car—Part→*trunk*	1	6	3.477620
car—Part→*boot*	1	3	3.477386
car—Part→*cockpit*	1	3	3.477386
car—Part→*clutch*	1	1	3.477241
car—Part→*dicky*	1	1	3.477241
car—Part→*fender*	1	1	3.477241
car—Part→*scoop*	1	1	3.477241
car—Part→*underside*	1	1	3.477241

*Table 3.4. **Part** relations for* **car**, *sorted by average mutual information weight*

Bookman (1994) further modified the average mutual information computation by removing the event in which neither x nor y occur and renormalizing the probabilities, resulting in what he called the average *conditional* mutual information. He did this to factor the rare events (those which generally do not occur together) out of the computation, claiming that it would therefore perform better when the frequencies were small. Such a change to the computation of the average mutual information weights in Table 3.4 results in some minor shifts in the upper half of the ranking. These shifts are good in that they penalize the semrels whose semrel-parts have the very highest frequency (the goodness of this action will be explained in the following section), but the overshadowing of the frequencies of the semrels by those of their semrel-parts still remains as a significant problem.

In conclusion, while weights based on average mutual information are symmetrical and give a lower ranking to semrels that only occur once, they do not consistently reward semrels with higher overall frequencies like *tf*idf* weights do. What is needed is a weighting scheme that combines the best features of both the average mutual information weights and the *tf*idf* weights, more closely approximating the notion of memorability as it has been defined and as it will yet be expanded.

3.3 Averaged vertex probability

One of the pioneering efforts in the use of statistical methods to analyze natural language texts was by Zipf (1949), who observed that when the frequencies of the unique word forms appearing in James Joyce's *Ulysses*—260,000 instances (tokens) of 30,000 unique word forms (types)—were plotted against the numerical rank of those word forms from most to least frequent, the distribution formed a hyperbolic curve (or a straight, sloped line, if graphed logarithmically). Another perspective of this distribution was that for any word form in the text, its observed frequency multiplied by its rank produced a near constant value. He observed this same distribution in other texts, and even in other phenomena related to human sociology (e.g., the populations of cities), and he attributed it to the workings of the forces of *unification* and *diversification*. That is, words which rank highly and occur frequently are the result of the unifying forces of language, seeking to combine the meanings and usages of words in an effort to economize the references to commonly occurring concepts. At the other end of the spectrum, diversifying forces tend to create unique words for concepts of low frequency, thus simplifying the reference to

those concepts while not harming the overall economy of the language, precisely because of the lower frequencies.

The hyperbolic frequency/rank distribution of NL texts was exploited by Luhn (1958) in early efforts related to information retrieval. He observed that words that were of neither high nor low frequency had the greatest *resolving power*, i.e. that the middle frequency words were the best discriminators of a particular text or document. Words that occur with high frequency are generally very ambiguous or are function words, in either case not conveying meaning that could be characterized as precise for a particular document. Low frequency words, on the other hand, do not occur enough to denote adequate importance in a document. The dotted curve in Figure 3.3 is meant to represent the resolving power of words, plotted against Zipf's hyperbolic frequency/rank distribution. Luhn suggested that high and low frequency cutoffs could be established, as shown in the figure, to eliminate less discriminating words from consideration in the context of document retrieval.

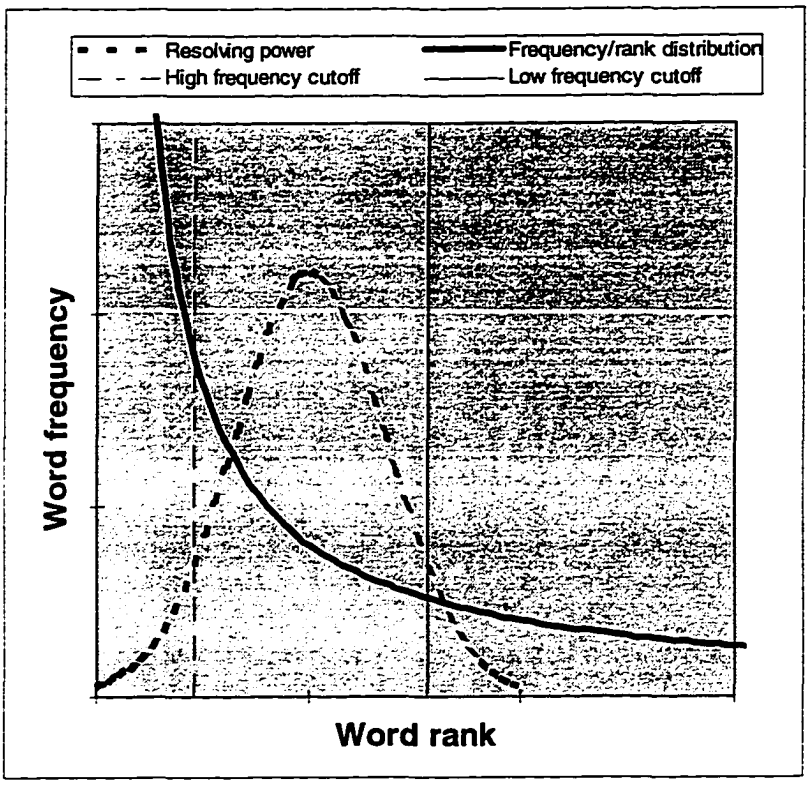

Figure 3.3. Word frequency/rank distribution and corresponding "resolving power," taken from Luhn (1958)

Salton and McGill (1983) picked up on the work by Luhn, but dismissed the notion of absolute cutoffs in favor of a weighting scheme that balanced the frequency of a word or term within a document against its frequency throughout a collection of documents; this is the *tf*idf* weighting scheme examined earlier in this chapter. Salton and McGill viewed this balance of frequencies as providing higher resolving power for middle-frequency words. High frequency words in a document would most likely occur very frequently throughout the document collection and thereby be penalized by the *idf* portion of the weight. Low frequency words would be penalized by the *tf* portion. However, in somewhat of a departure from the pure "middle frequency" approach of Luhn, the *idf* factor of *tf*idf* weights also rewards low frequency words in a document that are unique to that document, since they are good discriminators for the document as well. This is

why the *tf*idf* weights in Table 3.2 ranked semrels with semrel-parts occurring only once higher than those with semrel-parts occurring multiple times. It is this orientation toward discrimination in IR that counters the notion of memorability discussed in this research.

For a semrel to be memorable, it must have a strong cognitive association between its parts, and there must be a balance between frequency and uniqueness. It must be frequent enough to be salient, yet not so frequent that it looses that salience; it must also be distinctive, yet not so distinctive that it rarely occurs. This is precisely the balance of Zipf's forces of unification and diversification, and it is consistent with Luhn's original notion of resolving power. Inspecting Figure 3.3 again, one may observe that the peak of the resolving power curve is located approximately over the vertex of the hyperbolic frequency/rank distribution curve. Mathematically, the vertex of the latter curve is exactly the balancing point between its extremities. This thesis proposes that the vertex of this curve represents the highest degree of memorability for those elements represented by its distribution.

There is an adjustment that must be made, however, when applying this definition of memorability to the distribution of semrels. Because semrels are derived from LDOCE dictionary definitions, they include references to a large portion of the words in the language, including many uncommon ones. The frequency/rank distribution of semrels is therefore skewed such that there are many more unique semrels (113396 out of 137526 total semrels), and the overall frequency of semrels is lower than in a natural distribution (90 is the highest semrel frequency). The skew is also exacerbated by the lack of function words, which are typically the highest frequency words in text, in the "language" of

semrels. This skewed distribution does not exhibit the kind of curve observed by Zipf for real text. Instead, the curve is truncated along the frequency axis, and its vertex has a frequency that is too high to be representative of the most memorable semrels.

The skew problem may be overcome by changing the curve from a frequency/rank distribution to a *frequency-count*/frequency distribution, where the frequency-count is the number of unique semrels that have a particular frequency. So, instead of plotting the frequency of each unique semrel according to its rank, the number of unique semrels with a particular frequency is plotted against that frequency for each semrel frequency in the distribution. Both the rank/frequency (axes reversed) and frequency-count/frequency distributions for all the semrels in the LKB are shown in the graph in Figure 3.4. The reversal of the X and Y axes for the rank/frequency and frequency-count/frequency distributions is only for the purpose of graph readability and has no impact on the frequency value at the vertex of the curves.

The frequency-count/frequency distribution is not as skewed as the rank/frequency distribution for the same data, having an overall shape which is closer to the hyperbolic distribution observed by Zipf. This is due to the fact that separate data points (representing ranked, unique semrels) in the rank/frequency distribution are compressed into a single data point (representing a class of unique semrels having the same frequency) in the frequency-count/frequency distribution. Grishman and Sterling (1994), in their analysis of triples (like semrels) extracted from text corpora, also plotted frequency-counts against the frequencies of triples (similar to semrels) in order to normalize an otherwise skewed distribution, bringing it closer to a Zipf-like curve.

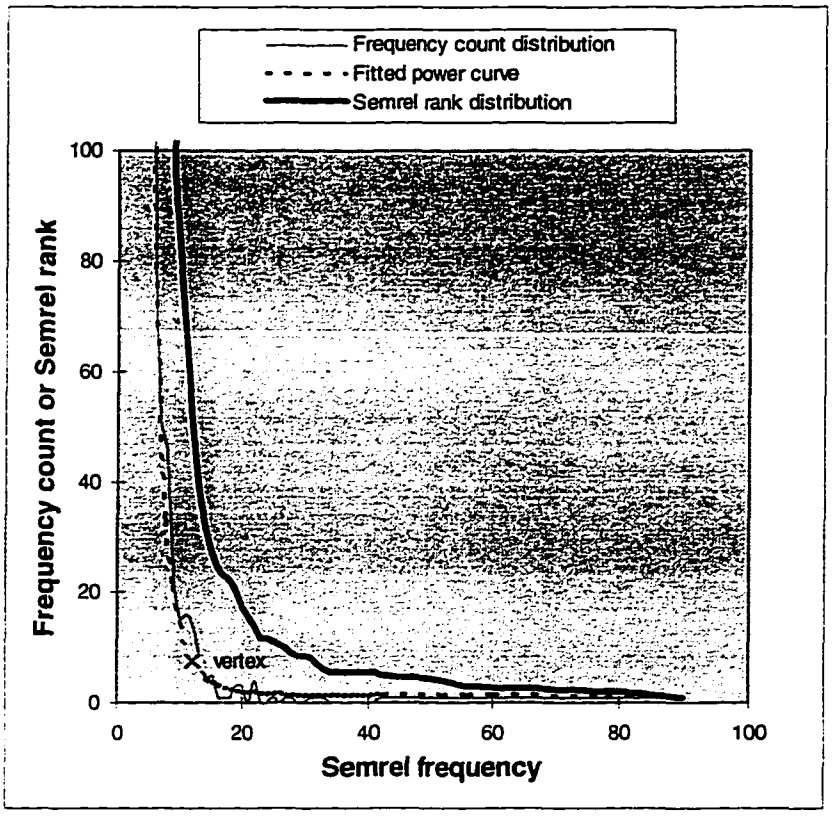

Figure 3.4. Distributions of frequency count and semrel rank over semrel frequency for all semrels

Although the use of frequency-count values produces a more steeply sloped, hyperbolic curve, there is also more variation in these values than in rank values, creating a need for some statistical smoothing of the curve. To this end, traditional curve fitting techniques are used whereby a power curve, defined in Equation 3.7 below, is fitted to the frequency-count/frequency distribution by minimizing the values of a squared-error cost function. The resulting fitted curve is also shown in Figure 3.4, bisecting the points of variation in the frequency-count curve (it may be difficult to distinguish, as it overlaps, especially at its extremities, with the frequency-count curve). It is the vertex of this fitted curve, marked by an X in Figure 3.4, that is actually used for determining semrel memorability in the computations that follow. The memorability of semrels is computed to increase as their frequency approaches the frequency of this vertex from either end of

the frequency scale, and the least memorable semrels have frequencies that are at the extremes of that scale (either very low or very high frequencies).

$$\boxed{Y = 1 + aX^{b}}$$

Equation 3.7. Power curve fitted to frequency-count distributions

The usefulness of individual semrel-part frequencies was discussed previously in the sections on *tf*idf* weights and average mutual information weights. In the weighting scheme proposed in this section, they also play an important role, and it is therefore necessary to consider their relative memorability as well. Figure 3.5 shows the frequency-count distribution (with fitted curve) of all the semrel-parts contained in the semrels in the LKB, as compared to their rank distribution. In this case, the truncation of the latter curve is even more severe than with the whole semrels in Figure 3.4, and the frequency of its vertex is much too high to represent the most memorable semrel-parts.

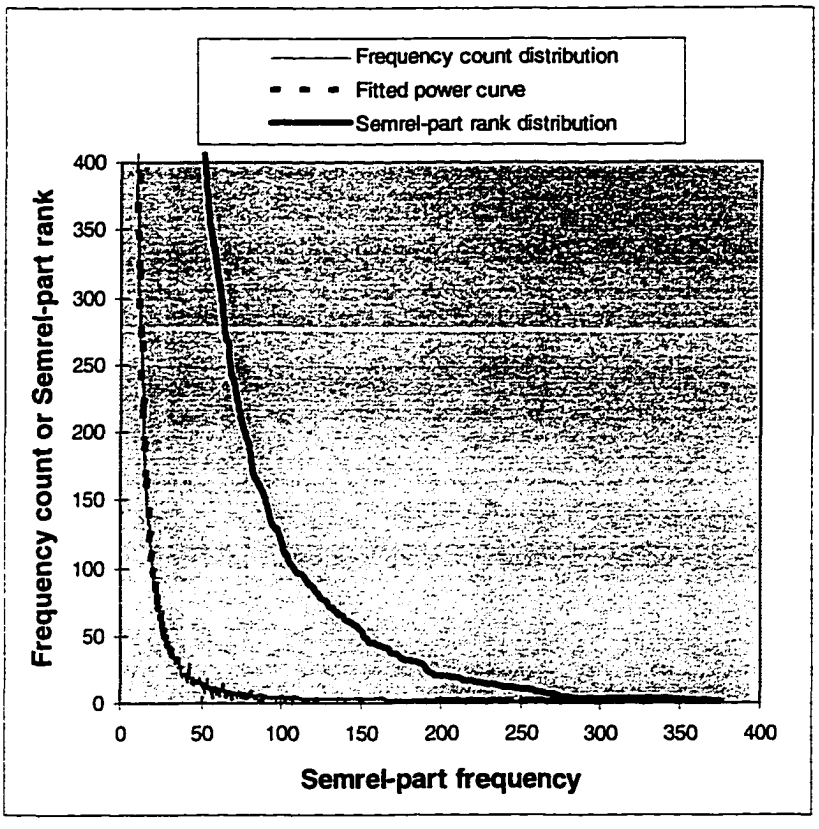

Figure 3.5. Distributions of frequency count and semrel-part rank over semrel-part frequency for all semrel-parts

Because the frequencies of semrels and the range of their distribution (the corresponding frequency-counts) can vary dramatically by relation type, separate distributions of semrels are computed for each relation type. This ensures that the highest frequency semrels for each relation type are assigned a lower memorability, but it does not eliminate the effect that many semrels of one relation type may be more memorable than semrels of another relation type due to overall higher frequencies. For example, Figure 3.6 shows the frequency-count distributions for semrels containing the **Tobj** relation and the **Purp** relation. The frequency of the vertex of the **Tobj** distribution is 11, while that of the **Purp** distribution is 4. Also, the highest semrel frequency in the **Purp** distribution is 11; the same as the frequency of the vertex of the **Tobj** distribution. The

frequency of the vertex of the distribution for all semrels, regardless of relation type, is around 12, therefore, if a separate **Purp** distribution were not computed, the **Purp** semrel and **Tobj** semrels of frequency 11 would be considered equally memorable, and this is simply not true. Examples of **Tobj** and **Purp** semrels whose frequencies are equal or near to the frequency of the vertex of the distribution for their respective relation type are: ***eat***—**Tobj**→***food***, ***draw***—**Tobj**→***line***, ***catch***—**Tobj**→***fish***, ***examine***—**Purp**→***find***, and ***shop***—**Purp**→***buy***.

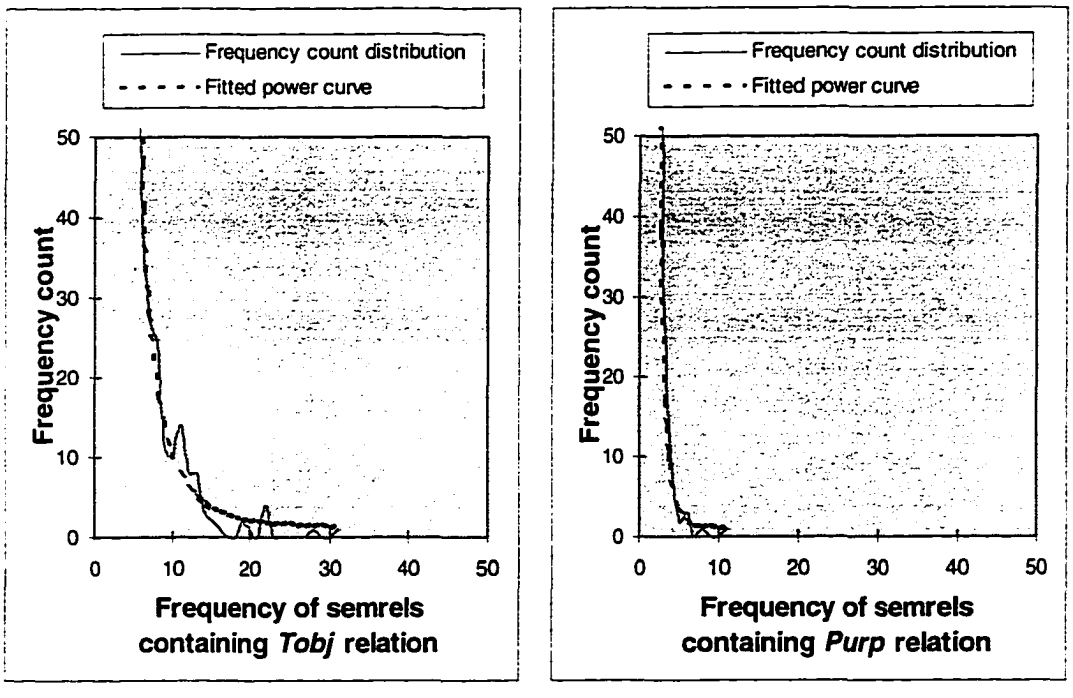

*Figure 3.6. Distribution of frequency count over semrel frequency for semrels containing the **Tobj** and **Purp** relations*

Using the notion of memorability discussed above, the *vertex frequency* (F_V) of a semrel $w_1\mathbf{R}_1w_2$ can now be formally defined in Equation 3.8. It is the frequency of the semrel, limited by the frequency of the vertex of the distribution of semrels having the same relation type. If the frequency of the semrel (the first argument to the min function in the equation) exceeds the frequency of the vertex, it is scaled back toward 1 (the lowest

frequency) according to the value of the fitted curve function for that distribution (the second argument to the min function). This means that as the actual semrel frequencies increase across the distribution for a particular relation type, the vertex frequencies of those semrels approach, possibly equal, then recede from the frequency of the vertex of the distribution.

$$F_V(w_1 R_1 w_2) = \min\left(F(w_1 R_1 w_2), \left(1 + a_{R_1} F(w_1 R_1 w_2)^{b_{R_1}}\right)\right)$$

Equation 3.8. Vertex frequency of a semrel

The distribution of vertex frequencies for semrels containing the **Tobj** relation is shown graphically in Figure 3.7, together with the fitted power curve for the **Tobj** relation from Figure 3.6. For frequencies higher than that of the vertex, the vertex frequency overlaps exactly with the fitted power curve, as one might expect based on the use of the min function in Equation 3.8.

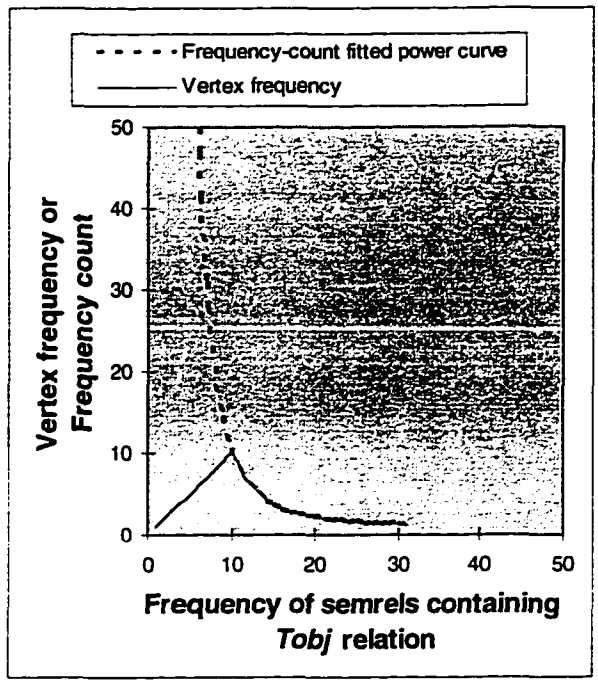

*Figure 3.7 Vertex frequency and frequency count distribution for semrels containing the **Tobj** relation*

Vertex frequencies may also be computed for semrel-parts, using the same equation and substituting the semrel-part frequency and the fitted curve function for the distribution of semrel-parts having the corresponding relation type. Because of generally smaller, but in some cases significant, differences, semrel-part distributions are further subdivided on the basis of the relation type directionality. Figure 3.8 shows the distributions for semrel-parts containing the **Part** relation, referred to as **Part** or **PartOf** depending on the directionality of the relation. Although these distributions are similar, there are still differences in the frequency of the vertex (approximately 13 for **Part** and 12 for **PartOf**) and the slope of the curve that result in different F_V values for the same semrel-part frequency ($F_V(21)$ is 6.21 for **Part** and 4.58 for **PartOf**).

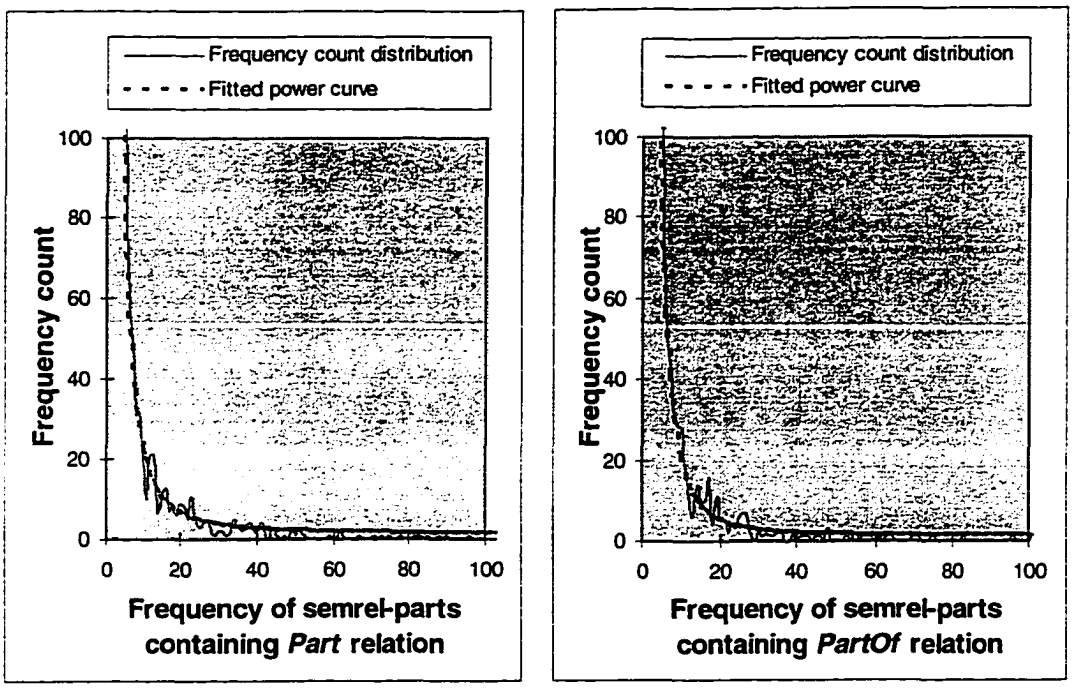

Figure 3.8. Distribution of frequency count over semrel-part frequency for all semrel-parts containing the ***Part*** *and* ***PartOf*** *relations*

The *vertex probability* (P_V) of a semrel $w_1\mathbf{R}_1w_2$ is shown in Equation 3.9. It is the essential computation for determining memorability, and it consists simply of the vertex frequency of the semrel divided by the frequency of all the semrels in the LKB. Vertex probabilities may be multiplied together, just like normal probabilities, but they are different in that they have as an upper bound the probability corresponding to the frequency of the vertex of the corresponding distribution.

$$P_V(w_1R_1w_2) = \frac{F_V(w_1R_1w_2)}{\sum_{\text{all } i,j,k} F(w_iR_jw_k)}$$

Equation 3.9. Vertex probability of a semrel

The vertex probabilities of semrel-parts $w_1\mathbf{R}_1$ and $\mathbf{R}_1w_2$ are given in Equation 3.10. Of note is that the vertex probability of the second semrel-part is conditioned by the existence of the relation type occurring in the first semrel-part to which it is joined. As

shown, the denominator of the vertex probability for the second semrel-part is therefore the number of semrel-parts that have the relation type $\mathbf{R}_1$, rather than the total number of semrel-parts in semrels in the LKB.

$$P_V(w_1R_1) = \frac{F_V(w_1R_1)}{\sum_{\text{all } i,j} F(w_iR_j)}$$

$$P_V(R_1w_2|R_1) = \frac{F_V(R_1w_2)}{\sum_{\text{all } k} F(R_1w_k)}$$

Equation 3.10. Vertex probability of first and second semrel-parts

With vertex probability as a model for the memorability of both semrels and their semrel-parts, the question remaining is how to combine the vertex probabilities of semrels and their parts in a way that appropriately balances the effect of each on the resulting weight and at the same time maintains symmetry. *tf*idf* weights were well balanced in this regard, though not symmetrical, and average mutual information weights, though symmetrical, were influenced too much by the frequency of the semrel-parts.

A common technique in working with small sample statistics is the use of a weighted averaging factor. Such a factor increases the influence of statistics as the number of observations (the observed semrel frequency in this case) from which they are derived increases. An averaging factor has been found to be particularly useful in balancing the effect of the vertex probabilities of semrels with the vertex probabilities of their semrel-parts, depending on the semrel frequencies. A simple averaging factor (A) based on the frequency of a semrel $w_1\mathbf{R}_1w_2$ is shown in Equation 3.11, together with its complementary factor (A'). As the semrel frequency increases, the value of A increases

toward 1 and the value of A' correspondingly decreases toward 0, the total of the two factors always remaining a constant 1.

$$A(w_1 R_1 w_2) = \frac{F(w_1 R_1 w_2)}{F(w_1 R_1 w_2)+1}$$
$$A'(w_1 R_1 w_2) = \frac{1}{F(w_1 R_1 w_2)+1}$$

Equation 3.11. Averaging factor and its complementary factor, based on semrel frequency

Using this averaging factor, it is now possible to define the *averaged vertex probability* ($\overline{P_V}$) of a semrel $w_1\mathbf{R}_1w_2$, shown in Equation 3.12. This computation combines the vertex probability of the semrel, multiplied by the averaging factor, with the product of the vertex probabilities of the semrel-parts, multiplied by the complementary averaging factor. The effect of multiplying by the averaging factor and its complement is that as the frequency of the semrel increases, the influence of the semrel's vertex probability increases over that of the product of the semrel-parts' vertex probabilities. The resulting value is symmetrical, like the average mutual information weights, and gives proper emphasis to the semrel frequency as it increases, like the *tf*idf* weights.

$$\overline{P_V}(w_1 R_1 w_2) = A(w_1 R_1 w_2) P_V(w_1 R_1 w_2) + A'(w_1 R_1 w_2) P_V(w_1 R_1) P_V(R_1 w_2 | R_1)$$

Equation 3.12. Averaged vertex probability of a semrel

An example computation of averaged vertex probability is given in Equation 3.13 for the semrel ***car*—Part→*engine***. Since the observed frequency of the semrel is 3, the averaging factor weights the vertex probability of the semrel by 3/4 and the vertex probability of the semrel-parts by 1/4. Note that since the semrel-part frequency of ***car*—Part**, 21, is greater that the frequency of the vertex of its distribution, 13, its vertex

frequency is scaled back to 6.213. This results from the use of Equation 3.8 to compute the vertex frequency, with $a_{Part} = 2419$ and $b_{Part} = -2.01675$.

$$\begin{aligned}\overline{P_V}(car\text{—Part} \rightarrow engine) &= A(car\text{—Part} \rightarrow engine)P_V(car\text{—Part} \rightarrow engine) + \\ &\quad A'(car\text{—Part} \rightarrow engine) \\ &\quad P_V(car\text{—Part})P_V(\text{Part} \rightarrow engine|\text{Part}) \\ &= \left(\frac{3}{3+1}\right)\left(\frac{3}{137526}\right) + \left(\frac{1}{3+1}\right)\left(\frac{6.213}{137526}\right)\left(\frac{11}{12365}\right) \\ &= .0000163706\end{aligned}$$

Equation 3.13. Averaged vertex probability of the semrel **car—*Part*→engine**

By applying the $\overline{P_V}$ computation to all of the semrels in the headword entry for ***car*** that contain the **Part** relation, the results listed in Table 3.5 are produced. This table may be compared with the previous tables containing *tf*idf* weights (Table 3.2) and average mutual information weights (Table 3.4) in order to evaluate the success of $\overline{P_V}$ in ranking the semrels according to their memorability.

Semantic relations	Semrel frequency	2nd Semrel-part frequency	Averaged vertex probability
car—Part→engine	3	11	.0000163706
car—Part→wheel	3	32	.0000163627
car—Part→window	2	5	.0000097012
car—Part→top	2	61	.0000096967
car—Part→horn	1	11	.0000036557
car—Part→trunk	1	6	.0000036466
car—Part→boot	1	3	.0000036411
car—Part→cockpit	1	3	.0000036411
car—Part→body	1	43	.0000036387
car—Part→front	1	48	.0000036384
car—Part→clutch	1	1	.0000036375
car—Part→dicky	1	1	.0000036375
car—Part→fender	1	1	.0000036375
car—Part→scoop	1	1	.0000036375
car—Part→underside	1	1	.0000036375

Table 3.5. ***Part*** *relations for* **car**, *sorted by averaged vertex probability weight*

In the table, the $\overline{P_v}$ weights have ranked semrels with higher frequency at the top of the list, just as the *tf*idf* weights did. However, they have also ranked the semrels containing ***horn*** and ***trunk*** (two fairly memorable parts of a car) at the top of those semrels that occur only once. The semrels containing ***body*** and ***front***, having semrel-parts of higher frequency (beyond the frequency of the vertex of their distributions), are moved to the middle of the single occurrence group, representing the observation that many different objects have a ***front*** or a ***body***, and therefore these are less memorable in specific relation to a car. Finally, the semrels whose parts only occur once remain at the bottom of the list. Certainly a ***scoop*** and a ***dicky*** belong here, but it would be nicer if ***fender*** could be moved up in the list. Such sparse data problems will remain, however, to be addressed by the addition to the LKB of semrels from another dictionary's definitions, as described in Chapter 4, and eventually from yet other sources.

In conclusion, weights based on averaged vertex probability overcome the identified weaknesses of *tf*idf* and average mutual information weights in that they are symmetrical and they provide rankings that reasonably model memorability, being well-adapted for the pervasiveness of small semrel frequencies in the LKB. In Chapter 4, their utility for measuring similarity will be evaluated.

3.4 Semrel path weights

Since $\overline{P_V}$ weights are probabilities, albeit modified ones, they may be multiplied together in order to represent the weights of multi-semrel paths. Assigning weights to paths is very important for the similarity determination discussed in the next chapter. It also allows the paths to be generated more efficiently as queries are made of the information contained in the LKB.

When combining the $\overline{P_V}$'s of semrels to obtain a semrel path weight, however, some minor adjustments must be made. Specifically, when considering two semrels $w_1\mathbf{R}_1w_2$ and $w_2\mathbf{R}_2w_3$ that together form the path $w_1\mathbf{R}_1w_2\,\mathbf{R}_2w_3$, the $\overline{P_V}$ of the second semrel must be conditioned for the existence of its initial word w_2 in the first semrel. In the $\overline{P_V}$ computation, this means conditioning both the P_V of the semrel $w_2\,\mathbf{R}_2w_3$ as well as the P_V of the first semrel-part $w_2\,\mathbf{R}_2$, as shown in Equation 3.14 below. In a manner similar to that used in Equation 3.10 for conditioning second semrel-parts, this entails making the denominator of the P_V of the semrel the number of semrels that have the same initial word w_2, and likewise for the first semrel-part.

$$\overline{P_V}(w_2R_2w_3|w_2) = A(w_2R_2w_3)P_V(w_2R_2w_3|w_2) + A'(w_2R_2w_3)P_V(w_2R_2|w_2)P_V(R_2w_3|R_2)$$

Equation 3.14. Averaged vertex probability of a semrel given the initial word

With the conditional $\overline{P_V}$ computation for semrels in place, the $\overline{P_V}$'s of the semrels that make up a path can be multiplied together to obtain a $\overline{P_V}$ weight for the entire path. The $\overline{P_V}$ computation for a semrel path of length *n* (the number of semrels in the path) is shown in Equation 3.15.

$$\overline{P_V}(w_1R_1w_2R_2w_3...w_nR_nw_{n+1}) = \overline{P_V}(w_1R_1w_2)\overline{P_V}(w_2R_2w_3|w_2)...\overline{P_V}(w_nR_nw_{n+1}|w_n)$$

Equation 3.15. Averaged vertex probability of a semrel path of length 2 or more (up to n*)*

Initially, it was thought that the frequency of an entire path that occurs explicitly in the LKB could be used to compute the P_V for that path. This P_V might then be averaged together with the multiplied $\overline{P_V}$'s from the individual semrels in the path, in a similar fashion to the computation for the individual $\overline{P_V}$'s themselves, in order to obtain a $\overline{P_V}$ for the entire path. However, most semrel paths, especially those consisting of more than a couple of semrels, turned out to be unique in the LKB, and therefore, there was no advantage in incorporating P_V's for entire paths into the computation for the $\overline{P_V}$'s of those paths.

Equation 3.16 provides an example of the computation for the $\overline{P_V}$ weight of the semrel path ***boat*—Purp→*travel*—Locn→*water***. Note the use of the conditioned $\overline{P_V}$ weight for the second semrel in the path, ***travel*—Locn→*water***.

$$\overline{P_V}(boat\text{—}\mathrm{Purp}\rightarrow travel) = A(boat\text{—}\mathrm{Purp}\rightarrow travel)$$
$$P_V(boat\text{—}\mathrm{Purp}\rightarrow travel)+$$
$$A'(boat\text{—}\mathrm{Purp}\rightarrow travel)$$
$$P_V(boat\text{—}\mathrm{Purp})P_V(\mathrm{Purp}\rightarrow travel|\mathrm{Purp})$$
$$=\left(\frac{1}{1+1}\right)\left(\frac{1}{137526}\right)+\left(\frac{1}{1+1}\right)\left(\frac{3}{137526}\right)\left(\frac{4.587}{6902}\right)$$
$$=.0000036429$$
$$\overline{P_V}(travel\text{—}\mathrm{Locn}\rightarrow water|travel) = A(travel\text{—}\mathrm{Locn}\rightarrow water)$$
$$P_V(travel\text{—}\mathrm{Locn}\rightarrow water|travel)+$$
$$A'(travel\text{—}\mathrm{Locn}\rightarrow water)$$
$$P_V(travel\text{—}\mathrm{Locn}|\mathrm{travel})$$
$$P_V(\mathrm{Locn}\rightarrow water|\mathrm{Locn})$$
$$=\left(\frac{2}{2+1}\right)\left(\frac{2}{184}\right)+\left(\frac{1}{2+1}\right)\left(\frac{3.282}{184}\right)\left(\frac{1.26}{11258}\right)$$
$$=.007247042$$
$$\overline{P_V}(boat\text{—}\mathrm{Purp}\rightarrow travel\text{—}\mathrm{Locn}\rightarrow water) = \overline{P_V}(boat\text{—}\mathrm{Purp}\rightarrow travel)$$
$$\overline{P_V}(travel\text{—}\mathrm{Locn}\rightarrow water|travel)$$
$$=(.0000036429)(.007247042)$$
$$=2.640025e\text{-}08$$

Equation 3.16. Averaged vertex probability of the semrel path **boat—*Purp*→travel—*Locn*→water**

The semrel path ***boat*—Purp→*travel*—Locn→*water*** occurs naturally in the LKB, i.e., it exists in an inverted semrel structure that was derived from a definition (one of the definitions of ***boat***, to be exact) through the semrel extraction process. But what about paths that are created from the semrels in more than one inverted semrel structure? The opening section of this chapter mentioned that semrels from two separate structures would be allowed in the creation of paths. This includes joining two individual semrels or two paths of semrels (which are sub-paths in this case) at a common word (or *join word*). Limiting to two the number of source semrel structures involved avoids the

combinatorial explosion that would result if paths were constructed piecemeal from the semrels of many different semrel structures.

An *extended semrel path* is a path created from sub-paths in two different inverted semrel structures. For example, ***car*** and ***truck*** are not related by a semrel or semrel path from any single semrel structure. However, if one allows the joining of the semrels in the first two columns of Figure 3.9, each from a different semrel structure, at the word ***vehicle***, the semrel paths in the third column result. Note once again that the directionality of the semrels that comprise a path does not produce a directionality for the entire path; the path is a bi-directional linkage between two words. Extended semrel paths exhibit a tradeoff between length and accuracy in that paths consisting of several semrels can be generated (where the sub-paths themselves consist of multiple semrels), and yet there is only one junction point where a higher possibility of inaccuracy exists (usually because of a potential word sense mismatch). As long as they are constrained adequately, extended semrel paths have proven invaluable in determining the relationship between certain words in the LKB that would not otherwise be connected.

1st semrel	2nd semrel	Extended semrel path
car—Hyp→vehicle	vehicle←Hyp—truck	car—Hyp→vehicle←Hyp—truck
car—Purp→carry	carry←Purp—truck	car—Purp→carry←Purp—truck

Figure 3.9. Creation of extended semrel paths

Equation 3.17 gives the $\overline{P_v}$ computation for an extended semrel path of length n, joining two sub-paths of length j and n-j at join word w_{j+1}. Except for the middle term of the equation, it is a natural extension of Equation 3.15, in that the $\overline{P_v}$ of the first sub-path

(consisting of the multiplied $\overline{P_V}$'s of the individual semrels in that sub-path) is multiplied with the conditioned $\overline{P_V}$ of the second sub-path.

$$\overline{P_V}\left(w_1 R_1 w_2 \ldots w_j R_j w_{j+1} \ldots w_n R_n w_{n+1}\right) = \overline{P_V}\left(w_1 R_1 w_2 \ldots w_j R_j w_{j+1}\right) \; P_V\left(w_{j+1} J w_{j+1} \middle| w_{j+1}\right) \; \overline{P_V}\left(w_{j+1} R_{j+1} w_{j+2} \ldots w_n R_n w_{n+1} \middle| w_{j+1}\right)$$

Equation 3.17. Averaged vertex probability of an extended semrel path of length n, joined at word $\mathbf{w}_{j+1}$

The middle term of Equation 3.17 represents the increased uncertainty introduced by joining the two sub-paths at join word w_{j+1}. Its purpose is simply to penalize the path weight of an extended path. The term is the vertex probability (P_V) of a pseudo-semrel consisting of one instance of w_{j+1} connected by the special **Join** (**J**) relation to another instance of w_{j+1}, conditioned by the first instance of w_{j+1}. The intuition underlying the pseudo-semrel is that the two instances of w_{j+1} come from different semrel structures, and they may therefore have different word senses, hence requiring a relation of some kind between them. The vertex frequency assigned to this semrel is the minimum (1), and the conditioning causes the denominator of the P_V to be the number of semrels that have the same initial word w_{j+1}. The P_V is not averaged in this cause because there are no frequencies available (or imaginable) for its semrel parts.

Equation 3.18 shows an example of the $\overline{P_V}$ computation for the weight of the extended path ***boat*—Purp→*travel*—Locn→*water*** (ignoring, for purposes of comparison, that it really does occur as a non-extended path in the LKB). Note that the $\overline{P_V}$ of the extended version of this path is lower than that of its original, non-extended

version (computed in Equation 3.16), being penalized by the P_V of the pseudo-semrel at the join point. This difference rightfully denotes the decreased confidence in the accuracy of the extended path. The $\overline{P_V}$'s of the two individual semrels comprising the path were computed previously in Equation 3.16 and are simply used again in Equation 3.18. The P_V of the pseudo-semrel is computed by the minimum frequency (1) over the number of semrels that have ***travel*** as their initial word (184).

$$\begin{aligned}\overline{P_V}(boat\text{—}\mathrm{Purp}\rightarrow\underline{travel}\text{—}\mathrm{Locn}\rightarrow water) &= \overline{P_V}(boat\text{—}\mathrm{Purp}\rightarrow travel)\\ &\quad P_V(travel\text{—}\mathrm{Join}\rightarrow travel|travel)\\ &\quad \overline{P_V}(travel\text{—}\mathrm{Locn}\rightarrow water|travel)\\ &= (.0000036429)\left(\frac{1}{184}\right)(.007247042)\\ &= 1.434796e\text{-}10\end{aligned}$$

Equation 3.18. Averaged vertex probability of the extended semrel path **boat—*Purp*→<u>travel</u>—*Locn*→water**

The $\overline{P_V}$ computations in Equation 3.15 and Equation 3.17 are clearly compositional in nature, producing higher semrel path weights for paths consisting of more memorable semrels. They also produce weights that exhibit a number of other reasonable ranking characteristics. The semrel paths in Table 3.6 give examples of some of these characteristics, as explained below:

- The first path in Table 3.6, which is non-extended, is ranked above the other extended paths in the table (and ranks above its own extended version, as discussed above). Extended paths usually rank below non-extended ones because of the penalty imposed by including the **Join** pseudo-semrel in the computation of extended path weights.

- The first and second paths in the table have a length of 2 while the third one has a length of 3. Shorter paths generally rank higher than longer paths (by virtue of multiplying fewer or more probabilities, respectively), although the rank may still vary depending on the $\overline{P_V}$ (the memorability) of the individual semrels comprising the path.

- The fourth path in the table, although shorter than the third path, is ranked after it because of the relatively high frequency of semrels beginning with the join word ***carry*** (and corresponding high frequency of ***carry*** throughout the LKB). Because the frequency of the join word of an extended path is a conditioning factor on the $\overline{P_V}$ of the second sub-path, as well as on the P_V of the **Join** pseudo-semrel, high frequency join words have a negative effect on the $\overline{P_V}$ of the entire path. This models the generally higher ambiguity of high frequency open class join words (semrels do not contain closed class words such as prepositions, conjunctions, and articles), which results in greater uncertainty about the correct matching of the word senses of the join word present in each sub-path.

Semrel paths	Averaged vertex probability
boat—Purp→travel—Locn→water	1.4622e-009
boat←Hyp—raft—Tobj→water	3.9355e-010
boat—Part→keel←Hyp—centreboard—Locn→water	2.5235e-011
boat←Locn—carry—Tobj→water	6.1944e-012

Table 3.6. Some semrel paths between **boat** *and* **water**, *ranked by averaged vertex probability*

One of the benefits of assigning weights to semrel paths mentioned at the beginning of this chapter was that of efficient path querying. For non-extended paths between two words, this is not really an issue, since they may be retrieved simply by looking up the first of the words and traversing the inverted semrel structures stored with it, searching for occurrences of the second word. For extended paths, however, the use of weights is crucial in eliminating the need to create all of the (sometimes hundreds or even thousands) possible semrel paths between words in order to identify the highest ranking ones.

The algorithm that retrieves the semrel paths between a *source word* and *target word* is also responsible for creating the extended paths between the words. This algorithm places retrieved paths in a ranked *output list*, and it consists of the steps outlined below:

1. Look up both the source and target words and traverse the inverted semrel structures stored with them, flagging the words that the structures from each word have in common (i.e., the *intersection words*).

2. Traverse the inverted semrel structures of the source word again, and create and save copies of the paths from the source word to each intersection word.

3. During the traversals in step 2, if an intersection word is also the target word, place the copy of the (non-extended) path from the source word to that intersection word in the output list, which has a predetermined maximum length *n* and is kept sorted in descending order by semrel path weight.

4. Traverse the inverted semrel structures of the target word again, and for each intersection word, compute the weight of the semrel path that would be formed by joining each of the source structure sub-paths saved in step 2 for that intersection word with the sub-path from the target word to that intersection word.

5. If the weight computed in step 4 is higher than the weight of the last (lowest weighted) path in the output list, or if there are fewer than *n* paths in the output list, create an extended path from the source and target sub-paths and place it in the output list.

6. Each time that adding an extended path to the output list in step 5 causes there to be more than *n* paths in the output list, discard the last path in the output list.

Because the order in which extended paths are created by the algorithm is random with respect to their path weight, the creation of many lower ranking paths is avoided, thus improving the efficiency of the query. This is especially true when *n* is set to a small number (50 by default) relative to the number of possible paths between words (many hundreds or even thousands in some cases). One could make an additional improvement to the algorithm, if desired, to minimize the amount of storage wasted for discarded paths by implementing another traversal of the target word semrel structures before step 4 that computes and saves the top-ranked path weights. These saved weights could then be compared with the weights of potential paths before they are actually created in steps 4 and 5.

As extended paths are created during step 5 in the algorithm, there is an additional set of constraints that must be met before they are placed in the output list. Together with the ranking effect of the weights, these constraints serve to further contain the explosion of possible paths, eliminating paths which are not likely to be useful or accurate.

The first constraint is fairly simple, checking for and disallowing the inclusion of duplicate paths in the output list. Duplicate paths may result from creating extended versions of non-extended paths by using sub-paths from the different (inverted) versions of the same semrel structure, which are stored with both the source and target word. Or, they may result from the legitimate repetition of (possibly quite memorable) semrels and semrel paths throughout the LKB.

The second constraint blocks the formation of paths that contain certain kinds of *cycles*. A cycle occurs when the same word is repeated in a semrel path. Figure 3.10 shows two paths containing cycles that are blocked by this constraint. In the first path between ***spoon*** and ***eat***, ***eat*** is directly related to ***spoon*** in the first semrel and is then related to itself through two semrels joined at ***food***. The second path between ***bird*** and ***nest*** consists of a first sub-path that relates ***bird*** to itself (from a definition for ***hawk***: "any of many types of bird, often large, which catch other birds ...") before a second sub-path relates it directly to ***nest***. In both of these cases, the direct relationship between source and target words is not enhanced by the additional relations in the cyclic paths.

```
spoon←Means—eat—Tobj→food—Purp→eat
bird←Hyp—hawk←Tsub—catch—Tobj→bird—Possr→nest
```

Figure 3.10. Some semrel paths blocked by the path cycle constraint

Some paths that are not blocked by this constraint are shown in Figure 3.11. The key difference between these paths and the ones in Figure 3.10 is that the repeated word is not the source or target word. The repeated word in these cases often relates the source and target words to another word that provides more specific information about that relationship between the source and target than if the repeated word itself were the join word (although it may currently still be the join word in other paths). This is true of the join words ***information*** and ***egg*** in this example as they relate to ***handbook/literature*** and ***bird/nest***, respectively.

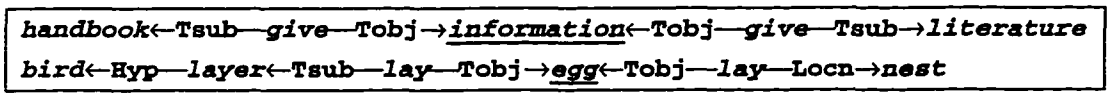

Figure 3.11. Some semrel paths not *blocked by the path cycle constraint*

The final constraint that has been implemented blocks the formation of paths with join words that are in a stop word list. Although frequently occurring join words, as discussed in conjunction with Table 3.6, have a negative effect on semrel path weights, some words are so common (and correspondingly ambiguous or lacking in relevant meaning) that they are best not considered at all, both as regards efficiency as well as accuracy. The stop word list currently includes some of the highest frequency words in the LKB, including ***thing***, ***make***, ***give***, ***use***, ***do***, ***put***, ***take***, and ***have***. However, it does not include some words that, although they also have very high frequencies, still participate in ostensibly meaningful relations, such as ***person***, ***place***, ***animal***, ***time***, ***move***, and ***work***.

Returning to Figure 3.11 for a moment, since ***give*** is in the stop word list, the version of the first path in this figure joined at ***give*** is blocked, leaving the version shown in the figure joined at ***information***. This makes intuitive sense, since saying that

handbooks and ***literature give*** without saying what they ***give*** is not very informative. However, the version of the second path joined at ***lay*** (not shown in the figure) is not blocked, since ***lay*** is not in the stop word list. This seems reasonable, since one can sensibly say that ***layers***, which are ***birds***, ***lay***, and that ***laying*** takes place in ***nests*** (***eggs***, of course, are strongly implied in this case).

There are two other constraints that deserve some mention at this point. The first, which would block paths when the part of speech of the join word in the two sub-paths does not match, has not yet been implemented. However, it is not clear whether this constraint in its simplest form is completely desirable. There are many cases where a word may have multiple parts of speech (e.g., ***raft***—a noun and a verb—in the second semrel path in Table 3.6) which are closely related semantically, and which would be compatible for the purpose of joining sub-paths into an extended path.

The second constraint, which has been implemented, involves matching the word senses of the join word in the two sub-paths. However, the assignment of word sense identifiers to the words in the semrel structures (and therefore, in the semrel paths) throughout the LKB is the result of work by Dolan (1994), and it is outside the scope of the research for this dissertation.

4. Determining similarity and inferring semantic relations

The previous chapters have described the creation of a richly connected lexical knowledge base (LKB) by extracting semantic relations (*semrels*) from the definitions of an online dictionary, organizing those semrels in *semrel structures*, inverting those semrel structures and propagating them across all the entries in the LKB, identifying connections, or *semrel paths*, contained in the (possibly combined) semrel structures, and assigning weights to the semrel paths for the purpose of ranking their salience or *memorability*. The semrel paths between words in the LKB provide explicit relational links, both simple and complex, that may be used effectively in NLP systems to perform lexical and structural disambiguation. Such is the case in the Microsoft NLP system, which uses these semrel paths heavily in its disambiguation processing.

But what of the tens of thousands or even hundreds of thousands of semrel paths which are valid, and necessary for disambiguation processing in a broad-coverage NLP system, but are not stored explicitly in the LKB? No matter how large an online dictionary or other corpus of information is used as a source for the LKB, there will always be gaps in the knowledge that is explicitly stored, even in more limited domains, and these gaps must be bridged by some sort of inferencing mechanism. Indeed, even humans are called upon to make inferences regularly in their processing of language for the same reason.

In large-scale NLP systems whether example-based, rule-based, or of any other orientation, information not explicitly present must therefore be inferred from that which

is present. Most inferencing is performed in these systems by finding an explicitly stored example of the desired relation or piece of information, and then establishing that the example is similar to the particular case at hand with a reasonable level of confidence. This is true in a wide variety of systems and methods, including those described by Sumita (1991), Sato (1991), Braden-Harder (1993), Bookman (1994), Grishman and Sterling (1994), and Resnik (1995). Work by these researchers and others is discussed in detail in Chapter 5.

For instance, in order to determine the correct attachment of the prepositional phrase ***with binoculars*** in sentence (1) below, an example-based system might search for the phrase ***observe with binoculars*** (as well as ***distance with binoculars*** and ***scene with binoculars***) in its data base of examples. If an explicit instance of this phrase were not found, then other phrases containing ***with*** (and possibly one of the other words in the phrase) would be sought, and the similarity between the content words in any examples found and the content words in the original phrase would be measured. If a particular example were identified as significantly similar to one of the phrases in the original text, the attachment would be performed accordingly.

(1) ***My friend observed the tragic scene from a distance with binoculars.***

In the Microsoft NLP system, the possible attachments of prepositional phrases are represented by specific semantic relations between the content words involved. In the above example, ***observe with binoculars*** is represented by the semrel ***observe***—**Means**→***binoculars***. (This abstraction allows a single semrel potentially to cover phrases employing a number of different prepositions; in this case **Means** may be

realized as ***by***, ***via***, ***through***, or ***with***.) If this semrel were not explicitly stored in the LKB, attempts would be made to find other semrels containing the **Means** relation between words which were similar to ***observe*** and ***binoculars***. In order to constrain the system and to avoid inferring relations that are not true, at least one of the two words in the semrel must still be an exact match instead of being inferred through similarity.

In order to determine similarity between words for the purpose of inferring relations, two general strategies have been employed. One is based on identifying direct, paradigmatic relations between the words, such as **Hypernym** (**Hyp**) or **Synonym** (**Syn**). The term *paradigmatic* refers to relations indicating that the words are members of the same conceptual paradigm, i.e., that they are taxonomically very close to one another. For example, they might be in a parent, child, or sibling relationship with each other in an **IsA** hierarchy. The other strategy is based on identifying syntagmatic relations with other words that similar words have in common. The term *syntagmatic* refers to relations such as **TypicalObject** (**Tobj**) or **TypicalSubject** (**Tsub**), which hold between words that might occur near to each other in a sentence. The fact that two words are frequently found to have the same syntagmatic relations with other words is often a strong indicator of similarity between the two words.

Formerly in the Microsoft system, similarity was determined by looking for simple, explicit, paradigmatic relations between words (such as **Hyp** or **Syn**) in the LKB, and in some cases for combinations of these. Sumita and Iida (1991) implemented a related method, determining similarity based on looking up words in an online thesaurus and finding the lowest common thesaurus classification between them, assuming a thesaurus

containing a hierarchically organized set of classifications in which words are placed. The shorter the path through a common classification between the words in the hierarchy, the more similar they were considered.

Syntagmatic strategies for determining similarity have often been based on statistical analyses of large corpora that yields clusters of words which occur in similar contexts (e.g., Brown et al. 1992, Pereira 1993, Yarowsky 1992). Most of the contexts in such analyses consist simply of nearby or even contiguous words. However, Hindle (1990), as well as Grishman and Sterling (1994), have made use of the predicate-argument structures generated by a broad-coverage parser, demonstrating that words which occur in the same argument position of a predicate (i.e., in the same syntagmatic relation with another word) often share similar characteristics. Sato (1991) utilized the same kind of predicate-argument basis for similarity, but only in a fairly impoverished example base of verb-object collocations rather than in a large corpus of text. Chapter 5 provides additional discussion of the work of these and other researchers on determining similarity.

The method for determining similarity employed in this work includes the best characteristics of the above methods in a single integrated procedure which is consistent with other processing in the LKB. It makes use of all of the relation types available in the LKB, both the paradigmatic ones as well as the syntagmatic ones, and in this regard brings to bear the power of both the thesaurus-based and the predicate-argument approaches. It also relies on the statistical analysis of a moderately-sized corpus (i.e., all the semrels contained in the LKB), as represented in its semrel weights and the frequency

with which certain patterns of semrels indicate a high degree of similarity. The experiments described in the following sections provide evidence that this method is quite accurate in determining similarity as well as useful in inferring relations not explicitly stored in the LKB.

4.1 Identifying similarity path patterns

In the context of the weighted, inverted semrel structures contained in the LKB, querying for the semrel paths between two words returns a ranked list of paths that indicates the most salient connections between the query words. For example, some of the top semrel paths between "pen" and "pencil", together with their semrel path weights, are shown in Table 4.1 below.

Semrel paths	Path weight
pen←Means—draw—Means→pencil	2.5235e-08
pen←Means—write—Means→pencil	8.3368e-09
pen←Means—drawing—Means→pencil	9.6161e-10
pen—Hyp→instrument←Hyp—pencil	7.6736e-12
pen—Hyp→write—Means→pencil	6.2517e-12
pen←Means—write←Hyp—pencil	5.7810e-12
pen←Means—draw←Hyp—pencil	5.5463e-12
pen—Hyp→write←Hyp—pencil	1.7100e-12

Table 4.1 High ranking semrel paths between **pen** *and* **pencil**

In the semrel paths in Table 4.1, a pattern of semrel symmetry clearly emerges in almost all of the paths. In fact, this same type of symmetry has been observed consistently in paths between words which are similar in various ways. This observation led to the hypothesis that similar words are typically connected in the LKB by semrel paths that frequently exhibit certain patterns of relations (exclusive of the words they actually connect), many patterns being symmetrical, but others not. Table 4.2 shows the

similarity path patterns for each of the semrel paths connecting ***pen*** and ***pencil*** in the example. Note that the directionality of each relation in the path pattern is represented by the presence or absence of an **Of** at the end of the relation name. This notation for semantic relations was introduced in Chapter 2, and was found to be simpler and clearer in the context of path patterns, where the intervening words are not included.

Semrel paths	Path patterns
pen←Means—*draw*—Means→*pencil*	MeansOf Means
pen←Means—*write*—Means→*pencil*	MeansOf Means
pen←Means—*drawing*—Means→*pencil*	MeansOf Means
pen—Hyp→*instrument*←Hyp—*pencil*	Hyp HypOf
pen—Hyp→*write*—Means→*pencil*	Hyp Means
pen←Means—*write*←Hyp—*pencil*	MeansOf HypOf
pen←Means—*draw*←Hyp—*pencil*	MeansOf HypOf
pen—Hyp→*write*←Hyp—*pencil*	Hyp HypOf

Table 4.2. Similarity path patterns

Note the lack of symmetry in the fifth, sixth, and seventh path patterns above. And yet, these are still patterns that legitimately indicate similarity. Such asymmetry often results when words such as pen and pencil, which are usually nouns but may also be used cross-categorically as verbs, are examined for similarity. In the fifth pattern, for example, the verb ***pen*** has ***write*** as a **Hypernym**, and the noun ***pencil*** is the **Means** of the verb ***write***. Sometimes, paths indicating similarity consist of semrels joined at a word which is considered a verb in one semrel and a noun in the other. At the end of Chapter 3, there was a discussion about how extended semrel paths with such a part of speech mismatch were not blocked by path constraints because of their intuitive semantic relatedness. In the context of similarity determination, there is now a more pragmatic reason to allow them.

In some cases, a path indicating similarity may be as simple as a single **Hyp** relation. These simple cases are then equivalent to the kind of processing employed previously in the Microsoft NLP system. But more often, symmetrical paths containing two or more semrels link similar words, and in some cases, there may be combinations of symmetrical sub-paths. Table 4.3 shows four examples of complex similarity path patterns. The first two are symmetrical, with the first exhibiting a type of nested symmetry and the second a type of sequential symmetry. The last two are partially symmetrical, but have an additional **Hyp** relation either at the beginning or end of the path.

Semrel path
Corresponding path pattern
affront—Hyp→*hurt*—Tobj→*feeling*←Tobj—*hurt*←Hyp—*offend*
Hyp Tobj TobjOf HypOf
bullet←Hyp—*ammunition*—Hyp→*bomb*←Hyp—*rocket*—Hyp→*missile*
HypOf Hyp HypOf Hyp
burn←Hyp—*flame*—Manner→*brightly*←Manner—*blaze*
Hyp Manner MannerOf
lance—Part→*pennon*←Part—*spear*←Hyp—*harpoon*
Part PartOf HypOf

Table 4.3. Complex similarity path patterns

Certainly, one way to identify similarity using the semrel paths between words would be to hand-code an algorithm that recognized any combination of sequential or nested symmetrical relations in paths. However, such an algorithm would lack the ability to quantify the degree of similarity indicated by a particular path pattern. It would also not necessarily recognize the legitimate asymmetrical path patterns in the examples above. The realization of these shortcomings led to the quantitative analysis of path pattern frequency between words known to be similar, thus obtaining both the path

patterns themselves as well as a measure of each pattern's relative strength in indicating similarity (or *similarity potential*).

In order to obtain the path patterns and their similarity potentials, up to 10 of the highest weighted semrel paths between pairs of words taken from an online thesaurus (used in Microsoft Word, from SoftArt, Inc.) were generated by querying the LKB. Over 33,000 noun and verb word pairs, which had one or more semrel paths connecting them in the LKB, were used, and over 180,000 semrel paths were obtained, an average of 5.5 paths per word pair. The resulting similarity path patterns were extracted and sorted, and the frequencies of 14,775 unique path patterns were obtained. This procedure is summarized in Table 4.4.

Training procedure #1	
Input	over 33,000 noun and verb thesaurus word pairs, connected by at least one semrel path in LKB
Process	Obtain up to 10 highest ranked semrel paths for each pair; extract and sort path patterns; count instances of each unique pattern
Output	over 180,000 semrel paths; 14,775 unique path patterns

Table 4.4. Summary of Training procedure #1

The 20 highest frequency patterns obtained from the training are shown in Table 4.5 below. As expected, the most frequently occurring patterns often involve the **Hyp** relation, but those containing the **Tobj** relation also occur frequently, lending further validity to the notion of using predicate-argument relations in the determination of similarity.

Path Patterns	Frequencies
Hyp HypOf	14239
Hyp HypOf Hyp	5596
HypOf Hyp HypOf	4891
HypOf Hyp	3453
Hyp	2916
HypOf	2755
TobjOf Tobj	2447
Tobj TobjOf	2339
Hyp Hyp	1928
Syn HypOf	1724
TobjOf HypOf	1691
HypOf HypOf	1640
Hyp Syn	1616
Hyp Tobj	1588
Hyp TobjOf	1245
Syn	1098
Tobj HypOf	1051
Tobj TobjOf Hyp	1021
Hyp Part	883
Syn Syn	852

Table 4.5. 20 most frequent similarity path patterns extracted from an online thesaurus

4.2 Using path patterns to determine similarity

The distribution of the path pattern frequencies extracted from the thesaurus, not surprisingly, turned out to be a hyperbolic curve, as shown in Figure 4.1, much like the distributions of frequencies used to formulate the semrel weights described in Chapter 3. For determining the similarity of unseen word pairs, it was decided that only path patterns with frequencies greater than that of the vertex of the hyperbolic curve (only 168 out of 14,775) were used. This decision was made intuitively at first, since the vertex is the point at which the distribution curve begins to flatten, and it was validated through later experimentation as described in the sections below. The 168 path patterns identified are included in Appendix A.

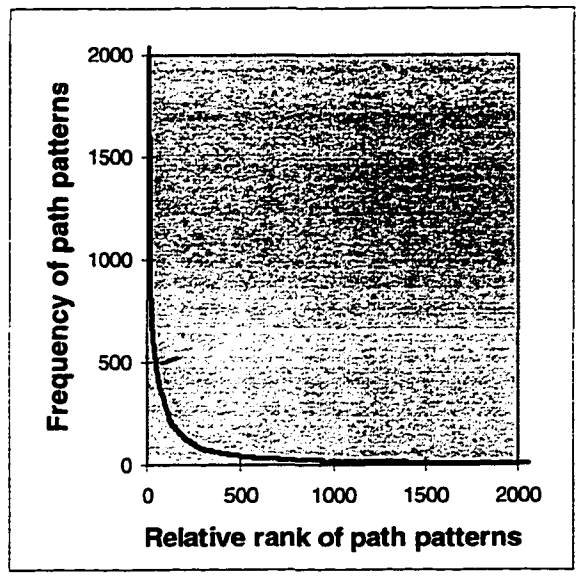

Figure 4.1. Distribution of similarity path pattern frequencies

The similarity potential (*SP*) of each path pattern (*p*) was computed as a simple probability, consisting of the frequency of that path pattern divided by the total number (*n*) of patterns extracted, as given in Equation 4.1 below.

$$SP(p_j) = \frac{F(p_j)}{\sum_{i=1}^{n} F(p_i)}$$

Equation 4.1. Similarity potential of a path pattern

Next, the extracted path patterns, together with their similarity potentials, were loaded into a table in order to be matched against other semrel paths using a simple finite state algorithm. The algorithm takes as input each relation in a given semrel path, and when there are no more relations in the input path and a pattern has been matched, the algorithm assigns a similarity potential to the path and terminates.

Given the extracted path patterns, their similarity potentials, and the matching algorithm, the procedure for determining the similarity of two input words is as follows:

1. Query the LKB for up to 10 semrel paths between the input words (some words may have fewer than 10 paths between them). This is the same number of paths that was used in the extraction of the path patterns and was determined to be a reasonable sample size by experimentation. The semrel paths returned from the query, of course, are ranked according to their semrel path weight, as described in Chapter 3. The first 10 paths represent the most salient connections in the LKB between the two words.

2. Apply the matching algorithm to each of the semrel paths returned from the query, obtaining the similarity potential for the matching path pattern, if any. Assign a similarity potential of zero if the algorithm does not succeed in matching the input path.

3. Compute the average similarity potential of the matched path patterns, including the zero potentials assigned when no match occurred. This average becomes the *similarity score* for the two words under examination. The similarity score (S) for two words (w_1 and w_2) is given formally in Equation 4.2, where M is the matching function described in step 2 above, $Path_i$ is a function which returns the i^{th} weight-ranked semrel path between the two words, and n is the number of semrels paths between the words.

$$S(w_1, w_2) = \frac{\sum_{i=1}^{\min(n,10)} M(Path_i(w_1, w_2))}{n}$$

Equation 4.2. Similarity score for two words

Because the path patterns used in the matching algorithm are only those with a frequency higher than that of the vertex of the distribution of path frequencies, many semrel paths between the input words may not be matched. If more of the path patterns were used, the number of input paths not matched would approach zero, but the performance of the path pattern matcher would decrease and the computer storage required to run it would increase. Through experimentation, it was determined that limiting the number of path patterns used as described above and assigning the similarity potential for unmatched paths to be zero resulted in no significant loss in the accuracy of the system in determining similarity.

By examining the similarity scores for various pairs of similar and dissimilar words, ranges of scores were identified that indicated moderate to strong similarity and conversely, moderate to strong dissimilarity. A threshold was hypothesized which appeared to mark the (somewhat fuzzy) boundary between similarity scores indicating similarity and those indicating dissimilarity. This *similarity score threshold* was .0015, and it was later validated in the experiments described below (in Section 4.3.3).

Returning one last time to the decision only to use path patterns with frequencies higher than that of the vertex of the frequency distribution, the similarity potential corresponding to that vertex frequency is .0006. Given the above threshold of .0015, and the fact that the highest similarity potential (for the most frequent path pattern) is .0787, it is now easy to see why the effective rounding of similarity potentials less than .0006 to zero in the score computation does not adversely affect the outcome of similarity determination.

As an example of the procedure for determining similarity described above, consider the words ***observe*** and ***see***. The 10 highest ranked semrel paths between these words are shown in Table 4.6[1]. The similarity potential of the path pattern that matches each of the paths is also given in the table (the third and ninth paths were not matched). When the similarity potentials are averaged, the resulting similarity score for these words is .013654, which is significantly higher than the similarity score threshold of .0015, indicating that the words are strongly similar.

Semrel paths	Similarity potentials of matched path patterns
observe*—Hyp→*see	.016144
observe*—Hyp→*notice*←Hyp—*see	.078705
observe*—Tobj→*law*—Purp→*see	0
observe*—Hyp→*notice*←Hyp—*overlook*—Hyp→*see	.030974
observe*←Hyp—*keep*←Tobj—*see	.002218
observe*—Hyp→*watch*←Hyp—*oversee*—Purp→*see	.002172
observe*—Syn→*note*—Syn→*recognize*←Hyp—*see	.002454
observe*—Syn→*note*—Syn→*recognize*—Syn→*see	.001803
observe*—Syn→*note*—Syn→*recognize*—Hyp→*see	0
observe*—Hyp→*act*←Hyp—*charade*←Tobj—*see	.002065

Table 4.6. Semrel paths between **observe** *and* **see**, *and the similarity potentials of the path patterns that match them*

4.3 Testing and enhancing the similarity procedure

In order to determine the accuracy of the similarity procedure described in the last section, a test was performed using a different set of over 34,000 noun and verb word pairs from the same online thesaurus having one or more semrel paths connecting them in the LKB. In the test, the pattern matching algorithm and the similarity score computation

[1] The fourth path in the table is interestingly the result of a deficiency in the semrel extraction process that allowed *notice* and *see* to be Hypernyms of *overlook* when the actual definition in LDOCE read "to look at but not see; not notice; miss." This deficiency, along with others, is already being addressed in further enhancements to the LKB outside the scope of this research.

were applied, as described in the similarity procedure, to up to 10 of the highest weighted semrel paths for each of the word pairs. The result was that the scores for 81% of the word pairs in the test set were greater than or equal to the similarity score threshold of .0015. This test is summarized in Table 4.7.

Test procedure #1a	
Input	Over 34,000 noun and verb thesaurus word pairs, connected by at least one semrel path in LKB
Process	Use similarity procedure with path patterns produced by Training procedure #1 to determine similarity
Result	81% of pairs had similarity score ≥ .0015

Table 4.7. Summary of Test procedure #1a

At first glance, this result seemed fairly promising, but there remained certain shortcomings. The ability of the procedure to determine the similarity of words already known to be similar had been established, but there was still no indication of how the procedure would fare with pairs of words in general, and specifically, with those which were definitely not similar. Even more serious for a broad-coverage NLP system was the concern that, while the procedure was reasonably accurate in determining the similarity of words that were connected by semrel paths in the LKB, there were many word pairs in the thesaurus that were not connected by semrel paths in the LKB (about 15,000, in addition to the over 34,000 connected pairs, for an approximate total of 49,000 pairs in the test set), even though the words themselves did have entries in the LKB. When these additional unconnected pairs were considered in the results for Test procedure #1a, the percentage of pairs having a similarity score greater than or equal to .0015 dropped from

81% to 57%! The method for overcoming this serious deficiency will be discussed shortly, but first, the determination of *dissimilarity* will be addressed.

4.3.1 Testing for dissimilarity

To deal with the first of these issues, there had to be a way to test the procedure on words known to be dissimilar, or simply unrelated to one another. (Antonyms, which are often similar to each other in many ways, would not be included in these words.) This was accomplished by the creation of an *anti-thesaurus* using the following procedure:

1. Randomly select words from the word pairs in the online thesaurus, and create new pairs, ensuring that both words in each of the new pairs has the same part of speech (noun or verb in this case), as designated in the thesaurus.

2. Look up both words in each new pair in the LKB to ensure that they are there (the reason for this will be discussed shortly).

3. Look up the new word pair in the thesaurus to ensure that it is *not* there, i.e., that the words are not similar.

Following the above steps, an anti-thesaurus was generated which contains over 100,000 noun and verb word pairs. While the procedure for creating the anti-thesaurus does not guarantee that every word pair will be dissimilar, a spot check of the resulting pairs nevertheless confirmed it to be quite reliable. Its reliability was further validated in an experiment involving evaluation by human reviewers which is described below (in Section 4.3.4).

To measure the ability of the similarity procedure to determine dissimilarity, another test was performed in which approximately 49,000 word pairs were extracted from the anti-thesaurus (matching the actual test set size of Test procedure #1a, including word pairs present in the LKB but not connected by semrel paths). The similarity procedure was then run on each of these word pairs, and the result was that 91% of the pairs had scores less than the similarity score threshold of .0015, confirming their dissimilarity. This test is summarized in Table 4.8.

Test procedure #1b	
Input	Over 49,000 noun and verb anti-thesaurus word pairs, with all words present in the LKB, but not necessarily connected by semrel paths
Process	Use similarity procedure with path patterns produced by Training procedure #1 to determine dissimilarity
Result	91% of pairs had similarity score < .0015

Table 4.8. Summary of Test procedure #1b

Although the result of the dissimilarity test was excellent, one additional set of training and testing was attempted to improve the result even further. Given the existence and viability of the anti-thesaurus, it was decided that dissimilar word pairs, as well as the original similar ones, could be included in the training data from which the similarity path patterns were extracted. Accordingly, a file of over 93,000 noun and verb word pairs was created, with half coming from the thesaurus and the other half from the anti-thesaurus. The words in all of the pairs were present in the LKB, but were not necessarily connected by semrel paths. An adjustment was also made during the training procedure so that when the instances of each unique path pattern were counted, any instance that had resulted from a word pair in the anti-thesaurus was subtracted from (instead of being

added to) the total frequency for that path pattern. The purpose of the subtraction was to penalize the similarity potential of a path pattern that occurred between dissimilar words (as well as possibly between similar ones).

Running the modified training procedure netted over 152,000 semrel paths between the word pairs in the input file and 13,269 unique path patterns with positive frequencies. Note that these numbers are lower than those from the first training procedure (without dissimilar word pairs: 180,000 semrel paths and 14,775 unique path patterns) because of the subtraction of paths from dissimilar word pairs. The procedure is summarized in Table 4.9.

Training procedure #2	
Input	Over 93,000 noun and verb word pairs, half from the thesaurus and half from the anti-thesaurus; all words in the LKB but not necessarily connected
Process	Obtain up to 10 highest ranked semrel paths for each pair; extract and sort path patterns; count instances of each unique pattern, subtracting those that came from the anti-thesaurus
Output	Over 152,000 (net) semrel paths; 13,269 unique path patterns with positive frequencies

Table 4.9. Summary of Training procedure #2

The unique path patterns extracted from this training procedure were then employed in a test in which a different set of over 98,000 noun and verb word pairs were examined for similarity and dissimilarity, half from the thesaurus and half from the anti-thesaurus, with all words present in the LKB but not necessarily connected by semrel paths. The result for identifying similar word pairs was the same as in Test procedure #1a (81% of connected pairs; 57% of all pairs), but the result for dissimilar word pairs improved from that in Test procedure #1b by one percentage point (to 92%). The test is summarized in

Table 4.10 below. While this is not a very significant improvement, every little bit helps, and given that the training data now included dissimilar word pairs, one would expect some improvement in identifying dissimilarity.

Test procedure #2	
Input	Over 98,000 noun and verb word pairs, half from thesaurus and half from anti-thesaurus, with all words present in the LKB, but not necessarily connected by semrel paths
Process	Use similarity procedure with path patterns produced by Training procedure #2 to determine similarity/dissimilarity
Result	81% of connected pairs and 57% of all pairs from thesaurus had similarity score ≥ .0015; 92% of all pairs from anti-thesaurus had similarity score < .0015

Table 4.10. Summary of Test procedure #2

4.3.2 Increasing the number of semrel paths in the LKB

While 92% detection of dissimilarity and 81% detection of similarity are excellent results, the serious inconsistency of considering only connected pairs in determining similarity, and the poor result of only 57% if one does not, needed to be resolved. One should rightly not expect that pairs of dissimilar words have semrel path connections between them, and therefore the basis for the dissimilarity tests seems justified. In the similarity tests, however, one can simply not count on the fact that only words having connections between them in the LKB will occur in general text.

Given the need for a common basis that includes words in the LKB, whether or not they are connected, the most obvious way of increasing the similarity test percentage is by increasing the number of words in the LKB that are connected, even though this may also increase the chances of falsely identifying dissimilar words as similar. In Section 2.3 of Chapter 2, the lack of connections between a wide range of words in the LKB was

attributed to the limited defining vocabulary of LDOCE, from which the LKB was created. It was suggested there that this lack could be overcome by adding semrel structures to the LKB that were extracted from the example sentences of LDOCE (not just the definitions) as well as from the definitions and example sentences of other dictionaries not having a limited defining vocabulary.

Such an expansion of the LKB was performed,[2] adding semrel structures from LDOCE example sentences and semrel structures from both the definitions and example sentences in the *American Heritage Dictionary, 3rd Edition* (Houghton Mifflin Company 1992) (abbreviated hereafter as AHD3). Although the AHD3 is a collegiate sized dictionary with an unrestricted defining vocabulary, the procedures for extracting semantic relations from the definitions were not modified or enhanced in any way during the creation of this first version of an expanded LKB (Dolan and Richardson 1996). Of even greater significance for the research performed by the author is that the same processes of semrel structure inversion, semrel weighting, semrel path creation, and similarity determination that were applied to the original version of the LKB based only on LDOCE were also applied to this expanded LKB with no modification. Of course, in creating the expanded LKB, many opportunities for future improvement were identified, but for the purposes of this research, simply using the same processes to increase the number of semrel path connections between a wider range of words validates the applicability and usefulness of those processes and maintains a consistent research

[2] The author is indebted to Joseph Pentheroudakis, Lucy Vanderwende, and William Dolan, colleagues at Microsoft, who were also principal participants in the creation of the expanded LKB.

context in which to compare the results of similarity training and testing procedures with those that used the original LKB.

The expanded LKB, based on noun and verb definitions and example sentences from both LDOCE and AHD3, contains a significant amount of additional information. In particular, the statistics regarding its creation, rounding numbers to the nearest thousand, are as follows:

- Semantic relation extraction (including parsing) processed 114,000 single word noun definitions, 33,000 single word verb definitions (a total of 147,000 definitions out of the 182,000 definitions in MIND), and 36,000 noun and verb example sentences in about 27 hours on a 120Mh Pentium PC. As a result of extraction processing, over 513,000 semantic relations contained in over 167,000 semantic relation structures were created. (Note that not all definitions and example sentences produced a semrel structure due to various implementation limitations, e.g., insufficient storage for processing.)

- Semrel structure inversion processed the 167,000 structures created during extraction to create almost 870,000 inverted semrel structures, stored on 82,000 of the 157,000 headwords in the expanded LKB. The inversion process, together with the computation and assignment of weights to the inverted structures, required about 4 hours of processing time on the same PC.

Comparing the statistics of the two LKBs in Table 4.11 below, one can easily see that the breadth of coverage of the expanded LKB significantly increased over the original LKB. (Statistics on the creation of the original LKB were given in detail at the

end of Section 2.3 in Chapter 2.) In particular, the number of headwords now associated with semrel structures (i.e., connected by semrel paths) increased from 23,000 to 82,000. This substantial increase constitutes the enhancement needed to improve the coverage of the similarity procedure.

	Original LKB	Expanded LKB
Dictionaries used	LDOCE only	LDOCE and AHD3
Headwords	39,000	157,000
Noun and verb definitions	45,000	147,000
Noun and verb example sentences	0	36,000
Semrels	147,000	513,000
Semrel structures	45,000	167,000
Inverted semrel structures	180,000	870,000
Headwords with semrel structures	23,000	82,000

Table 4.11. Comparison of information in original and expanded LKBs

4.3.3 Using the expanded LKB

With the expanded LKB, similarity training and testing procedures were performed using the same data as with the original LKB. A file of over 102,000 noun and verb word pairs, half from the thesaurus and half from the anti-thesaurus, was used to obtain similarity path patterns from the expanded LKB. The increase in training data over the 93,000 word pairs used in Training procedure #2 was due to an increase in the number of words present in the LKB (the coverage of the expanded LKB being greater, of course). Instances of path patterns that resulted from word pairs in the anti-thesaurus were also subtracted from the frequency tallies, as in Training procedure #2.

Running the training procedure with the expanded LKB resulted in obtaining over 285,000 (net after subtractions—there were over 397,000 total) semrel paths between the word pairs in the input file and 13,477 unique path patterns with positive frequencies.

Note that the number of unique path patterns in this case is very close to the number of unique patterns extracted in Training procedure #2 (13,269), even though the number of semrel paths increased by 87%. This confirms that the same patterns continue to indicate similarity throughout the LKB, no matter the number of paths, and it further validates the method of using semrel paths to determine similarity. The training procedure is summarized in Table 4.12.

Training procedure #3	
Input	Over 102,000 noun and verb word pairs, half from the thesaurus and half from the anti-thesaurus; all words in the expanded LKB but not necessarily connected
Process	Obtain up to 10 highest ranked semrel paths for each pair; extract and sort path patterns; count instances of each unique pattern, subtracting those that came from the anti-thesaurus
Output	Over 285,000 (net) semrel paths; 13,477 unique path patterns with positive frequencies

Table 4.12. Summary of Training procedure #3

The unique path patterns extracted from this training procedure were used to test a different set of over 107,000 noun and verb word pairs for similarity and dissimilarity, much like Test procedure #2. This time, however, due to the significant increase in semrel path connections in the expanded LKB, 84% of *all* of the word pairs (not only the connected ones) from the thesaurus were recognized as being similar. (Comparing with the previous similarity test procedures, 89% of the *connected* word pairs were identified as similar.) Unfortunately, the percentage of word pairs from the anti-thesaurus recognized as dissimilar dropped to 82% (from 92% in the previous tests). This, too, is undoubtedly caused by the increase in semrel path connections in the LKB (creating additional false indications of similarity), but it also appears to be the price that must be

paid to improve the similarity percentage significantly. The test is summarized in Table 4.13.

Test procedure #3	
Input	Over 107,000 noun and verb word pairs, half from thesaurus and half from anti-thesaurus, with all words present in the expanded LKB, but not necessarily connected by semrel paths
Process	Use similarity procedure with path patterns produced by Training procedure #3 to determine similarity/dissimilarity
Result	84% of all pairs from thesaurus had similarity score ≥ .0015; 82% of all pairs from anti-thesaurus had similarity score < .0015

Table 4.13. Summary of Test procedure #3

In spite of the decrease in accuracy of dissimilarity recognition, the average accuracy of the similarity procedure for identifying both similarity and dissimilarity rose from 74% (57% similarity, 92% dissimilarity) to 83% (84% similarity, 82% dissimilarity) by using the expanded LKB. It is also this same average accuracy rate that was used to further validate the similarity score threshold of .0015. By running the combined similarity/dissimilarity test procedure repeatedly using slightly higher or lower thresholds each time, it was observed that the average accuracy rate was highest when the threshold was approximately .0015, and as the threshold was adjusted upward or downward from this point, the accuracy dropped. For example, if the threshold is decreased to .001, the average accuracy drops a half percentage point to 82.5%.

4.3.4 Comparing the similarity procedure to human performance

During the many experiments described above, it was observed that the training data and testing data taken from the thesaurus and anti-thesaurus did not always match the author's intuitions about similarity or dissimilarity. Because of this observation, and

the desire to accurately assess the similarity procedure's ability to make correct judgments regarding the similarity or dissimilarity of a given word pair, a final test was performed using human evaluators.

To perform the test, a file of 200 word pairs (with their part-of-speech designation—noun or verb) was created by taking 100 random pairs from the thesaurus and another 100 random pairs from the anti-thesaurus. These pairs were then randomly arranged in the file, although all the noun pairs were kept together and all the verb pairs were kept together to facilitate human evaluation. The file was then given to five individuals who were asked to indicate whether each pair contained similar or dissimilar words, and it was also processed by the similarity procedure. The file is included in Appendix B.

The individuals were informed that half the word pairs came from a thesaurus and the other half from an anti-thesaurus (the creation of the anti-thesaurus was explained), so that they might have a better understanding of the criterion for judging similarity (i.e., those that came from the thesaurus). Four of the five individuals were linguists or lexicographers, and only one of the five was a non-native speaker of English (although the non-native speaker was also a highly competent lexicographer of English). The high level of lexicographical expertise among the participants ensured the most demanding comparison possible in evaluating the capability of the similarity procedure.

The results of the test are given in Table 4.14. Surprisingly, the similarity procedure performed nearly as well as the humans in determining similarity. (Or one might say that the humans performed as poorly as the similarity procedure in this case!)

The humans did perform significantly better than the procedure in determining dissimilarity, however. In examining the results of the similarity procedure more carefully on individual word pairs, a good part of the reason for the procedure's weakness in determining dissimilarity can be traced to the presence of inaccurate semrel paths in the expanded LKB, creating false indications of similarity. Remember that no enhancements to the process of semrel extraction were made when the expanded LKB was created. It is anticipated that through iterative refinements, making allowances for the richer vocabulary and definitional structure in the AHD3, many of these inaccuracies can and will be eliminated.

	Similar pairs correct	Dissimilar pairs correct
Human #1	91	91
Human #2	74	97
Human #3	80	96
Human #4	86	93
Human #5	86	89
Average human scores	83	93
Similarity procedure	82	80

Table 4.14. Results of test comparing similarity procedure to human performance

In making an overall comparison of accuracy (averaging the similarity and dissimilarity scores together) between the similarity procedure and competent human performance, it is certainly pleasing to note that the difference of 7% (88% average similarity/dissimilarity score for humans and 81% for the similarity procedure) is certainly less than the 19% deficiency (from 100%) one observes when looking at the similarity procedure's results alone. In this regard, the author's intuitions about the quality of the training and testing data were justified, and the question may be raised as to how well the similarity procedure would perform if the training data were improved. One

possibility in this regard, although a tremendously tedious one, would be to have human lexicographers manually review some or all of the training data before extracting the semrel path patterns. For now, however, planned enhancements will most likely focus on improved semrel extraction.

4.3.5 Advantages of the similarity procedure

To conclude this section on testing and enhancing the similarity procedure, it seems appropriate to list a few of the advantages it has over other approaches that have been tried. Perhaps the principal advantage over approaches that use a thesaurus directly to determine similarity is that the similarity score obtained from semrel paths in the LKB easily captures types of similarity other than straight synonymy. For example, word pairs such as "pen-pencil" and "mother-father" are not in the online thesaurus used in this research, and neither are there any clear indirect connections (such as through "writing instrument" or "parent"), although both pairs are identified as being very similar. Of course, this may differ from thesaurus to thesaurus, but the LKB approach uses a wealth of different semantic relations, any number of the combinations of which may indicate similarity, according to the semrel path patterns used, and it is not limited to a hard-coded taxonomy of paradigmatic relations.

Another advantage of the overall approach of using an automatically generated LKB in similarity determination is its extensibility into new domains based on the availability of definitional or even straightforward expository text. Many technical or specialty domains have dictionaries, glossaries, or simple manuals that can be processed automatically to obtain the semantic relations needed to populate an LKB and to provide

the basis for similarity determination. Although it would be more difficult to obtain paradigmatic relations (e.g., **Hyp**, **Syn**, **Part**) from straight text (such as a manual or other corpus), obtaining certain syntagmatic relations (e.g., **Tobj**, **Tsub**) would not be difficult, and as shown above, and in the work by Hindle (1990) and Grishman and Sterling (1994), these can also be very useful in determining similarity.

Lastly, the similarity procedure, based as it is on semrel paths in the LKB, is seamlessly integrated with the process of querying the LKB for specific relations between words. Both processes share the capabilities provided by semrel inversion, semrel weights and the extended semrel path mechanism. Together, they constitute a coherent, integrated methodology for inferring semantic relations, as discussed in the next section.

4.4 Inferring relations using the similarity procedure

At beginning of this chapter, inferring semantic relations not explicitly present in the LKB was discussed in the context of the example sentence:

(1) ***My friend observed the tragic scene from a distance with binoculars.***

In order to resolve the attachment ambiguity of the prepositional phrase ***with binoculars***, the Microsoft NLP system queries the LKB for semrels representing the different possible attachments, including the semrel ***observe***—**Means**→***binoculars***. If the semrel is not in the LKB (and it is not), then it must be inferred through similarity processing, if possible. Now that the similarity procedure has been described in detail, the process of inferring semrels may also be described.

The idea behind inferring semrels is simple: find a semrel that is explicitly in the LKB that is very close to the input semrel, e.g., one word is different, and check the similarity score between the original word and the different word. For example, if the semrel to be inferred is ***observe*—Means→*binoculars***, and ***look*—Means→*binoculars*** is explicitly in the LKB, then check the similarity score between the words ***observe*** and ***look***. If the words are similar, then the semrel may be inferred.

However, the actual inference procedure requires the resolution of certain issues. For instance, in the above example, should all semrels in the LKB that contain **—Means→*binoculars*** be queried, and then the similarity between ***observe*** and the words in the first position of those semrels checked, or should semrels that contain ***observe*—Means** be queried instead, and then the similarity between ***binoculars*** and the words in the second position be checked? Or should both checks be performed to ensure adequate coverage of all the possibilities in the LKB?

A weakness of this approach (with either one check or two) is that, very possibly, the sense in which two words are similar is not the same sense of the word that participates in the relation. During the discussion of extended semrel path creation back in Chapter 3, it was noted that any time there is a *join* of simple semrel paths (i.e., those paths that exist solely within the context of a single semrel structure) into an extended path, a possible sense mismatch exists. In this approach, there are multiple possibilities where three simple semrel paths (with two join points) may participate in the inference—two in the extended semrel paths used to determine similarity and an additional one that consists of the relation in question. The result in such cases may easily be insufficiently

constrained ambiguity leading to inferences that are spurious. In the future, this approach may be much more viable with a sense disambiguated LKB, which is work currently in progress (Dolan 1994).

Another approach would be to query all the extended semrel paths between the two words in the semrel to be inferred (***observe*** and ***binoculars*** in the example) and then check each extended path to see if it contains either word in a **Means** relation, with the remainder of the extended path indicating similarity according to the similarity path patterns. Such a path between ***observe*** and ***binoculars*** currently in the LKB is:

(2) *observe*←**Hyp**—*watch*—**Hyp**→*look*—**Means**→***binoculars***

The weakness with this approach is that it is perhaps a little too constrained, since the similarity procedure as described above is based on a sample of up to 10 paths between words, and in this case, a determination would be being attempted on the basis of only one semrel path (even if, as in the example case here, it is correct). Nevertheless, the positive aspect of this approach is that it does keep the context constrained to the paths between the two words.

As a compromise, a hybrid approach was implemented. All the extended semrel paths between the two words in the relation to be inferred (***observe*** and ***binoculars***) are queried, as in the second approach, and if either query word is found in a **Means** relation (***binoculars*** in (2) above), then the other word in the relation (***look***) is checked for similarity, using a complete application of the similarity procedure, with the other query word (***observe***). This approach keeps the context constrained and at the same time allows for full similarity checking. An interesting side effect is that in a particular path that is

the basis for the inference, the remainder of the path (other than the query relation—**Means** in the example) does not necessarily need to indicate similarity. In (3) below, for example, which is another path between ***observe*** and ***binoculars*** currently in the LKB, the **Quesp** (or **Quespernym**; defined in Table 2.1) relation by itself is a very weak indication of similarity (it is ranked below the vertex of the similarity path pattern distribution), but the complete application of the similarity procedure to the words ***telescope*** and ***binoculars*** yields a high similarity score, resulting in the desired inference.

(3) ***observe***—**Means**→***telescope***←**Quesp**—***binoculars***

To give an idea of the similarity coverage that may still be obtained with the hybrid inference procedure (despite the constraints imposed on it) for the —**Means**→***binoculars*** example, Table 4.15 lists the words that may be inferred in the first position of that relation, depending on the algorithm being used. The first column lists the only word contained in an explicit semrel in the LKB, the second column lists those words that may be inferred with the hybrid procedure, and the third column lists the additional words inferred using the first approach described above, which, although it provides more correct inferences, also generates many incorrect ones. Note that most of the common words and even some uncommon ones may be inferred with the hybrid algorithm.

Explicit semrels	Semrels inferred using only paths between words	Semrels inferred using all paths containing relation
look	behold	inspect
	contemplate	peruse
	examine	reconnoiter
	glimpse	spot
	observe	spy
	peek	sweep
	peer	
	scan	
	scout	
	scrutinize	
	see	
	sight	
	survey	
	view	
	watch	

Table 4.15. Words explicit or inferred in the first position of the relation —***Means***→**binoculars**

In order to measure the capability of the (hybrid) inference procedure, a test was performed in which 295 semrels were gathered from human feedback on a questionnaire[3] designed to solicit common relationships between objects and actions. For example, one question was "What do you use to do an activity? (e.g., you might draw with a pen, cut with a knife, row with an oar)." (Appendix C contains the complete questionnaire.) Based on the responses, semrel representations were formulated for 5 different relation types: **Tobj**, **Tsub**, **Means**, **Matr**, and **Locn**. These semrels were then used as input to the inference procedure, and the number of semrels recognized correctly (either because they were explicit in the LKB or, more often, inferred) for each relation type is given in Table 4.16. As expected, the **Tobj** and **Tsub** relations were recognized correctly more

[3] Thanks goes to Lucy Vanderwende for administering the questionnaire to17 individuals and compiling their responses.

often, given their higher frequency throughout the LKB. The **Matr** relation scored lowest, ostensibly because the material of which an object is made is not often given in dictionary definitions, and even with the ability to infer relations, there are simply not enough **Matr** relations to serve as the basis for broad coverage of this relation.

Semrel relation types	Recognized/Total	Recognition percentage
Tobj	43/47	91
Tsub	52/57	91
Means	35/50	70
Matr	21/42	50
Locn	71/99	72
All relations	222/295	75

Table 4.16. Recognition rates of semrels taken from human feedback

While the results in the table indicate the inference procedure's overall coverage (also known as *recall*), they do not indicate its precision, i.e., how many of the semrels it will recognize are valid vs. invalid. In this regard, when semrels are inferred, the path weight of the highest ranked path that was used to infer the semrel may be used to provide a score of relative confidence in the validity of the inferred semrel. To check this hypothesis, another test was performed in which all the semrels representing the human feedback for a particular relation type were modified so that the same word was in the first position. The word chosen was one of the valid words for one or more of the relations, but for the majority of the others, it was not valid. The inference procedure was then run on all the modified semrels and the results were ranked according the path weight as described above. The top five semrels in each of two tests (one using the word ***bake*** in a **Tobj** relation, and one using the word ***write*** in a **Means** relation) are given in Table 4.17 below. (The complete results are given in Appendix D.)

Weight	Semrel
5.12860E-06	***bake*—Tobj→*cake***
5.12860E-06	***bake*—Tobj→*food***
1.68270E-07	***bake*—Tobj→*clay***
1.01620E-08	***bake*—Tobj→*dish***
8.37530E-09	***bake*—Tobj→*cookie***
1.52410E-07	***write*—Means→*brush***
6.18020E-11	***write*—Means→*crayon***
1.42230E-12	***write*—Means→*whisk***
1.22460E-12	***write*—Means→*pen***
7.70900E-13	***write*—Means→*pencil***

Table 4.17. Highest ranked 5 semrels in each of two inference procedure tests

The examples in the table show clearly that the path weights provide an intuitively correct ranking for many of the semrels. Since the rankings are relative for each set of relations, it is difficult (and not really necessary) to establish an exact threshold for correctness or to obtain a meaningful precision percentage. One could achieve 100% precision in these tests by arbitrarily establishing a cutoff after the fifth semrel. The recognition (or recall) percentage would suffer a bit, however, since, as shown in the complete data in Appendix D, there are a few semrels that are placed incorrectly in the lower portion of the ranking, mainly due to deficiencies in the information in the LKB.

In conclusion, the inference and similarity procedures described in this chapter have a number of advantages. In Chapter 5, these advantages will be positioned favorably in relation to existing methods. The scope of the work reported here shows that these procedures do indeed "scale up" to acceptable levels of usefulness and accuracy in the processing of general text. Much refinement remains to be done, but these procedures have already proven to be invaluable components of a broad-coverage NLP system.

5. Comparison with related research

In previous chapters, several references have been made to related research by others. In particular, Chapter 3 provided an extensive comparison of the *averaged vertex probability* weighting method to weighting methods employed in other work. This chapter contains an overview of the important issues that distinguish the current research from other work. In so doing, some of the references will be revisited in greater detail and additional ones will be provided. The information contained herein is organized by the issues involved rather than by the examination of each reference sequentially, with the hope that the reader will obtain a better understanding of the unique aspects of this work.

The present research has its roots firmly planted in dictionary-based (DB) methods for NLP. This stems from using the original work by Jensen and Binot (1987) and some of the extensions to that work proposed by Montemagni and Vanderwende (1992) as the foundation for this research. However, the desire to integrate aspects of example-based (EB) processing, specifically the methods for determining similarity that are commonplace in EB approaches, into the matching algorithms used to infer semantic relations (semrels) during prepositional phrase attachment in the Microsoft NLP system led the author to examine more closely the potential similarities between DB and EB methods. The advent of an increased focus on corpus-based (CB) methods in recent years provided an additional perspective on determining similarity and inferring relations between words, and this perspective was also found to complement and validate the work in progress. The result is the synthesis of some aspects of the three methods into a consistent framework, as described in previous chapters.

A number of parallels exist among DB, EB, and CB methods, not the least of which is that linguistic knowledge is extracted directly from NL text by automatic or semi-automatic means and structured in a way that facilitates NL processing. In each case, attempts have been made to extract and represent information that is deeper, or more semantic, in nature, rather than simply strings of characters or words. The primary unit in many of these representations has been a triple of the form ***word1*** **Relation** ***word2*** (which is certainly true in the present research as well) although oftentimes the relation is unspecified and the unit takes the form of a superficial word co-occurrence bigram. Another common thread in all three of these methods has been the use of a similarity metric and a corresponding inference procedure to provide a fuzzy match capability for matching new relations in input text to existing, extracted relations, whether in an LKB (in DB methods), in an example base (in EB methods), or as represented by statistical parameters (in CB methods).

Each of the three methods has been used to overcome the so-called *knowledge acquisition bottleneck* and automatically internalize large amounts of linguistic (preferably semantic although often as simple as lexical co-occurrence) information sufficient to perform adequately several NLP tasks, including word sense disambiguation, phrasal attachment and other structural disambiguation, and lexical or structural translation between languages. CB methods have been used additionally (and widely) for speech recognition processing.

5.1 Dictionary-based methods

Dictionary-based methods make use of available machine-readable dictionaries (MRDs) to create computational lexicons, or lexical knowledge bases (LKBs). The organization of these LKBs has ranged from structured dictionary-like entries that contain semantic selectional feature information, pronunciations, and usage information (Byrd et al. 1987, Alshawi et al. 1989) to hierarchical noun taxonomies (Copestake 1990, Klavans et al. 1990, Vossen 1991, Bruce and Guthrie 1992) to more complex networks (Veronis and Ide 1990, Kozima and Furugori 1993). The structure of the LKB created in this research is clearly most similar to the complex network type.

DB methods have been applied to aspects of NL analysis such as phrasal attachment (e.g., Jensen and Binot 1987, Vanderwende 1990) and word sense disambiguation (e.g., Veronis and Ide 1990, Braden-Harder 1993, Wilks et al. 1992, Niwa and Nitta 1994). This is consistent with the use of the methods implemented in this research to determine correct prepositional phrase attachment (Richardson et al. 1993)and word sense disambiguation (Dolan and Richardson 1996) in the context of the Microsoft NL understanding system. DB methods have also been applied to determine textual cohesion (Kozima and Furugori 1993) and to aid in information retrieval processing (Chakravarthy 1994).

5.1.1 Taxonomies vs. networks

Some of the earliest DB work by Amsler (1980) identified the taxonomic nature of the genus terms in the *Merriam-Webster Pocket Dictionary.* Chodorow et al. (1985) extended this notion, focusing on the extraction of those taxonomies from the definitions

in an MRD. Chodorow and his colleagues pointed out that definitions generally consist of a genus term and differentia, which distinguish the term being defined from other terms belonging to the same genus. Through a straightforward heuristic pattern-matching algorithm, he found that one could identify the genus terms in definitions and then structure them in a hierarchical taxonomy.

A focus on extracting taxonomies from MRDs has persisted throughout the history of DB research (e.g., Markowitz et al. 1986, Copestake 1990, Klavans et al. 1990, Vossen 1991, Bruce and Guthrie 1992), and many researchers have devoted significant effort to creating neatly-structured hierarchies like the one shown in Figure 5.1, at times seeming to overlook the true tangled, circular nature of the taxonomies actually defined by many of the dictionary genus terms.

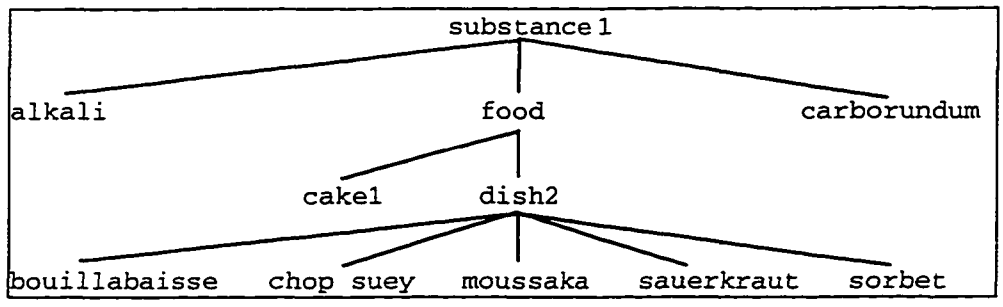

Figure 5.1 Example of a substance taxonomy extracted from LDOCE, taken from Vossen and Copestake (1994)

As the ACQUILEX project in Europe drew to a close, Ide and Veronis (1993) observed that attempts to create formal taxonomies automatically from MRDs had failed to some extent. They cited problems with circularity and inconsistency (among others) in the resulting hierarchies. These are demonstrated in Figure 5.2, where a ***tool*** is defined as an ***implement***, an ***implement*** as a ***tool*** or a ***utensil***, and a ***utensil*** as a ***tool*** or an ***implement***. Also shown is the inconsistency in the genus terms for ***spoon***, ***knife***, and

fork. It almost seems strange that researchers would take so long to come to such realizations when some of the early work (e.g., by Amsler 1980 and Chodorow et al. 1985) actually discussed these issues.

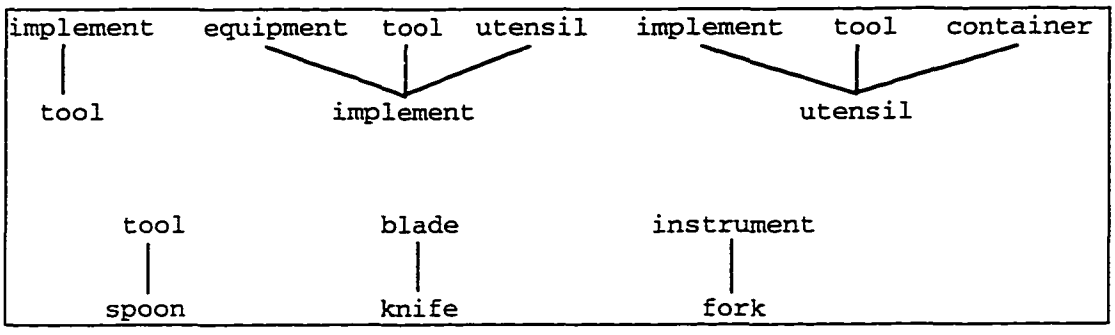

Figure 5.2 Examples of genus term circularity in the Collins English Dictionary, and genus term inconsistency in LDOCE, taken from Ide and Veronis (1993)

A few years before their critical evaluation of the progress made with MRD processing, Veronis and Ide (1990) described an alternative to creating hierarchical taxonomies. They created a series of explicit links between the content words in dictionary definitions and the dictionary entries for those words themselves, thereby creating a forward-chaining network structure throughout the dictionary. They then experimented with a spreading activation technique common to neural network processing in order to perform word sense disambiguation. A few years later, Kozima and Furugori (1993) described related work in which they structured a subset of LDOCE as a neural network and used spreading activation to determine word similarity. In both cases, although only a small portion of the MRD had been converted into a network, the researchers nevertheless explored beyond the paradigmatic taxonomies that had been the object of so much effort and identified some of the additional potential in the syntagmatic relations contained in MRDs.

The present research is certainly not limited to taxonomies, but through the use of fully inverted semrel structures, it creates a networked LKB of a scope that has not been previously documented in DB research. The resulting network exploits both the paradigmatic and syntagmatic relations contained in dictionary definitions and example sentences in a consistent fashion, thereby squeezing every potential piece of useful information it can out of the MRDs processed during its creation.

5.1.2 Co-occurrence relations vs. labeled relations

Identifying and quantifying instances of word co-occurrence has been a long-standing CB method for obtaining useful information about a language in order to perform, for example, speech recognition. It has also been used with some success on dictionary definition text, and in this sense, the text contained in the definitions is considered a NL corpus like any other.

Wilks et al. (1989, 1992) describe work by Plate, who processed the definitions in LDOCE and obtained co-occurrence statistics for the words contained in the LDOCE defining vocabulary. Using special display programs, he was then able to establish a threshold for word relatedness and to show in network format all of the words related to a given word as well as the relations among those words and other closely related words. This local network constituted a sort of "semantic space" around the given word which was useful for tasks such as word sense disambiguation.

Others who created network structures from dictionaries through the use of co-occurrence statistics were Veronis and Ide (1990), Kozima and Furugori (1993), and Niwa and Nitta (1994). In each of these cases, as well as in the work by Plate, semantic

relatedness was measured through a comparison of the networked contexts surrounding words. While such measurements have been found useful for word sense disambiguation (the goal of those efforts), because the actual relations between words have not been specified (or labeled), they have not been as useful for the needs of structural disambiguation (e.g., prepositional phrase attachment).

Labeled relations, while being more difficult to obtain, provide greater power for handling many aspects of NLP. For example, in the work of this dissertation, they provide a means for correctly attaching ambiguous prepositional phrases by specifying the relation that holds between the object of the preposition and the constituent being modified by the prepositional phrase. This kind of relation is clearly syntagmatic, and Slator (in Wilks et al. 1992) discussed the possibility of extracting such relations (e.g., **Purpose** and **Instrument**) from dictionary definitions, but did not actually do it. Alshawi (1989) did demonstrate the limited extraction of other fairly generic syntagmatic relations such as **Property** and **Object**.

In contrast, the kinds of labeled relations found in the taxonomies created by the researchers cited in the previous section are paradigmatic; they include **IsA** (or **Hypernym**), **Part** (or **Meronym**) and **Synonym**. These relations are generally very useful for word sense disambiguation and similarity determination, in like fashion to unlabeled co-occurrence relations, but their specificity provides additional power. The **Part** relation, for example, although inherently paradigmatic, can play a useful role in structural disambiguation, such as in the sentence ***I saw the bird with the yellow beak***,

where the **Part** relation between ***bird*** and ***beak*** indicates that the prepositional phrase should be attached to ***bird*** rather than to ***saw***.

Labeled relations, as used in the present work and in the work of others, clearly provide additional flexibility and usefulness over simple co-occurrence relations. While others acknowledge their utility, they are often either unable (e.g., lacking computational tools such as the semrel extraction procedure described in Chapter 2) or unwilling (e.g., exclusively focused on statistical methods) to make the effort to obtain them. The result is that, while they may have been able to determine that pairs of words such as ***river/bank*** (Plate in Wilks et al. 1992) and ***write/pen*** (Veronis and Ide 1990) are related, they have not been able to specify the relations ***river***—**Part**→***bank*** and ***write***—**Means**→***pen*** as do the semrels contained or inferred in the LKB described in this work. Documented research that has extracted (or discussed the possibility of extracting) such relations from dictionaries has not done so with the same scope (across 167,000 definitions and example sentences) or for as great a number of relations (16 bi-directional relations) as the present work.

5.1.3 Pattern-matching vs. parsing for extraction

In early work on extracting information from the definitions of MRDs, Chodorow et al. (1985) used a heuristic pattern-matching algorithm to identify genus terms. Alshawi (1989) extended the use of patterns, or templates, to extract other kinds of relations (e.g., **Object**) as well. Although these efforts were fairly successful, both were limited—Chodorow to extracting just genus terms and Alshawi to processing only 500 definitions. The weaknesses of using pattern matching became increasingly clear; some dictionary

definitions were quite complex syntactically, and to do a good job of extracting a variety of relations, a more powerful analysis, if such were available, would be required.

Ahlswede and Evens (1988) attempted to evaluate the relative merits of string-based pattern-matching vs. syntactic parsing for relation extraction from dictionary definitions. They used the Linguistic String Parser (LSP), purported to be a broad-coverage syntactic parser, to process definitions, implementing LSP grammars that were specifically adapted to handle definitions according to the part of speech of the defined word. Unfortunately, they found the LSP to be prohibitively slow and lacking the flexibility to handle complex and possibly ill-formed text. This led them to conclude that their string matching algorithms were more productive and useful for the time being. One could easily expect that present day advances in parsing technology and computer performance would have made a significant difference in outcome had they been available at that time.

Slator's "lexicon producer" methodology described in Wilks et al. (1989, 1992, 1996) used a chart parser that was specially created to process LDOCE definitions (*not* a broad-coverage parser, the authors emphasized) in order to extract the genus terms. These were then placed in frame-like structures along with other information extracted from LDOCE to create sub-domain specific lexicons for use in parsing. The possibility of using this parser to extract other semantic relations (e.g., **Instrument** and **Purpose**) was described, but there is no indication that the extraction of such relations was ever carried out to a significant extent.

Meanwhile, researchers participating in the ACQUILEX project in Europe developed parsers, again specifically designed to process dictionary definitions, to obtain taxonomic relations as well as predicate argument relations (e.g., Vossen 1991). These parsers were applied to all the definitions in LDOCE as well as to a few thousand definitions in the Van Dale dictionary for Dutch. In both the Wilks and Vossen work, parsing definitions with limited syntactic parsers for the extraction of genus terms (i.e., **Hypernym** relations) was fairly successful, but the extraction of other types of relations proved more difficult.

The foundation for the present research was laid by Jensen and Binot (1987), who exploited the broad-coverage PLNLP syntactic parser to analyze definitions in *Webster's Seventh New Collegiate Dictionary* to extract semantic relations such as **PartOf** and **Instrument**. They used this information together with a set of heuristic rules to rank the likelihood of alternate prepositional phrase attachments in parse trees. Vanderwende (1990) extended this work to the determination of participial phrase attachments, and Montemagni and Vanderwende (1992) significantly increased the number of semantic relations extracted from 4,000 noun definitions using sophisticated syntactically-based patterns.

Vanderwende (1996) described in detail the semrel extraction process used in this work, which is based on the broad-coverage MEG syntactic parser. An overview of this process, including a brief description of the parser, was provided in Chapter 2. The MEG parser, which proved capable of producing reasonable logical forms for over 167,000 definitions and example sentences from LDOCE and AHD3, has been the enabling

component of this research. The use of this or a similar NL parser is essential to the automatic extraction of a rich set of semantic relations from dictionaries, and in the future, from other sources such as encyclopedias and text corpora.

5.1.4 Simple relations vs. inverted relation structures

Early DB work, as already cited above, focused on the extraction of paradigmatic relations, in particular **Hypernym** relations (e.g., ***car*—Hypernym→*vehicle***). Almost exclusively, these relations as well as syntagmatic ones have continued to take a form similar to that of triples (Wilks et al. 1992, Chakravarthy 1994, Vossen and Copestake 1994), and although these relations have sometimes been taken from larger structures (e.g., the predicate argument structures of Alshawi (1989) or Vossen (1991)), the full contexts provided by these structures have generally not been exploited. Instead, the individual triples have been connected to form hierarchies or stored in *qualia* structures (Vossen and Copestake 1994) or other types of frame structures (Wilks et al. 1992). For labeled relations, only Alshawi (1989), and more recently, Barrière and Popowich (1996), appear to have been interested in entire *semantic structures* extracted from dictionary definitions, as shown in Figure 5.3 below. However, in neither case were a significant number of them extracted, and in the former case, no linked network structure was formed. In the latter case, a fairly rich network structure of very limited scope was created from definitions in a children's dictionary.

```
(launch)

(a large usu. motor-driven boat used for carrying
 people on rivers, lakes, harbours, etc.)

((CLASS BOAT) (PROPERTIES (LARGE))
  (PURPOSE
   (PREDICATION (CLASS CARRY) (OBJECT PEOPLE))))
```

Figure 5.3 "Semantic structure" extracted for **launch**, *taken from Alshawi 1989*

Veronis and Ide (1990), on the other hand, explicitly linked, with *unlabeled* links, the content words in a subset of the *Collins English Dictionary* definitions to the entries for those words themselves, thereby creating a neural-network-like structure of a few thousand nodes. Nodes representing words from the textual context of a word to be disambiguated were "activated" in the network, and the activation spread forward through the links until the network reached a stable state in which nodes representing the correct word senses (corresponding to specific definitions) had the highest activation level. Kozima and Furugori (1993) and Kozima and Ito (1995) built a similar neural-network structure from a subset of LDOCE definitions and used a spreading activation algorithm to determine word relatedness in order to measure textual cohesion.

In the above-referenced neural-network research, the links (and therefore activation) were only forward-chaining. This means that headwords were only related directly to the words in their definitions, which were then looked up as headwords and related to the words in their definitions, and so forth. Words co-occurring in the same definition were not related directly and neither was a word related to any of the headwords whose definitions mentioned it (i.e., no backlinking). In contrast, the fully inverted semrel structures stored in the LKB described in this dissertation link words in a backward direction as well, i.e., they relate words to the headwords whose definitions

mention them. In addition, two words may be related by virtue of their joint appearance in the definition of a third word. For example, Kozima and Furugori (1993) complained that in their network, ***tree*** was not related to ***leaf*** because the latter was not mentioned in the definition of the former. However, in fully inverted semrel structures, we find that there is a **Part** relation between ***tree*** and ***leaf*** in a host of definitions for specific types of trees (e.g., in the definition of ***beech***: "a type of tree with smooth grey trunk, spreading branches, and dark green or copper-coloured leaves").

The full inversion of complete semrel structures in this work is unique in all documented DB research. Also distinctive is the richness of the semrel structures themselves, as demonstrated by the structure for a definition of ***market*** in Figure 5.4. The massive LKB network constructed for this dissertation invalidates the criticism leveled against DB methods by Yarowsky (1992) and Ide and Veronis (1993) that LKBs created from MRDs provide spotty coverage of a language at best. The comprehensive coverage of the similarity procedure described in Chapter 4 is evidence that this criticism is simply not true for the methods and data used in this research.

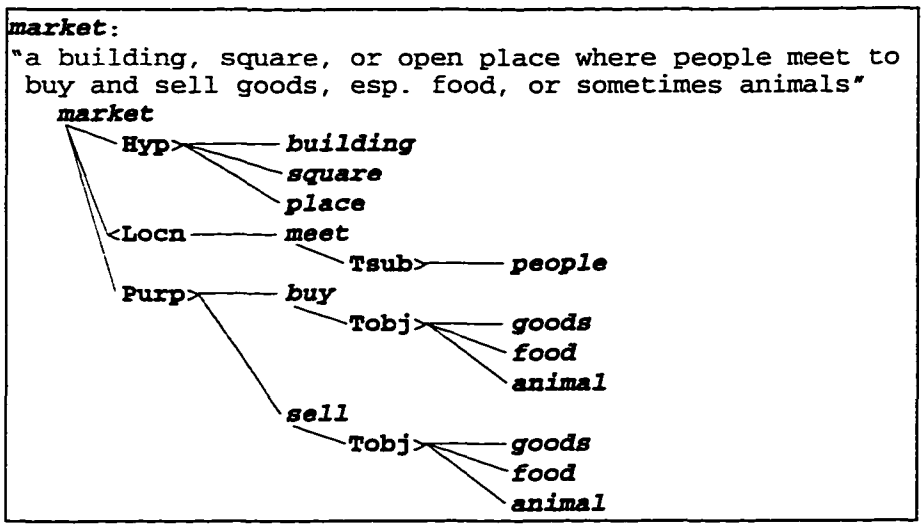

Figure 5.4 Semantic relation structure for a definition of **market**

5.1.5 Relatedness vs. Similarity

Although the term *similarity* has been used throughout this thesis, it is important here to distinguish between two types of similarity discussed in DB, EB, and CB research. They correspond to the two types of labeled relations discussed in Section 5.1.2 above and throughout Chapter 4.

The first is paradigmatic, or substitutional, similarity. Two words that are paradigmatically similar may be substituted for one another in a particular context. For example ***book*** may be substituted for ***magazine*** in the sentence ***I speed-read a magazine at the doctor's office***, and the two words are therefore paradigmatically similar. In this work, words that are similar in this way are often joined in the LKB by semrel paths containing paradigmatic relations (e.g., ***magazine***—**Hyp**→***book***). However, since they often appear in the same context, they may also be joined by symmetrical syntagmatic relations (e.g., ***book***←**Tobj**—***speed-read***—**Tobj**→***magazine***), as was observed in Chapter 4.

The second type is syntagmatic similarity. Two words that are syntagmatically similar may often occur together in the same context. For example, ***walk*** and ***park*** are syntagmatically similar in that they often occur together in a sentence such as ***I like to walk in the park***, but they can generally not be substituted for one another. In this research, words that are similar in this way are often joined in the LKB by semrel paths containing syntagmatic relations (e.g., ***walk*—Locn→*park***).

For the purposes of discussion in this and following sections, paradigmatic similarity will be the default meaning of the word *similarity* used herein, as it has been throughout this thesis. The phrase *syntagmatic similarity* will be used whenever there is a need to refer uniquely to this type of similarity, and the term *relatedness* will include both types of similarity.

It is interesting to note that in the majority of DB research, the two types of similarity discussed above have not been adequately distinguished. EB and CB research, however, have generally made this distinction. In this regard, the present research is more like the latter methodologies; the similarity procedure attempts to identify words that are paradigmatically similar and not those that are syntagmatically similar. For example, the similarity procedure identifies ***waiter*** and ***restaurant*** as being dissimilar (although there are many semrel paths between them that show them to be closely related). In the work by Kozima and Furugori (1993), however these words were expected to and did have a high similarity score, even though these researchers claimed to be focused on paradigmatic similarity.

The heuristic rules of Jensen and Binot (1987) for prepositional phrase attachment included a sort of binary similarity procedure, counting on the similarity of hypernyms as it searched for relations that could not be extracted directly from the definition of a given word, by looking at the definitions of its hypernyms. In this way, this early basis for the present research also depended on similarity to infer relations from multiple dictionary definitions.

DB researchers focusing on word sense disambiguation, on the other hand, have typically used methods that can only really determine relatedness. Lesk (1987) showed that by measuring the overlap (i.e., relatedness) between words in dictionary definitions and words in the context of a particular word to be disambiguated, a correct sense can be selected with some accuracy for a small sample. In the "statistical lexical disambiguation method" described by Wilks et al. (1992), a similar measurement of overlap was extended to identify words that were "semantically related" to the words in dictionary definitions. This relatedness factor was based on word co-occurrence as measured across all the words in all dictionary definitions. Matching contexts from input text to those in definitions was performed by vector similarity measurements (where the *similarity* of vectors in this case indicated *relatedness* of words). Braden-Harder (1993) also matched vectors representing the combined information from LDOCE dictionary entries (including subject codes, features, and predicate arguments extracted by parsing example sentences) with vectors representing the contexts of words in input text in order to identify word senses. Vector similarity methods have had a history of usefulness in determining document similarity in information retrieval systems as well (Salton 1983).

Because the neural-network structures built from MRD definitions (e.g., by Veronis and Ide 1990, Kozima and Furugori 1993, Niwa and Nitta 1994) have not contained labeled relations, they have only been able to determine word relatedness, like the work of Wilks et al. and Lesk above. (Braden-Harder, by using labeled relations, could distinguish actual similarity but did not.) This has nevertheless been adequate for the applications of these networks, mainly word sense disambiguation (Veronis and Ide 1990, Niwa and Nitta 1994), but also textual cohesion (Kozima and Furugori 1993).

The present research can also determine general relatedness for word sense disambiguation through the generation of sets of related words. This is accomplished by querying the semrel paths between a pair of words and then extracting the words contained in those paths as well as the words whose definitions were used to create the paths. These sets of related words are known as *path words* and *definition words*, respectively, and are shown for three of the paths between ***stock*** and ***soup*** in Figure 5.5. The path words shown in the figure do not include the source and target word for the sake of brevity. Note that semrels from two different definitions were used to construct each of the three (extended) paths, e.g., the second path consists of one semrel from a definition of ***polenta*** and another from a definition of ***won ton***.

Semrel paths	Path words	Definition words
stock—Hyp→*broth*—Hyp→*soup*	*broth*	*stock, broth*
stock←Locn—*boil*—Locn→*soup*	*boil*	*polenta, won ton*
stock←Tobj—*prepare*—Tobj→*soup*	*prepare*	*stockpot, soup*

Figure 5.5 Path words and definition words for a few of the semrel paths between **stock** *and* **soup**

Figure 5.6 shows clusters of words related to specific senses of the query words ***stock/soup***, ***stock/investment***, and ***stock/cattle***. These clusters were obtained by extracting path words and definition words from the top 20 paths between the query words. Sense-related clusters such as these are useful not only in word sense disambiguation, but in the expansion of queries for information retrieval systems as well.

stock ↔ soup
Path words: ***stock broth soup prepare cock-a-leekie veloute thicken boil vegetable situation food fish meat cook***
Definition words: ***broth stock soup cock-a-leekie veloute thicken roux file polenta won_ton boil Swiss_steak stew fishpond stockpot pastina dumpling***

stock ↔ investment
Path words: ***stock invest investment handle acquire good capital beta portfolio type hold company disinvestment furnish yield share sell divest make preferred_stock buy market diversify holding***
Definition words: ***invest investment stock handle raid good dollar_diplomacy beta portfolio unit_trust reserve run disinvestment yield shareholder acquire blind_pool market privilege put short stock_exchange stockbroker stop_order divest restock preferred_stock diversify holding***

stock ↔ cattle
Path words: ***stock farm cattle livestock keep buy breed punch sell animal raise race rustle***
Definition words: ***stock livestock repertory stockpot stockyard rodeo corral cattle call market pyramid right stock_exchange stockbroker market_town carry open_stock stockbreeder breed blank punch blind_pool privilege put short stop_order station ox anthrax ranch raise cattleman race rustle***

Figure 5.6 Related words from semrel paths between **stock** *and* **soup**, **stock** *and* **investment**, *and* **stock** *and* **cattle**

The words in these clusters exhibit a more general relatedness, as in the work cited above. The specificity of the relations in the LKB, however, enables a distinction

between this kind of relatedness and the substitutional similarity needed to infer relations for prepositional phrase attachment.

5.1.6 Related work using WordNet

Because of the difficulties encountered by some researchers in extracting large amounts of consistent semantic information from MRDs and the criticisms of DB methods cited in Section 5.1.4 above, WordNet (Miller et al. 1990) has begun to be used as an LKB for NL tasks such as word sense disambiguation. Although it has not been created directly from an online resource (and therefore suffers from the *knowledge acquisition bottleneck* as a methodology for creating other LKBs), dictionaries were certainly used by those who manually created it, and related work with WordNet is mentioned here because of its similarity to LKBs created by DB methods. Other relevant research involving WordNet and its integration with CB methods (e.g., by Resnik 1992, 1995) is discussed later, mainly in Section 5.3.3.

The main use of WordNet in word sense disambiguation has been to determine word similarity. Because the hand-coded relations in WordNet are paradigmatic, including **IsA** (**Hypernym/Hyponym**), **Part** (**Meronym/Holonym**), **Synonym**, and **Antonym**, its links really do indicate paradigmatic similarity between words, although this is not stressed by the researchers using it. Agirre and Rigau (1996), however, did criticize the lack of "cross-categorial semantic relations" (i.e., syntagmatic relations) in WordNet, saying that such relations would be a significant help in determining similarity.

The similarity metrics used by these researchers have tended to be based on the distance between words in the WordNet hierarchy. Li et al. (1995) stated that their

similarity score was inversely proportional to this distance. Sussna (1993) assigned weights to links in the hierarchy depending on the type of relation (e.g., the **Synonym** weight was 0, and the **Hypernym** weight ranged from 1-2) and then computed a weighted distance to indicate similarity. Agirre and Rigau (1996) used an interesting notion of "conceptual density" to determine similarity. This notion was based on comparing the number of senses of a word and the words in its context in input text occurring in the same WordNet sub-hierarchy with the total number of senses in that sub-hierarchy. In this way, the total of the distances between the word and its context words was being considered with a single measurement. Although the researchers cited have achieved good success with WordNet, using it as a richly enhanced thesaurus/taxonomy much like EB researchers use thesauri as described in the next section, the criticism regarding lack of syntagmatic relations is a valid one.

5.2 Example-based methods

Example-based (EB) methods have used example phrases or sentences taken from real text, represented either as strings or in some more structured form, to resolve ambiguities in NL analysis systems or to determine corresponding translations in machine translation (MT) systems (e.g., Sumita and Iida 1991, Jones 1992, Okumura et al. 1992, Watanabe 1992, Tsutsumi 1993, Cranias et al. 1994, Uramoto 1994, Utsuro et al. 1994). Principally, EB methods have included:

1. the creation of an example base as automatically as possible,

2. the matching of text to be processed (i.e., analyzed or translated) against the entries in the example base, including a provision for matching when an exact match does not exist, and

3. the adaptation of the output corresponding to the example that was matched in the example base.

The discussion that follows will focus on methods 1 and 2, which overlap in interesting ways with DB and CB methods, and will leave method 3, which is specific to EB processing, unexplored. In particular, method 1 corresponds to the creation of an LKB using DB methods, and method 2 relates to the determination of similarity found in both DB and CB methods. Matching examples when exact matches do not exist is equivalent to inferring non-existent relations in LKBs with DB methods and to parameter smoothing to overcome sparse data by using CB methods.

Many EB efforts have focused on analyzing Japanese (e.g., Sumita and Iida 1991, Okumura et al. 1992, Watanabe 1992, Utsuro et al. 1994), typically because recourse to lexical semantics is required earlier in this process than for many languages, even, for example, at the stage of determining word boundaries in input text. In the absence of sufficient structured semantic information, EB methods have filled an important gap, allowing resolution of ambiguity through recourse to a data base of examples collected from previously processed text.

Because of the syntactic and lexical disparity between English and Japanese, EB methods have also been used in the transfer phase of translation between these languages (e.g., Sumita and Iida 1991, Watanabe 1992, Utsuro et al. 1994). The complexity of the

transformations required between source and target structures has led these researchers to the more tractable approach of mapping from one complete structure to another through EB methods.

5.2.1 Example bases vs. LKBs

To date, the kind of information present in the example bases of documented EB systems has ranged from unprocessed strings (Sato 1991 and Furuse and Iida 1992), to simple syntactic patterns (Sumita and Iida 1991, Sato 1991, and Tsutsumi 1993), to deeper, semantic structures (Sadler 1989, Takeda et al. 1992, and Uramoto 1994). These deeper structures, which Sato (1991) and Sadler (1989) have claimed to be most desirable for inclusion in an example base, have often represented semantic relations which are almost identical to the semantic relations of this research.

Invariably, the deeper the processing involved to produce the examples, the more difficult it has been to obtain them. This has also been true of the DB extraction methods described above in Section 5.1.3. Sadler (1989) was able to populate the LKB in the DLT system with 76,000 examples by relying on the completely regular semantic case markings of Esperanto, something which truly natural languages do not exhibit. Takeda et al. (1992) used a parser together with interactive human verification to obtain 30,000 examples for the Shalt 2 translation system from definitions and example sentences in the *IBM Dictionary of Computing* and a small subset of LDOCE. The examples consisted of structures representing verbs with modifying prepositional phrases, not unlike an **Instrument** or **Location** relation between a verb and a noun in the present research.

Uramoto (1994) reported two years later that the number of examples in Shalt 2 had been increased to 57,000.

Many of the example bases described in the literature are quite small, containing from a few hundred to a few thousand examples, and while some have been constructed by semi-automatic means, most have employed some degree of manual crafting. The inability to automatically bootstrap more examples into example bases has been due to the lack of a parser that is not dependent on the examples being gathered. Some DB research (including this work) has not had this problem, since it has utilized an existing syntactic parser. In spite of this weakness in the creation of example bases, they have nevertheless served as valuable repositories of semantic information. And, if the semantic examples in some of them were further linked into network-like structures, they would become almost identical to some of the LKBs produced by DB methods.

5.2.2 Determining similarity paradigmatically vs. syntagmatically

The similarity that is the focus of this research is clearly lexical, dealing with relations between words. Some EB systems, however, have also dealt with structural similarity, especially those employed in the transfer phase of translation, as described in the introduction to EB methods above.

While the examples stored in example bases themselves have been generally syntagmatic (i.e., examples of strings, phrases, or syntactic or semantic relations taken from corpora), a separate resource of paradigmatic information, such as an online thesaurus, has often been used for the determination of similarity in EB methods. However, this has not always been the case since, as discussed in Section 5.1.5,

syntagmatic relations (in this case, examples) may also be used to indicate similarity between words if those words share the same relations. Therefore, in some documented EB systems, the examples themselves have participated in the determination of similarity.

To determine similarity paradigmatically, Sumita and Iida (1991), Furuse and Iida (1992), Tsutsumi (1993), and Utsuro et al. (1994) made use of existing online synonym (and in the Tsutsumi case, taxonym) dictionaries. A similarity score was computed between words based on their proximity in the synonym/taxonym hierarchies. An example of a hierarchy used by Sumita and Iida (1991) is shown in Figure 5.7. In the hierarchy, ***taizai*** (***stay***) is much more similar to ***touchaku*** (***arrive***) than ***kaigi*** (***conference***) is, because the distance through the hierarchy is considerably smaller between the first two words than it is between the first and last word.

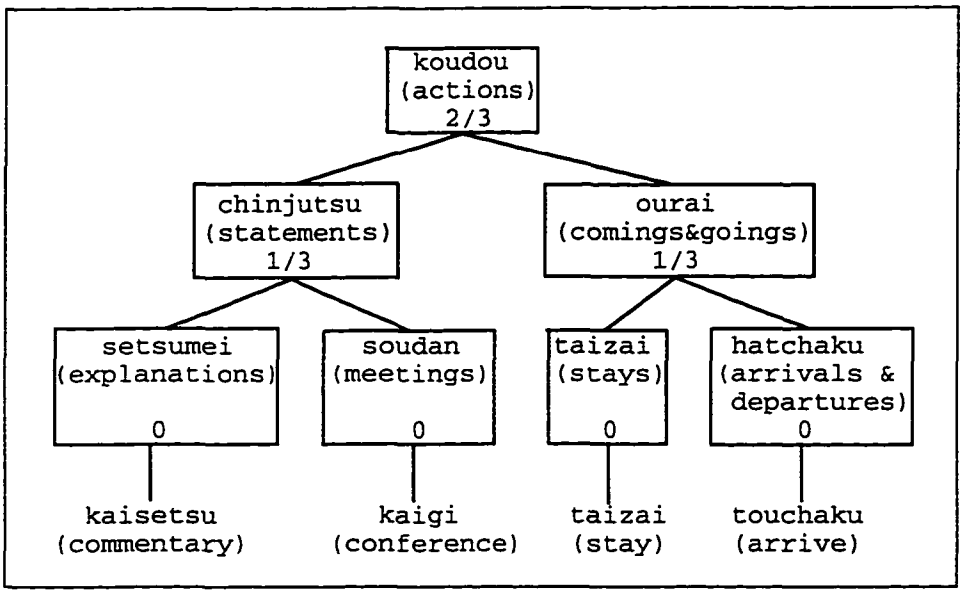

Figure 5.7 Thesaurus hierarchy with distances from leaf nodes, taken from Sumita and Iida (1991)

Relying on syntagmatic information, Sato (1991) used vector similarity measurements between the words in his example base to determine similarity

relationships that could then be applied to incoming text. These measurements were based on words in the syntagmatic contexts surrounding a given word in the example base. Sadler (1989) also used a computation that measured the overlap of contexts in his example base, but it used a proximity function based on simple counts instead of vectors.

The most interesting recent work in this area has been by Uramoto (1994), who described two enhancements made to an EB MT system to augment the determination of similarity performed with an online thesaurus. These enhancements were:

1. extracting about 700 "conjunctive relations" (which indicated "interchangeability" and therefore, similarity) from a corpus of computer manual text through pattern matching and using them, along with the synonymy relations in the thesaurus, to indicate similarity, and

2. creating a "context base," by superficially parsing the input text, which contained all the syntagmatic relations involving words that were not found in the thesaurus, to be used for determining similarity for those unfound words.

Clearly, these two enhancements targeted the addition of syntagmatic information to the paradigmatic information already used by the system in order to increase the coverage and accuracy of similarity determination. While the first enhancement provided an ad hoc incremental improvement to the information in the thesaurus, the second enhancement was a legitimate procedure that could also be used to augment the LKB of this research with relations for unfound words. Aside from the latter benefit, though, the LKB would probably not be helped much by the former type of improvement, since it already contains a vast amount of tightly and consistently integrated paradigmatic and

syntagmatic information. That is not to say, of course, that more information could not be added to the LKB from corpora other than dictionary definitions and example sentences. On the contrary, this is a planned extension for this work, but it will be performed with the same powerful set of parsing and extraction methods that were applied to the dictionary text.

5.3 Corpus-based methods

Corpus-based methods have extracted relations between words, often unlabeled word bigrams, but in some cases labeled triples as well, from large text corpora that usually contain millions of words through the use of statistical co-occurrence processing. They have generally been used to create statistical *language models* (i.e., arrays of statistical parameters that can be used to predict legitimate word combinations) for use in speech recognition (e.g., Jelinek 1990), but have fairly recently been applied to other NL tasks as well, such as MT (e.g., Brown et al. 1990) and word sense disambiguation (e.g., Brown et al. 1992, Resnik 1992, Yarowski 1992).

A common problem addressed in CB methods is the handling of sparse data in the parameter arrays, i.e., determining the probability of a word bigram or other similar event when it has not been seen in the training data. This is also known as the need for *parameter smoothing*. The most frequently used solution to this problem, which was first proposed by Katz (1987), is "backing off" to an independence assumption about the probabilities of the words in the unseen bigram. This means assuming that the probability of words w_1 and w_2 in the bigram are independent of one another and that

their combination, which would normally be $P(w_2|w_1)$ can be adequately estimated by the $P(w_2)$ alone, since the more frequent w_2 is, the higher $P(w_2|w_1)$ will be, regardless of w_1.

More recently, other proposals have been made to address the sparse data problem, including the use of class-based *n*-grams (Brown et al. 1992, Resnik 1992, Pereira 1993) and direct similarity determination (Dagan et al. 1993, Dagan et al. 1994, Grishman and Sterling 1994), which will be discussed below. At this point, it should be noted that the problem of handling sparse data in CB methods is identical in nature to handling the lack of exactly matching examples by EB methods or semantic relations by DB methods. The CB solutions to this problem are also directly comparable to using a similarity function to perform EB fuzzy matching or DB semrel inference.

5.3.1 Simple co-occurrence relations vs. labeled relation structures via parsing

The majority of CB work has focused on the identification of word bigrams in corpora, since researchers in this field have typically eschewed complex rule-based parsers in favor of seeing how much information they can extract automatically from raw (or tagged) textual data. The bigrams so extracted are of course syntagmatic in nature, but do not specify the type of relations between words, although one might be able to infer some relations in limited contexts (e.g., in sentence initial position, a word that can only be an adjective followed by one that can only be a noun probably modifies that noun). Paradigmatic similarity between words can also be inferred from the syntagmatic contexts, as described in Section 5.1.5, but the specific types of paradigmatic relations between words cannot be inferred from the corpus alone. Yarowsky (1992) successfully

used additional information about word classes from *Roget's International Thesaurus* to extract hypernyms automatically from an untagged corpus.

Other CB work has actually extracted syntactic or semantic triples (similar to the semrels of this work) from corpora through semi-automatic analysis relying on human verification (Alshawi and Carter 1994) and through automatic parsing (Grishman and Sterling 1992, 1994). Although this CB work has made the leap from co-occurrence to labeled relations, enabling the relations in the Grishman and Sterling case to be used as "selectional patterns" for guiding the formation of structures in a syntactic parser, the relations have still been only binary. Still lacking has been the extraction and utilization of more complex relational structures, such as the semrel structures of this research.

In CB work, syntagmatic relations (other than general relatedness) between words separated by some distance in text have been very hard, if not impossible, to identify (usually for want of a syntactic parser in such work), making the extraction of a syntagmatic relational structure containing multiple words equally unlikely. Those who have used additional hand-crafted resources such as a thesaurus (Yarowsky 1992) or WordNet (Resnik 1992) in order to extract paradigmatic relations from corpora, perhaps even within contexts several words wide, have been fairly successful, but they have still been precluded from obtaining more complex structures that would meaningfully relate the multiple words in those contexts simultaneously, as semrel structures do.

5.3.2 Word classes vs. direct similarity measurement

In order to overcome the sparse data problem described in the introduction of the section on CB methods above, a number of researchers have proposed substituting the

probabilities of word classes when specific instances of bigrams do not exist in the training corpus (e.g., Brown et al. 1992). The classes have been established by determining the co-occurrences of words in a text and then clustering those words with similar co-occurrence distributions. The co-occurrence probability of a given bigram that does not occur can then be estimated by the averaged co-occurrence probability of the classes of the words in the bigram. Because the number of classes established in this approach has usually been determined empirically, and has therefore been subject to variations which, depending on the training corpus, might lead to less accurate results, Resnik (1992) used information from the WordNet taxonomy to help determine the granularity of the classes, and Pereira (1993) based membership of a word in a particular class on probabilities.

The problem with the class-based approach, as pointed out by Dagan et al. (1993), is that the use of a predetermined number of word classes or clusters has the potential to cause information loss. They proposed a method that is based on directly measuring the similarity between two words instead of grouping words into similarity classes. They determined the similarity between two words by comparing vectors of mutual information scores for words that tend to co-occur with the two words being examined. This is not unlike the strategies of some DB and EB methods to compare the syntagmatic contexts surrounding words in order to determine similarity. In other work by Dagan et al. (1994), they retained the notion of direct similarity determination while replacing the use of mutual information scores with probability estimates, so that an actual co-occurrence probability estimate for unseen bigrams can be obtained.

Grishman and Sterling (1994) also rejected the class-based approach and adopted Dagan's same notion of direct similarity measurement in their work, which is perhaps the closest to the present research of all the CB research cited. They determined similarity between two words by estimating the probability that one word will appear in the same "selectional pattern" context as the other word. These selectional patterns are equivalent to some of the syntagmatic semantic relations of the present work and therefore constitute a valid context for the determination of paradigmatic similarity, as was described in Section 5.1.5. However, an important difference is that matching contexts of selectional patterns is only equivalent to checking the symmetrical syntagmatic semrel paths between words. While symmetrical paths are certainly strong indicators of similarity, Appendix A shows that there are many important asymmetrical ones as well. And of course, the use of paradigmatic relations in addition to syntagmatic ones is more significant still.

Grishman and Sterling (1994) provided one example of the output of their similarity computation, which was trained on a corpus of business text, shown in the left two columns of Table 5.1 below. The words in the first column are ranked according to their similarity score in relation to the word ***bond***. As evidence of the additional power for determining similarity that exists in the methods implemented in this work, the same words were submitted to the similarity procedure described in Chapter 4 (one at a time, together with the word ***bond***, *with no specific training for handling business text*) and the results, also ranked by similarity score are shown in the right two columns of Table 5.1. If one further assumes a similarity threshold of .0015 (represented by the dark horizontal line across the right two columns under the word ***maturity***), which was the one empirically determined to be the best in Chapter 4, the result is that most all of the noise

words in the Grishman and Sterling list, i.e., the words not intuitively similar to ***bond***, drop below the similarity threshold, matching human intuition. Possible exceptions are located just above the threshold: the words ***objective*** and ***maturity***. Even then, the latter is arguably similar syntagmatically, but not substitutionally.

Word	Grishman & Sterling score	Word	Similarity score (this work)
eurobond	0.133	mortgage	0.0280
foray	0.128	debenture	0.0242
mortgage	0.093	coupon	0.0196
objective	0.089	yield	0.0182
marriage	0.071	note	0.0168
note	0.068	commitment	0.0137
maturity	0.057	subsidy	0.0136
subsidy	0.046	marriage	0.0127
veteran	0.046	security	0.0118
commitment	0.046	eurobond	0.0056
debenture	0.044	issue	0.0037
activism	0.043	objective	0.0034
mile	0.038	maturity	0.0021
coupon	0.038	foray	0.0011
security	0.037	mile	0.0009
yield	0.036	activism	0.0000
issue	0.035	veteran	0.0000

Table 5.1 Similarity score adapted from Grishman and Sterling (1994) compared to similarity score described in Chapter 4 of this thesis

The evolution of CB methods to the use of direct similarity measurements in the handling of sparse data situations validates the methods used in this work to determine similarity and to infer semantic relations. It also demonstrates the convergence of DB, EB and CB methods in the handling of similar phenomena in natural language.

5.3.3 Related work using WordNet

Resnik (1992) has used the WordNet taxonomy to help determine the granularity of the similarity classes used for parameter smoothing, as described in the last section. He has also used probabilities derived from a corpus to determine a given word's *informativeness*, which is inversely proportional to its probability of occurrence in the corpus (Resnik 1995). This measure of informativeness was used to enhance his algorithm for determining similarity (for word sense disambiguation) by allowing him to identify the most informative (i.e., least frequently occurring in the corpus) subsumer, or ancestor, in the WordNet taxonomy of the two words being tested for similarity. This subsumer was then an indicator of the two words' similarity and also of their correct word sense.

While Resnik (1995) used information from a corpus somewhat indirectly to improve a paradigmatic similarity procedure, Hearst and Grefenstette (1992) used a CB similarity measurement, based on head-modifier pairs found in a corpus, to place new relations in WordNet synonym sets. Neither of these efforts really attempted to integrate directly into WordNet the syntagmatic relations that would increase its similarity determining power, as discussed in Section 5.1.6., and make it more like the LKB described in this research.

6. Conclusion

This research has focused on the automatic creation of a large-scale LKB and novel methods for accessing explicit and implicit information contained therein. The primary motivation for this work has been the desire to address the long-standing need for a truly broad-coverage semantic component in an NL understanding system. Such a component would have a long-term impact on the quality and coverage of NL applications across the board, including writing aids, machine translation, automatic summarization, information retrieval, data base query, command and control, speech recognition, and speech synthesis.

In many significant ways, this research has succeeded, and in others, it has only laid the foundation for important work yet to be done. The implementation described in this thesis possesses a combination of strengths that has not been seen in the literature, as well as a number of individually unique features. These features include averaged vertex probability weighting, creation of extended semrel paths, and the training and evaluation of the similarity procedure using an online thesaurus.

A list of the salient contributions of this research is provided below. The first contribution is the result of efforts by several members of the NLP group at Microsoft Research, as outlined in the introduction of this thesis. All the other contributions are specifically the work of the author, performed in the context of this research.

1. Automatic and reasonably accurate extraction of complex semantic relation structures, using a broad-coverage parser that produces logical forms,

from approximately 167,000 definitions and example sentences contained in two machine readable dictionaries, LDOCE and AHD3.

2. Completely inverted structures of labeled semantic relations, propagated across all the headwords of the dictionaries, forming a richly connected semantic network (not only a hierarchical taxonomy) consisting of approximately 870,000 inverted structures, which overcomes much of the spotty coverage of previous methods using MRDs.

3. Creation of extended semantic relation paths, using the structures from two different definitions and/or example sentences, specifying potentially complex relations between words in the LKB.

4. Computation of averaged vertex probability weights for individual semantic relations as well as for the paths created from them in the LKB.

5. A quantitatively-based similarity procedure, trained using over 100,000 word pairs from an online thesaurus and "anti-thesaurus," and based on both syntagmatic and paradigmatic semantic relations contained in the semantic relation paths obtained from the LKB.

6. A semantic relation inference procedure that is fully integrated with the similarity procedure, being based on the same mechanism for storing and accessing semantic relation paths in the LKB.

7. A procedure for obtaining sets of words closely related to the specific senses of a pair of query words, based on words (and definition headwords) taken from the semantic relation paths connecting the query words.

8. Thorough quantitative results indicating the breadth and accuracy of the similarity and inference procedures.

In addition to the specific contributions listed above, the analysis of dictionary-based (DB), example-based (EB), and corpus-based (CB) methods related to the methods used in this research revealed that these three methodologies have a number of characteristics in common. Shared characteristics include: 1.) the extraction of knowledge from repositories of NL texts, 2.) the central role of similarity determination, and 3.) a method based on similarity determination for inferring information that is not explicit in the NL text repositories. From the perspective of this research, these characteristics, as well as others from the DB, EB, and CB methodologies, may be synthesized into a unified framework.

Future work based on this research, some of which has already begun in the Microsoft NLP group, will include the following:

1. Work on automatic sense disambiguation of all the words in all the inverted semrel structures in the LKB, including the clustering of related senses (Dolan 1994). A first pass on this has already been made with the LDOCE-only version of the LKB and a subset of the combined AHD3-LDOCE LKB. This should provide a significant increase in the accuracy of the joins used to create extended semrel paths and should also enable further exploration of automatic

word sense disambiguation using the procedure for obtaining closely related words from semrel paths. Yet other enhancements to constraining the formation of extended semrel paths include analyzing the patterns of valid and invalid joins, possibly using quantitative training methods like those used to train the similarity procedure.

2. Use of similarity and semrel inference in a bootstrapping mode to extract more accurate semrels from the dictionary. Vanderwende (1995) has already proposed using multiple passes to accomplish more accurate extraction. This is especially important for improving the accuracy of semrels such as **Part** and **Material** (e.g., ***necklace of gold*** should yield the semrel ***necklace*—Matr→*gold***, but ***necklace of pearls*** should yield ***necklace*—Part→*pearls***).

3. Customization of LKB content for specific domains by processing corpora or domain-specific glossaries. Uramoto (1994) performed a type of limited customization to handle unknown words with his EB MT system. Customization based on the research of this thesis will involve applying the techniques used in semrel extraction to more general types of text (bridging from DB to CB methods). This has already been done to some extent in the processing of example sentences in dictionary entries. Specialized dictionaries and encyclopedic reference materials are also candidates for straightforward extraction.

Appendix A. Most frequent similarity path patterns

The following table contains the 168 highest frequency path patterns occurring in the LDOCE-based LKB that indicate similarity. This includes all path patterns with frequency greater than the frequency of the vertex of the hyperbolic curve representing the distribution of frequencies.

Path Patterns	Frequencies
Hyp HypOf	14239
Hyp HypOf Hyp	5596
HypOf Hyp HypOf	4891
HypOf Hyp	3453
Hyp	2916
HypOf	2755
TobjOf Tobj	2447
Tobj TobjOf	2339
Hyp Hyp	1928
Syn HypOf	1724
TobjOf HypOf	1691
HypOf HypOf	1640
Hyp Syn	1616
Hyp Tobj	1588
Hyp TobjOf	1245
Syn	1098
Tobj HypOf	1051
Tobj TobjOf Hyp	1021
Hyp Part	883
Syn Syn	852
PartOf HypOf	843
Hyp TobjOf Tobj	798
HypOf Tobj TobjOf	794
TobjOf Tobj HypOf	747
Hyp Locn	708
Hyp LocnOf	700
Purp HypOf	696
Locn LocnOf	694
PartOf Part	680
Syn HypOf Hyp	659
Means HypOf	657

Hyp PurpOf	639
Hyp TobjOf HypOf	638
Hyp MeansOf	622
Locn HypOf	614
LocnOf HypOf	613
TsubOf HypOf	604
Hyp Tobj HypOf	603
Hyp HypOf TsubOf	601
Syn Hyp	579
HypOf TobjOf Tobj	577
Hyp Tsub	575
Manner MannerOf Hyp	549
Hyp PartOf	549
HypOf Hyp HypOf Hyp	547
LocnOf Locn	537
Hyp Syn Syn	530
HypOf Hyp Syn	526
HypOf Syn	519
TsubOf Tsub	509
Part HypOf	503
Part Hyp HypOf	503
Tsub Hyp HypOf	500
LocnOf Tobj	489
TobjOf Tobj Hyp	488
Manner MannerOf	478
Syn Syn HypOf	444
HypOf TobjOf HypOf	441
Locn TobjOf	426
HypOf PartOf HypOf	424
Part PartOf	413
Tobj LocnOf	412
Hyp Tobj Hyp	411
TobjOf Locn	410
Tobj HypOf TsubOf	407
TsubOf Tobj	402
HypOf TobjOf	401
HypOf Hyp Hyp	395
Hyp HypOf Purp	393
Hyp Part Hyp	392
Tobj Hyp	381
HypOf HypOf Hyp	378
Tobj Hyp HypOf	374
Hyp HypOf TobjOf	373

LocnOf Hyp HypOf	365
TobjOf Tsub	363
Hyp Tobj TobjOf	361
Hyp HypOf PartOf	356
HypOf Manner MannerOf	355
Hyp Tobj TobjOf HypOf	348
TobjOf HypOf Hyp	346
HypOf Hyp Tobj	346
PurpOf Hyp HypOf	343
TobjOf Hyp	341
Purp PurpOf	341
Tobj HypOf Hyp	340
Hyp Part PartOf	340
Tsub Hyp TobjOf	339
Tobj TobjOf HypOf	334
PartOf TobjOf	330
PartOf HypOf Hyp	330
Part PartOf HypOf	329
Syn Syn Syn	326
HypOf Tobj	325
Tobj Part	323
HypOf Hyp TobjOf	323
Tsub Hyp HypOf TsubOf	315
Tobj Hyp TobjOf	298
HypOf Hyp Part	296
Hyp TsubOf	295
PartOf Part Hyp	288
Hyp HypOf Locn	287
Tsub TobjOf	281
Hyp Tsub Hyp	281
Hyp Means	276
Tobj HypOf TobjOf	273
Purp Tobj	272
Tobj TsubOf	271
Tsub TsubOf	267
Hyp PurpOf Hyp	266
Part Hyp	260
Means MeansOf	260
MeansOf Hyp HypOf	259
TobjOf PurpOf	253
HypOf Purp HypOf	251
HypOf PartOf	251
Hyp LocnOf Locn	251

HypOf TsubOf HypOf	250
Tsub HypOf	244
Hyp TobjOf Hyp	238
HypOf Part	237
Hyp HypOf Means	234
MeansOf HypOf	233
LocnOf Hyp	231
HypOf PartOf Part	224
Purp HypOf Hyp	221
PartOf Hyp HypOf	221
Hyp Purp	219
PartOf Hyp	216
Hyp MeansOf Hyp	214
HypOf Locn	213
HypOf Tobj HypOf	212
Hyp Tsub TsubOf	210
Locn HypOf Hyp	206
Hyp HypOf Quesp	203
TobjOf	200
PartOf LocnOf	200
Hyp LocnOf Hyp	200
HypOf LocnOf	199
TobjOf Syn	197
Locn LocnOf Hyp	192
HypOf TsubOf	192
TsubOf HypOf Hyp	191
LocnOf Locn HypOf	190
Quesp HypOf	189
Syn Tobj	188
HypOf TsubOf Tsub	188
Part	187
HypOf Purp	187
Means TobjOf	185
HypOf Hyp LocnOf	185
HypOf Means	184
Hyp PurpOf Purp	184
TsubOf Tobj HypOf	183
Locn Hyp	182
Hyp HypOf Part	182
HypOf Locn HypOf	181
Tobj	179
TsubOf Tsub Hyp	177
HypOf Means HypOf	174

MeansOf Hyp	172
Hyp Tobj HypOf Hyp	172
Hyp QuespOf	172
QuespOf Hyp HypOf	170
Tsub TsubOf HypOf	169
TobjOf Hyp HypOf	169
Means HypOf Hyp	169
Locn Part	169

Appendix B. Similarity/dissimilarity test data

The following random 200 word pairs were given to human evaluators to assess the accuracy of the thesaurus and anti-thesaurus data, and consequently, to assess the performance of the similarity procedure. The indication of similarity (s) or dissimilarity (d) given in the first column for each word pair indicates whether the pair came from the thesaurus or anti-thesaurus, respectively. These indications served as the baseline against which both the human responses as well as the similarity procedure output were measured.

Similarity/ dissimilarity	First word	Noun/Verb	Second word
d	lamasery	N	abutment
s	factor	N	administrator
d	pasturage	N	admonition
s	sanctuary	N	adytum
d	gradient	N	allotment
s	proportion	N	arrangement
s	encouragement	N	assistance
d	infliction	N	attacker
d	idealist	N	aviation
d	janitor	N	bark
s	footing	N	basis
d	mansion	N	batsman
s	sire	N	begetter
s	ribbon	N	belt
d	object	N	blend
s	hop	N	bound
s	receptacle	N	box
d	laxity	N	brace
d	glory	N	braggart
d	once-over	N	brasserie
d	palate	N	brocade
s	grass	N	bulrush
d	hardihood	N	carrion
d	humanitarian	N	cease-fire

d	mail	N	chastisement
d	lenity	N	civility
s	fullness	N	completion
s	shape	N	condition
s	regularity	N	conformity
d	locale	N	connection
s	husband	N	consort
s	quarrel	N	controversy
d	parasol	N	coppice
s	ejaculation	N	copulation
d	impoliteness	N	core
d	midst	N	crowd
d	list	N	cultivator
d	incubator	N	cuticle
d	mockery	N	deal
d	lowland	N	decease
s	pretension	N	demand
s	purpose	N	direction
d	kind	N	discount
d	marksman	N	dismay
s	revelation	N	divulgement
s	emission	N	ejection
d	grunt	N	employment
d	masterstroke	N	equivocation
d	inspection	N	euphemism
d	inundation	N	exuberance
d	journey	N	eyewash
s	school	N	faction
d	osteopath	N	fellow
d	notable	N	footwork
d	participant	N	forgiveness
d	pentagram	N	gate
s	gift	N	gratuity
d	gape	N	gravestone
d	mammy	N	greenback
s	fatness	N	grossness
s	evidence	N	grounds
s	powder	N	gunpowder
s	rapidity	N	haste
s	pull	N	haul
d	narrator	N	heating
s	scabbard	N	holster
d	lyricist	N	hotbed

s	friction	N	impedance
s	harvest	N	ingathering
s	ruler	N	king
d	melee	N	leech
s	epic	N	legend
d	injustice	N	liberal
d	pavement	N	looks
d	mule	N	molest
d	intended	N	mooncalf
s	privation	N	need
s	gossip	N	news
s	responsibility	N	obligation
s	family	N	offspring
s	sentiment	N	opinion
s	reading	N	oration
s	form	N	outline
s	saddle	N	packsaddle
s	relish	N	partiality
s	play	N	pastime
s	providence	N	predetermination
d	playboy	N	presbyter
s	fellow	N	professor
s	hill	N	prominence
s	predestination	N	providence
d	hellcat	N	quadrangle
s	quirk	N	quibble
s	eccentric	N	quiz
d	look	N	rampart
d	jiffy	N	recourse
d	outrider	N	responsibility
d	lariat	N	reverberation
d	industrialist	N	revetment
s	fight	N	riot
d	imitation	N	ruler
d	pedestal	N	running
d	merriment	N	rustling
d	hothead	N	sack
d	newsmonger	N	scoop
s	flavor	N	seasoning
d	kayak	N	self
s	segregation	N	separation
s	slip	N	sheet
d	leaven	N	shoal

d	gong	N	sire
d	panorama	N	skyscraper
s	dust	N	soot
s	essence	N	spirits
s	fracture	N	sprain
s	politician	N	statesman
d	gun	N	strife
s	refinement	N	suaveness
d	maze	N	swish
d	irrelevance	N	tally
d	hatred	N	tandem
d	ocean	N	technique
d	mould	N	tegument
d	oracle	N	tinkle
d	money	N	tootsie
d	nod	N	typescript
d	periphery	N	vet
s	showing	N	view
s	hum	N	whir
s	hag	N	witch
s	secede	V	abdicate
s	glorify	V	acclaim
s	hang	V	adorn
s	placard	V	announce
d	hew	V	bless
s	sign	V	bless
s	plug	V	boost
s	raise	V	build
d	hear	V	canker
d	lilt	V	capture
s	gather	V	conclude
d	miscarry	V	concrete
d	interplay	V	condemn
s	groan	V	creak
s	promise	V	declare
d	hiss	V	disbar
s	exhibit	V	disclose
s	resent	V	dislike
s	emanate	V	emit
s	engrave	V	enchase
s	ignite	V	encourage
s	drop	V	end
d	moralize	V	enrich

d	gravel	V	entitle
s	hold	V	esteem
d	haggle	V	exert
s	explain	V	explicate
s	shine	V	face
s	skirt	V	flank
d	gesture	V	forebode
d	overlay	V	form
s	head	V	govern
d	grip	V	gravel
d	hurt	V	house
s	excite	V	inflame
s	enter	V	inject
d	knock	V	litter
s	heighten	V	loft
s	preponderate	V	outstrip
d	incarcerate	V	overturn
d	hoot	V	peak
d	fumigate	V	pet
s	profane	V	pollute
d	gladden	V	prize
d	liberate	V	program
s	posit	V	pronounce
d	gaze	V	recuperate
s	recover	V	repossess
s	impeach	V	reprehend
s	settle	V	reside
s	gain	V	secure
s	fire	V	shoot
s	scream	V	squeal
d	misread	V	stick
s	roam	V	stray
s	flow	V	stream
d	gabble	V	sully
s	repress	V	suppress
s	guess	V	surmise
d	hollow	V	taboo
d	fritter	V	taper
d	nullify	V	transfix
d	neglect	V	unfit
d	handclap	V	untie
d	mine	V	vacillate
d	oxygenate	V	varnish

s	rotate	V	whirl
d	muster	V	withdraw
s	extort	V	wrest
s	render	V	yield

Appendix C. Questionnaire for semrel data

Responses to the following questionnaire were obtained from 17 individuals. These responses were then translated into simple semrel representations which were used in the evaluation of the inference procedure. For example, for the first question, one response was "I walk in the park." The semrel representation for this response was ***walk*—Locn→*park***. Questions 1-6 solicit responses used to create **Locn** (questions 1 and 2), **Matr**, **Means**, **Tsub**, and **Tobj**, respectively.

1. Where do you typically do an activity? E.g., you might

travel to a country
travel in a car
sell things in a store

2. Where do you typically find a thing? E.g., you might find

pigs in a pen
coffee in a mug
bullet in a rifle

3. What is something typically made of? E.g.,

necklaces can be made of beads
tires are made of rubber
books are made of pages

4. What do you use to do an activity? E.g., you might

draw with a pen
cut with a knife
row with oar

5. Who typically does some activity? E.g.,

people dance
librarians find information
horses graze / eat grass

6. What is the typical object of some verb/activity? E.g., you might

read letters
watch movies
buy groceries

Appendix D. Semrel inference test results

The following semrels were created by propagating ***bake*** as the word in the first position of each **Tobj** semrel that was created from responses to the questionnaire in Appendix C. The weights in the first column were obtained using the inference procedure and the semrels were then sorted by those weights.

Weight	Semrel
5.12860E-06	***bake*—Tobj→*cake***
5.12860E-06	***bake*—Tobj→*food***
1.68270E-07	***bake*—Tobj→*clay***
1.01620E-08	***bake*—Tobj→*dish***
8.37530E-09	***bake*—Tobj→*cookie***
8.57040E-11	***bake*—Tobj→*car***
1.96790E-11	***bake*—Tobj→*tree***
6.96630E-12	***bake*—Tobj→*water***
3.59750E-12	***bake*—Tobj→*gas***
2.13800E-12	***bake*—Tobj→*wood***
1.32740E-12	***bake*—Tobj→*game***
7.53360E-13	***bake*—Tobj→*tear***
7.37900E-13	***bake*—Tobj→*dog***
7.37900E-13	***bake*—Tobj→*boat***
7.36210E-13	***bake*—Tobj→*dinner***
7.36210E-13	***bake*—Tobj→*lunch***
7.36210E-13	***bake*—Tobj→*garlic***
7.36210E-13	***bake*—Tobj→*kiss***
3.74970E-13	***bake*—Tobj→*beer***
1.52410E-13	***bake*—Tobj→*letter***
1.03750E-13	***bake*—Tobj→*time***
7.99830E-14	***bake*—Tobj→*death***
5.49540E-14	***bake*—Tobj→*candle***
4.44630E-14	***bake*—Tobj→*card***
1.18850E-14	***bake*—Tobj→*drum***
1.18850E-14	***bake*—Tobj→*cigar***
1.09650E-14	***bake*—Tobj→*music***
4.69890E-15	***bake*—Tobj→*tax***

3.66440E-15	***bake*—Tobj→*song***
3.31890E-15	***bake*—Tobj→*flower***
2.03700E-15	***bake*—Tobj→*clock***
7.69130E-16	***bake*—Tobj→*lawn***
4.84170E-16	***bake*—Tobj→*fever***
4.28550E-16	***bake*—Tobj→*appointment***
3.63080E-16	***bake*—Tobj→*scotch***
3.62240E-16	***bake*—Tobj→*movie***
1.71400E-16	***bake*—Tobj→*canoe***
0	***bake*—Tobj→*grocery***
0	***bake*—Tobj→*closet***
0	***bake*—Tobj→*chess***
0	***bake*—Tobj→*bicycle***
0	***bake*—Tobj→*bike***
0	***bake*—Tobj→*sauce***
0	***bake*—Tobj→*lettuce***

The following semrels were created by propagating ***write*** as the word in the first position of each **Means** semrel that was created from responses to the questionnaire in Appendix C. The weights in the first column were obtained using the inference procedure and the semrels were then sorted by those weights.

Weight	Semrel
1.52410E-07	***write*—Means→*brush***
6.18020E-11	***write*—Means→*crayon***
1.42230E-12	***write*—Means→*whisk***
1.22460E-12	***write*—Means→*pen***
7.70900E-13	***write*—Means→*pencil***
6.99840E-13	***write*—Means→*computer***
5.99790E-14	***write*—Means→*parachute***
3.06900E-14	***write*—Means→*racket***
2.37140E-14	***write*—Means→*pan***
1.86640E-14	***write*—Means→*knife***
9.44060E-15	***write*—Means→*typewriter***
3.77570E-15	***write*—Means→*towel***

1.67880E-15	*write*—Means→*razor*
1.67880E-15	*write*—Means→*regulator*
1.67880E-15	*write*—Means→*shovel*
1.67880E-15	*write*—Means→*spatula*
9.79490E-16	*write*—Means→*fork*
5.61050E-16	*write*—Means→*sponge*
3.41980E-16	*write*—Means→*spoon*
6.26610E-17	*write*—Means→*colander*
0	*write*—Means→*brain*
0	*write*—Means→*cane*
0	*write*—Means→*car*
0	*write*—Means→*chop stick*
0	*write*—Means→*club*
0	*write*—Means→*dispenser*
0	*write*—Means→*eye*
0	*write*—Means→*foot*
0	*write*—Means→*hand*
0	*write*—Means→*head*
0	*write*—Means→*heart*
0	*write*—Means→*key*
0	*write*—Means→*keyboard*
0	*write*—Means→*leg*
0	*write*—Means→*lung*
0	*write*—Means→*money*
0	*write*—Means→*mouth*
0	*write*—Means→*oar*
0	*write*—Means→*pogo stick*
0	*write*—Means→*saw*
0	*write*—Means→*tank*
0	*write*—Means→*volleyball*
0	*write*—Means→*weight*
0	*write*—Means→*wing*

References

Agirre, E., and G. Rigau. 1996. Word sense disambiguation using conceptual density. In *Proceedings of COLING96*, 16-22.

Ahlswede, T., and M. Evens. 1988. Parsing vs. text processing in the analysis of dictionary definitions. In *Proceedings of the 26th Annual Meeting of the ACL*, 217-224.

Alshawi, H. 1989. Analyzing the dictionary definitions. In *Computational lexicography for natural language processing*, ed. B. Boguraev and E. Briscoe, 153-169. London: Longman Group.

Alshawi, H., B. Boguraev, and D. Carter. 1989. Placing the dictionary on-line. In *Computational lexicography for natural language processing*, ed. B. Boguraev and E. Briscoe, 41-63. London: Longman Group.

Alshawi, H., and D. Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics* 20, no. 4:635-648.

Amsler, R. 1980. The structure of the Merriam Webster Pocket Dictionary. Ph.D. diss., University of Texas, Austin.

Barrière, C., and F. Popowich. 1996. Concept clustering and knowledge integration from a children's dictionary. In *Proceedings of COLING96*, 65-70.

Bookman, L. 1994. *Trajectories through knowledge space: A dynamic framework for machine comprehension.* Boston, MA: Kluwer Academic Publishers.

Braden-Harder, L. 1993. Sense disambiguation using an online dictionary. In *Natural language processing: The PLNLP approach*, ed. K. Jensen, G. Heidorn, and S. Richardson, 247-261. Boston, MA: Kluwer Academic Publishers.

Briscoe, E. 1992. ACQUILEX II: The acquisition of lexical knowledge. In *Synopses of American, European, and Japanese projects presented at the international projects day at COLING 1992*, ed. G. Varile and A. Zampolli, 3-6. Pisa: Giardini Editori.

Brown, P., J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics* 16, no. 2:79-85.

Brown, P., V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18, no. 4:467-479.

Bruce, R., and L. Guthrie. 1992. Genus disambiguation: A study in weighted preference. In *Proceedings of COLING92*, 1187-1191.

Byrd, R., N. Calzolari, M. Chodorow, J. Klavans, M. Neff, and O. Rizk. 1987. Tools and methods for computational linguistics. *Computational Linguistics* 13, no. 3-4:219-240.

Chakravarthy, A. 1994. Representing information need with semantic relations. In *Proceedings of COLING94*, 737-741.

Chodorow, M., R. Byrd, and G. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd Annual Meeting of the ACL*, 299-304.

Church, K., and P. Hanks. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Meeting of the ACL*, 76-83.

Copestake, A. 1990. An approach to building the hierarchical element of a lexical knowledge base from a machine-readable dictionary. In *Proceedings of the First International Workshop on Inheritance in Natural Language Processing* (Tilburg, The Netherlands), 19-29.

Cranias, L., H. Papageorgiou, and S. Piperidis. 1994. A matching technique in example-based machine translation. In *Proceedings of COLING94*, 100-104.

Dagan, I., S. Marcus, and S. Markovitch. 1993. Contextual word similarity and estimation from sparse data. In *Proceedings of the 31st Annual Meeting of the ACL*, 164-171.

Dagan, I., F. Pereira, and L. Lee. 1994. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd Annual Meeting of the ACL*, 272-278.

Dolan, W. 1994. Word sense ambiguation: Clustering related senses. In *Proceedings of COLING94*, 712-716.

Dolan, W., and S. Richardson. 1996. Not for its own sake: Knowledge as a byproduct of NLP. In *Proceedings of AAAI Fall Symposium* (Boston, MA), in preparation.

Dolan, W., L. Vanderwende, and S. Richardson. 1993. Automatically deriving structured knowledge bases from on-line dictionaries. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics* (Vancouver, Canada), 5-14.

Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19, no. 1:61-74.

Furuse, O., and H. Iida. 1992. An example-based method for transfer-driven machine translation. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation* (Montreal, Canada), 139-150.

Grishman, R., and J. Sterling. 1992. Acquisition of selectional patterns. In *Proceedings of COLING92*, 658-664.

———. 1994. Generalizing automatically generated selectional patterns. In *Proceedings of COLING94*, 742-747.

Guha, R., and D. Lenat. 1990. Cyc: A midterm report. *AI Magazine* 11:32-59.

Hearst, M., and G. Grefenstette. 1992. Refining automatically-discovered lexical relations: Combining weak techniques for stronger results. In *Statistically-Based Natural Language Programming Techniques, Papers from the 1992 AAAI Workshop* (Menlo Park, CA), 64-72.

Hindle, D. 1990. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting of the ACL*, 268-275.

Houghton Mifflin Company. 1992. *The American Heritage Dictionary of the English Language, third edition*, ed. A. Soukhanov. Boston, MA: Houghton Mifflin Company.

Ide, N., and J. Veronis. 1993. Extracting knowledge bases from machine-readable dictionaries: Have we wasted our time? In *Proceedings of KB&KS '93* (Tokyo), 257-266.

Jelinek, F. 1990. Self-organized language modeling for speech recognition. In *Readings in speech recognition*, ed. A. Waibel and K. Lee, 450-506. San Mateio, CA: Morgan Kaufmann Publishers.

Jensen, K. 1993a. PEG: The PLNLP English grammar. In *Natural language processing: The PLNLP approach*, ed. K. Jensen, G. Heidorn, and S. Richardson, 29-45. Boston, MA: Kluwer Academic Publishers.

———. 1993b. PEGASUS: Deriving argument structures after syntax. In *Natural language processing: The PLNLP approach*, ed. K. Jensen, G. Heidorn, and S. Richardson, 203-214. Boston, MA: Kluwer Academic Publishers.

Jensen, K., and J.-L. Binot. 1987. Disambiguating prepositional phrase attachments by using on-line dictionary definitions. *Computational Linguistics* 13, no. 3-4:251-260.

Jensen, K., G. Heidorn, and S. Richardson. 1993. Introduction to the PLNLP approach. In *Natural language processing: The PLNLP approach*, ed. K. Jensen, G. Heidorn, and S. Richardson, 1-11. Boston, MA: Kluwer Academic Publishers.

Jones, D. 1992. Non-hybrid example-based machine translation architectures. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation* (Montreal, Canada), 163-171.

Katz, S. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing* 35, no. 3:400-401.

Klavans, J., M. Chodorow, and N. Wacholder. 1990. From dictionary to knowledge base via taxonomy. In *Proceedings of the 4th Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary: Electronic Text Research*, 110-132.

Kozima, H., and T. Furugori. 1993. Similarity between words computed by spreading activation on an English dictionary. In *Proceedings of the 6th Conference of the European Chapter of the ACL*, 232-239.

Kozima, H., and A. Ito. 1995. Context-sensitive measurement of word distance by adaptive scaling of a semantic space. In *Proceedings of RANLP-95* (Bulgaria), 161-168.

Lesk, M. 1987. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986 ACM SIGDOC conference*, 24-26.

Li, X., S. Szpakowicz, and S. Matwin. 1995. A WordNet-based algorithm for word sense disambiguation. In *Proceedings of IJCAI'95*, 1368-1374.

Longman. 1978. *Longman Dictionary of Contemporary English*, ed. P. Proctor. London: Longman Group.

Luhn, H. 1958. The automatic creation of literature abstracts. In *IBM Journal of Research and Development* 2, no. 2:159-165.

Magerman, D., and M. Marcus. 1990. Parsing a natural language using mutual information statistics. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 984-989.

Markowitz, J., T. Ahlswede, and M. Evens. 1986. Semantically significant patterns in dictionary definitions. In *Proceedings of the 24th Annual Meeting of the ACL*, 112-119.

Microsoft. 1996. Natural Language Processing. Home page on the Internet at http://research.microsoft.com/research/nlp. Redmond: Microsoft Corporation.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to WordNet: an on-line lexical database. In *International Journal of Lexicography* 3, no. 4:235-244.

Montemagni, S., and L. Vanderwende. 1992. Structural patterns vs. string patterns for extracting semantic information from dictionaries. In *Proceedings of COLING92*, 546-552.

Niwa, Y., and Y. Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of COLING94*, 304-309.

Okumura, A., K. Muraki, and K. Yamabana. 1992. A pattern-learning based, hybrid model for the syntactic analysis of structural relationships among Japanese clauses. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation* (Montreal, Canada), 45-54.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the ACL*, 183-190.

Resnik, P. 1992. WordNet and distributional analysis: A class-based approach to lexical discovery. In *Statistically-Based Natural Language Programming Techniques, Papers from the 1992 AAAI Workshop* (Menlo Park, CA), 48-56.

———. 1995. Disambiguating noun groupings with respect to WordNet senses. In *Proceedings of the Third Workshop on Very Large Corpora*, 54-68.

Richardson, S., L. Vanderwende, and W. Dolan. 1993. Combining dictionary-based and example-based methods for natural language analysis. In *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation* (Kyoto, Japan), 69-79.

Sadler, V. 1989. *Working with analogical semantics: Disambiguation techniques in DLT.* Dordrecht, The Netherlands: Foris Publications.

Salton, G., J. Allan, and C. Buckley. 1994. Automatic structuring and retrieval of large text files. *Communications of the ACM* 37, no. 2:97-108.

Salton, G., and M. McGill. 1983. *Introduction to modern information retrieval.* New York: McGraw-Hill Book Company.

Sato, S. 1991. Example-based translation approach. In *Proceedings of the International Workshop on Fundamental Research for the Future Generation of Natural Language Processing,* ATR Interpreting Research Laboratories, 1-16.

Spark Jones, K. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28, no. 1:11-21.

Sumita, E., and H. Iida. 1991. Experiments and prospects of example-based machine translation. In *Proceedings of the 29th Annual Meeting of the ACL*, 185-192.

Sussna, M. 1993. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Management*, 67-74.

Takeda, K., N. Uramoto, T. Nasukawa, and T. Tsutsumi. 1992. Shalt 2—a symmetric machine translation system with conceptual transfer. In *Proceedings of COLING92*, 1034-1038.

Tsutsumi, T. 1993. Word sense disambiguation by examples. In *Natural language processing: The PLNLP approach*, ed. K. Jensen, G. Heidorn, and S. Richardson, 263-272. Boston, MA: Kluwer Academic Publishers.

Uramoto, N. 1994. A best-match algorithm for broad-coverage example-based disambiguation. In *Proceedings of COLING94*, 717-721.

Utsuro, T., K. Uchimoto, M. Matsumoto, and M. Nagao. 1994. Thesaurus-based efficient example retrieval by generating retrieval queries from similarities. In *Proceedings of COLING94*, 1044-1048.

Vanderwende, L. 1990. Using an on-line dictionary to disambiguate verbal phrase attachment. In *Proceedings of the 2nd IBM Conference on NLP*, 347-359.

———. 1995. Ambiguity in the acquisition of lexical information. In *Proceedings of the AAAI 1995 Spring Symposium Series, working notes of the symposium on representations and acquisition of lexical knowledge*, 174-179.

———. 1996. The analysis of noun sequences using semantic information extracted from on-line dictionaries. Ph.D. diss., Georgetown University, Washington, DC.

Veronis, J., and N. Ide. 1990. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of COLING90*, 289-295.

Vossen, P. 1991. Converting data from a lexical database to a knowledge base. ESPRIT BRA-3030 ACQUILEX working paper, no. 027.

Vossen, P., and A. Copestake. 1994. Untangling definition structure into knowledge representation. In *Inheritance, defaults and the lexicon*, ed. E. Briscoe, A. Copestake, and V. de Paiva. Cambridge: Cambridge University Press.

Watanabe, H. 1992. A similarity-driven transfer system. In *Proceedings of COLING92*, 770-776.

Wilks, Y., D. Fass, C. Guo, J. McDonald, T. Plate, and B. Slator. 1989. A tractable machine dictionary as a resource for computational semantics. In *Computational lexicography for natural language processing*, ed. B. Boguraev and E. Briscoe, 193-228. London: Longman Group.

———. 1992. Providing machine tractable dictionary tools. In *Semantics and the Lexicon*, ed. J. Pustejovsky, 99-154. Boston, MA: Kluwer Academic Publishers.

Wilks, Y., B. Slator, and L. Guthrie. 1996. *Electric words: Dictionaries, computers, and meanings*. Cambridge, MA: The MIT Press.

Yarowsky, D. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of COLING92*, 454-460.

Zipf, G. 1949. *Human behavior and the principle of least effort*. Reading, MA: Addison-Wesley Press.