PhD Dissertation

International Doctorate School in Information and Communication Technologies

DIT - University of Trento

An Organizational Approach to the Polysemy Problem in WordNet

Abed Alhakim Freihat

Advisor:

Prof. Fausto Giunchiglia

Università degli Studi di Trento

April 2014

Abstract

Polysemy in WordNet corresponds to various kinds of linguistic phenomena that can be grouped into five classes. One of them is homonymy that refers to the cases, where the meanings of a term are unrelated, and three of the classes refer to the polysemy cases, where the meanings of a term are related. These three classes are specialization polysemy, metonymy, and metaphoric polysemy.Another polysemy class is the compound noun polysemy.

In this thesis, we focus on compound noun polysemy and specialization polysemy. Compound noun Polysemy corresponds to the cases, where we use the modified noun to refer to a compound noun. Specialization polysemy is a type of related polysemy referring to the polysemy cases, when a term is used to refer to either a more general meaning or a more specific meaning. Compound noun polysemy and specialization polysemy in WordNet are considered the main reasons behind the highpolysemous nature of WordNet that make WordNet redundant and too fine grained for natural language processing.

Another problem in WordNet is its polysemy representation. WordNet represents the polysemous terms by capturing the different meanings of them at lexical level but without giving emphasis on the polysemy classes these terms belong to.

The highpolysemous nature and the polysemy representation in WordNet affect the usability of it as suitable knowledge representation resource for

natural language processing applications. In fact, the polysemy problem in WordNet is a challenging problem for natural language processing applications, especially in the field of information retrieval and semantic search.
To solve this problem, many approaches have been suggested. Although all the state of the art approaches are good to solve the polysemy problem partially, they do not give a general solution for it.
In this thesis, we propose a novel approach to solve the compound noun and specialization polysemy problem in WordNet in the case of nouns.
Solving the compound noun polysemy and the specialization polysemy problem is an important step that enhances the usability of WordNet as a knowledge representation resource.
The proposed approach is not an alternative to the existing approaches. It is a complementary solution for the state of the art approaches especially the systematic polysemy approaches.

Keywords

[WordNet, polysemy, compound noun polysemy, specialization polysemy]

Acknowledgments

I would like to thank my Advisor prof. Fausto Giunchiglia for every thing, especially for the patience, advice, teaching and believing in me. Without his support, inspiration, and encouragement it would have been impossible to finish this thesis.

I am also thankful to Biswanath Dutta for his contribution, collaboration and his continuous feedback.

I am also thankful to Giulia Zordan, Raffaele Guarasci, Ahmed Abu Raad, and Ahmed Fadhil for their contribution in the approach evaluation.

I am also thankful to my colleagues Vincenzo Maltese, Feroz Farazi, Mattia Fumagalli, Mohammad Abu Garib, Moaz Riyad, Fahed Alkhabbas and Kamal Badawi for their encouragement and support.

Finally, I would like to express my deep gratitude to my Wife Suzan for the tremendous support and inspiration which she gave me all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Natural languages are polysemous in nature. Any language contains terms that refer to more than one meaning. Polysemy [1] in natural languages corresponds to various kinds of linguistic phenomena and can be grouped in various polysemy classes . These classes are homonymy [2] which refers to the cases, where the meanings of a polysemous term are unrelated, and three classes that refer to the polysemy cases, where the meanings of a polysemous term are related [3]. These classes are specialization polysemy [4], metonymy [5], and metaphoric polysemy [6] [7]. Another form of polysemy is the compound noun polysemy [8] that refers to the cases, where we use the modified noun to refer to a compound noun.

1.1 The Problem

WordNet [9] represents the polysemous terms by capturing the different meanings of these terms at lexical level, but without giving emphasis on the polysemy classes these terms belong to [10]. In addition, WordNet contains too many cases of redundancy [11], too fine grained senses[12] [13], and sense enumerations [14] that make WordNet highpolysemous [15] [16] [17]. The lack of information regarding the polysemy types of the polysemous terms [18] and the highpolysemous nature of WordNet [19] affect its usability as suitable knowledge representation resource for Natural language processing (NLP) [20], especially Information Retrieval (IR) [21] and semantic search [22].

In the last, decades many approaches have been introduced to solve the polysemy problem through merging the similar meanings of polysemous terms [23] [24] [25]. These approaches are sometimes helpful in cases, where terms have meanings that are similar enough, need to be merged. However, merging polysemous terms with similar meanings is a sub-case of the solution of specialization polysemy [26]. In fact, a significant portion of the polysemous senses should not be merged, as they are just similar in meaning and not redundant.

In another approach, CORELEX [3] has been introduced as an ontology of systematic polysemous nouns extracted from WordNet. Although, the suggested underspecification method in CORELEX reduces the high-polysemous nature in metonymy cases, it does not reduce the highpolysemous nature in other polysemy classes. In particular, it does not solve the metaphoric polysemy, specialization polysemy, and compound polysemy problems.

Similar to CORELEX, new regular polysemy approaches [27] [28] [4] that attempt to extract implicit semantic relations between the polysemous senses via regular structural patterns have been introduced. The basic idea in these approaches is that the implicit relatedness between the polysemous terms corresponds to variety of semantic relations. Extracting these relations and making them explicitly should improve wordNet [29] [27]. Although the semantic relation extraction approaches are good for discovering the relations between polysemous synsets in a reasonable amount of polysemy cases, these approaches are good to discover the relations between figurative polysemy (metonymy and metaphoric) cases only. However, they do not give a solution to reduce the highpolysemous nature of WordNet.

1.2 The Solution

In this thesis, we classify the polysemy problem in WordNet into two main problems:

1. **The problem of the highpolysemous nature of WordNet**: The highpolysemous nature of wordNet makes it very difficult to be used by NLP applications [30].

2. **The problem of unspecified information**: WordNet does not differentiate between the polysemy classes. Recognizing the polysemy class of a given polysemous term is essential for NLP [3].

Accordingly, we present a novel approach to reduce the highpolysemous nature of WordNet by solving the specialization polysemy and compound noun polysemy problems and solve the problem of unspecified information in the case of homonymy and metaphoric polysemy. Our approach has three phases organized as follows.

S1. **Reducing the highpolysemous Nature of WordNet**

S1.P1. **Solving the compound noun polysemy problem**

1. **Compound noun polysemy discovery**:
   In this phase, we discover the Compound noun polysemy cases by means of regular term patterns.
2. **Compound noun polysemy disambiguation**:
   In this phase, we disambiguate the polysemous terms of the identified cases.

S1.P2. **Solving the specialization polysemy problem**

1. **Specialization polysemy discovery**:
   In this phase, we discover the specialization polysemy cases by means of regular structural patterns. In addition, a subset of homonymy and metaphoric cases are discovered in this phase.
2. **Specialization polysemy organization**:
   In this phase, we organize the identified cases by means of regular synset patterns.

S2. **Solving the problem of unspecified information in WordNet**

1. **Homonymy and metaphoric polysemy discovery**:
   This phase is needed if we want to solve the homonymy and metaphoric cases only. In our approach, this phase is included in phase S1.P1.1.
2. **Homonymy and metaphoric polysemy organization**:
   In this phase, we explicitly annotate the discovered cases by means of two semantic relations `is_homograph` and `is_metaphor`.

Our approach does not solve the polysemy problem in metonymy cases, where the state of the art approaches [3] [27] [28] [4] offer good solutions to this problem. That means, the presented solution is not an alternative solution for the state of the art solutions. Our approach is a complementary solution for these solutions especially CORELEX. The basic idea in our solution is that metonymy, specialization polysemy and compound noun polysemy are responsible for the highpolysemous nature in WordNet. CORELEX reduces the high polysemy in WordNet in the case of metonymy. Complementary to CORELEX, our approach reduces the high polysemy in WordNet in the case of compound noun polysemy and specialization polysemy.

Our approach does not discover all homonymy and metaphoric polysemy cases in WordNet. Nevertheless, our approach identifies a reasonable amount of homonymy and metaphoric cases.

1.3 Structure of the Thesis

The thesis is organized in four parts:
I The Problem: This part contains three chapters. Chapter 2 is an overview of WordNet. In Chapter 3, we describe the various polysemy types in WordNet. In Chapter 4, we define the problem.
II State of the Art: This part contains Chapter 5 that describes the current approaches for solving the polysemy problem in WordNet.
III The Solution: This part contains 7 chapters. In Chapter 6, we give an overview of the proposed solution. In Chapter 7, we give an overview of the algorithms in S1 and S2. Chapter 8 contains the formal definitions for the data structures that we use in our approach. Chapter 9 describes the algorithm in S1.P1. In Chapter 10, we present the structural patterns discovery algorithm in S1.P2. In Chapter 11, we discuss the structural pattern classification. In Chapter 12, we explain the polysemy organization algorithm.
IV Results: This part contains two chapters. In Chapter 13, we discuss the results and evaluation of our approach. In Chapter 14, we conclude the thesis and describe our future research work.

Part I

The Problem

Chapter 2

WordNet

WordNet or Princeton WordNet is a machine readable online lexical database for the English language. Based on psycholinguistic principles, WordNet has been developed since 1985 by linguists and psycholinguists as a conceptual dictionary rather than an alphabetic one [31]. Since that time, several versions of WordNet have been developed. In this thesis, we are concerned with WordNet 2.1.

A word or lemma is the basic lexical unit in WordNet. In contrary to conventional dictionaries, WordNet classifies the words or lemmas based on the grammatical category of the words (part of speech) into `nouns`, `verbs`, `adjectives` and `adjectives`. For example, the word `love` belongs to two grammatical categories in WordNet: `love` as a noun and `love` as a verb. In this thesis, we use the notion term to refer to a word and its grammatical category.

Synset is the fundamental structure in WordNet. A synset in WordNet corresponds to a lexical concept, role, or to an instance of a lexical concept [32]. For example, `Einstein` is an instance, `person` is a lexical concept, and `physicist` is a role.

#1 `Einstein, Albert Einstein: physicist born in Germany.`

#1 `physicist: a scientist trained in physics.`

#1 `person, individual, someone, somebody, mortal, soul: a human being; "there was too much for one person to do".`

Notice that wordNet does not distinguish between lexical concepts and roles [33].
A synset consists of the following elements:

1. *Synset lemmas*
2. *Synset gloss*
3. *Synset relations*

In the following, we give an overview of these elements.

2.1 Synset Lemmas

Synset lemmas are synonymous terms that belong to the same grammatical category. WordNet considers two terms to be synonyms (denote the same concept) if they are exchangeable in some context [34]. For example, the nouns `love` and `passion` are exchangeable in the following two sentences.

`The theater was her first love.`

`He has a passion for cock fighting.`

WordNet organizes the relation between terms and synsets through senses (term synset pair). A term may have one or more senses. For example the term `man` has 11 senses.
An important issue related to synset synonyms in WordNet is the coverage issue. The coverage of WordNet is not complete as follows.

- *Missing terms*: WordNet contains synsets with missing terms [35]. For example, the term `brocket` denotes two synsets in wordNet:

  `#1 brocket: small South American deer with unbranched antlers.`

  `#2 brocket: male red deer in its second year.`

The synonyms of the two synsets are incomplete. The terms `red brocket` and `Mazama americana` which are synonyms of the terms in #2 are missing. The two synsets do not even include the term `brocket deer`[1].

- *Missing senses*: Despite the highpolysemous nature of wordNet, there are substantial amount of missing senses in WordNet [36]. For example, WordNet does not contain the following sense for the term `Folder`: `folder: a virtual container within a digital file system, in which groups of files and other folders can be kept and organized.`[2]

2.2 Synset Gloss

Synset gloss is a natural language text that defines the corresponding lexical concept of the synset. WordNet sometimes enriches the glosses with example usage (example sentences) to show that the synset synonyms are exchangeable in some context. For example, the following gloss definition is enriched with two example sentences to show the synonymy between the terms `love` and `passion`.

#2 `love, passion: any object of warm affection or devotion; "the theater was her first love"; "he has a passion for cock fighting".`

A gloss contain two parts:

- *Genus*: Corresponds to the classifying property of the concept. For example, the genus of the gloss in the previous example is `object`.
- *Differentia*: Corresponds to the distinguishing characteristics of the concept. For example, the differentia of the gloss in the previous example is `warm affection or devotion`.

[1]http://en.wikipedia.org/wiki/Brocket_deer
[2]http://en.wikipedia.org/wiki/Folder

Notice that glosses are informal descriptions of concepts. This leads to an ambiguity of glosses due to the ambiguity of natural language. Another point is that there is no explicit distinction between genus and differentia. In any case, the construction of glosses in WordNet is adhoc and there is no systematic procedure or construction rules to define glosses so that WordNet glosses need disambiguation [37] [38] [39].

2.3 Synset Relations

WordNet uses lexical relations to organize the relations between words and semantic relations to organize the relations between synsets. Some relations are both lexical and semantic relations. WordNet 2.1 uses 26 relations, where 4 relations are only lexical, 15 relations are only semantic and 7 relations are both lexical and semantic.

A relation in WordNet can be represented as triple ⟨`source_category, relation, target_category`⟩, where `source_category` and `target_category` are grammatical categories. For example, the `hypernym` relation holds between nouns ⟨`noun, hypernym ,noun`⟩ and verbs ⟨`verb,hypernym, verb`⟩, but not between adjectives or adverbs. The source_category and target_category can be different categories. For example the relation `derivationally related form` can be between nouns ⟨`noun, derivationally related form, noun`⟩ or between nouns and verbs ⟨`noun, derivationally related form, adjective`⟩.

In the following, we list the main semantic relations for nouns in WordNet:

- *Hypernym*: `big cat` is a hypernym of `jaguar`.

- *Hyponym*: `jaguar` is a hyponym of `big cat`.

- *Member holonym*:`orthography` is a Member holonym of `punctuation`.

- *Member meronym*: `eta` is a Member meronym of `Greek_alphabet`.

- *Part holonym*: `mane` is a Part holonym of `lion`.

- *Part meronym*: `wishbone` is a Part meronym of `bird`.

- *Substance holonym*: `blood` is a Substance holonym of `blood_plasma`.

- *Substance meronym*: `oxygen` is a Substance meronym of `ozone`.

Although WordNet relations are useful to organize the relations between the synsets, crucial relationships between the synsets remain implicit or sometimes missing in the synset glosses. For example, the relation between `correctness` and `conformity` is implicit and the relation between `fact or truth` and `social expectations` in the following two meanings of the term `correctness` is missing.

#1 `correctness, rightness: conformity to fact or truth.`

#2 `correctness: the quality of conformity to social expectations.`

A human being may understand that `correctness` is a hyponym of `conformity` and `fact or truth` is a hyponym of `social expectations`, but this is extremely difficult or impossible for a machine because `conformity` is neither the hypernym of #1 nor #2. The relation between `fact or truth` and `social expectations` is missing because `social expectations` is simply not defined in WordNet.

2.4 Preferred Term and Preferred Sense

As explained previously, synonymy means that a synset may be denoted by more than one term. On the other hand, polysemy means that a term may denote more than one synsets of the same grammatical category. For example, the polysemous term `collaboration` denotes the following two synsets.

#1 `collaboration, coaction: act of working jointly; "they worked either in collaboration or independently".`

#2 `collaboration, collaborationism, quislingism: act of cooperating traitorously with an enemy that is occupying your country.`

Due to synonymy and polysemy, the relation between terms and synsets is many to many relationship. Two important questions here are:

- In case of synonymy: which is the best term to denote a synset?
- In case of polysemy: which is the best synset is denoted by the polysemous term?

To answer the first question, synset lemmas in WordNet are associated with term rank. This rank reflects which is the best term to denote a synset. The best term is called *the preferred term*. For example, `universe` is the preferred term of the following synset.

#1 `universe, existence, creation, world, cosmos, macrocosm: everything that exists anywhere.`

To answer the second question, wordNet orders the synsets of polysemous terms. This ordering reflects which is the best synset that is denoted by the polysemous term. The synset with the highest rank is called *the preferred sense*. In the previous example, the preferred sense of `collaboration` is #1.

Chapter 3

Polysemy in WordNet

WordNet 2.1. contains 147,257 words, 117,597 synsets and 207,019 word-sense pairs. these words there are 27,006 polysemous words, where 15776 of them are nouns. In this thesis, we are dealing with polysemous nouns at

# of senses	# of nouns	in percentage
1	89760	86.1%
2	9328	8.95%
3	2762	2.65%
4	1083	1.05%
5	555	0.54%
6	277	0.25%
7	194	0.18%
8	90	0.07%
9	88	0.07%
10	54	0.05%
>10	94	0.09%
Total	104285	100%

Table 3.1: Polysemous nouns in WordNet

the concept level only. We do not consider polysemy at instance level. After removing the polysemous nouns that refer to proper names, the remaining polysemous nouns are 14530 nouns. The number of senses a polysemous noun may have, ranges from 2 senses to 33 senses. Table 3.1 shows the distribution of the polysemous nouns at the concept level according to the number of senses they have. WordNet defines polysemy as follows:

#1 `polysemy, lexical ambiguity: the ambiguity of an individual word or phrase that can be used (in different contexts) to express two or more different meanings.`

We briefly describe the various polysemy classes in WordNet.

3.1 Compound Noun Polysemy

A term in wordNet can be a single word such as `center` or a collocation such `nerve center`. In the case of nouns, collocations correspond to compound nouns. A compound noun contains two parts.

1. *noun adjunct/modifier*: a noun that modifies another noun in a compound noun.

2. *noun head/modified noun*: the modified noun in a compound noun.

For example, the noun `head` is the noun adjunct and `word` is the modified noun in the compound noun `head word`. Compound noun polysemy [8] corresponds to the polysemy cases, in which the modified noun or the noun adjunct is synonymous to its corresponding noun compound and belongs to more than one synset. For example, the term `center` is synonymous to the compound noun in the following synsets.

#2 `center field, center: the piece of ground in the outfield directly ahead of the catcher.`

#6 `center, center of attention: the object upon which interest and attention focuses.`

#7 `center, nerve center: a cluster of nerve cells governing a specific bodily process.`

#15 `mall, center, shopping mall, shopping center: mercantile establishment consisting of a carefully landscaped complex of shops ... .`

WordNet contains a substantial amount of compound noun polysemy. However, it is not clear, which rule wordNet is following by adding the noun head or the noun modifier terms as a synonym to their corresponding compound nouns. In this example, it is not clear, why wordNet considers the term `center` to be a synonym of the compound noun in the previous cases and it does not consider it a synonym of the terms `city center`, `medical center`, or `research center`.

#1 `city center, city centre, central city: the central part of a city.`

#1 `medical center: the part of a city where medical facilities are centered.`

#1 `research center, research facility: a center where research is done.`

3.2 Specialization Polysemy

Specialization polysemy is a type of related polysemy which denotes a hierarchical relation between the meanings of a polysemous term [14]. In case of abstract meanings, we say that a meaning A is a more general meaning of a meaning B. We say also that the meaning B is a more specific meaning of the meaning A. In the cases, where the meanings denote physical entities, we may also use the taxonomic notations type and subtype instead of more general meaning and more specific meaning respectively. For example, we say that the first meaning of `turtledove` is a subtype of the second meaning.

#1 `australian turtledove, turtledove: small Australian dove.`

#2 `turtledove: any of several Old World wild doves.`

The first meaning of `correctness` in the following example is more specific than the second meaning.

#1 `correctness, rightness: the quality of conformity to fact or truth.`

#2 `correctness: the conformity to social expectations.`

The relation between the meanings of specialization polysemy cases is hierarchical. This implies that these meanings should belong to the same type (taxonomic category) which corresponds to the common root of both meanings. The common root may be a direct parent of the meanings as in the `turtledove` example. It is also possible that the meanings are connected indirectly to common root, i.e. a least common subsumer that can be considered as a more general meaning of these meanings. For example, the common root of both meanings of `correctness` is `attribute`.

3.3 Metonymy

Metonymy polysemy happens when we substitute the name of an attribute or a feature for the name of the thing itself [40]. For example, the term in the second meaning refers to part of fox.

#1 `fox: alert carnivorous mammal with pointed muzzle and ears and a bushy tail.`

#2 `fox: the grey or reddish-brown fur of a fox.`

In metonymy, there is always a base meaning of the term and other derived meanings that express different aspects of the base meaning [41]. Meaning #1 of the term `fox` in the previous example is the base meaning and meaning #2 is a derived meaning of the term. Metonymy is different from specialization polysemy in the following way: The meanings of metonymy terms belong to different types/ concept classes. Thus the relation more general meaning/ more specific meaning is not applicable for metonymy.

For example, the base meaning of the term `fox` belongs to `animal` while the derived meaning belongs to `artifact`. This means, the relation between the derived meanings and the base meaning of a metonymy term cannot be hierarchical as it is the case in specialization polysemy.

3.4 Metaphoric Polysemy

Metaphoric polysemy cases are the cases in which a term has literal and figurative meanings [42]. In the following example, the first meaning of the term `honey` is the literal meaning and the second meaning is the figurative.

#1 `honey: a sweet yellow liquid produced by bees.`

#2 `beloved, dear, dearest, loved one, honey, love: a beloved person.`

The metaphoric relation between the literal meaning and the figurative meaning may disappear or it may become difficult to understand the metaphoric link between the figurative and literal meaning. We call such cases dead metaphors. For example, the meanings of the term `animator` indicate a dead metaphor.

#1 `energizer, animator: someone who imparts energy and vitality to others.`

#2 `animator: the technician who produces animated cartoons.`

From hierarchical point of view, metaphors differ from metonymy and specialization polysemy. The meanings of a metonymy case belong to different categories and the meanings a specialization polysemy case should belong to the same category. In the case of metaphors, we may find metaphoric cases whose meanings belong to different categories and we may find cases whose meanings belong to the same category (local metaphors [43]). For example the literal meaning of `honey` belongs to `food`, while the metaphoric meaning belongs to `person`. On the other hand, both the literal and the figurative meaning of the term `role player` belong to `person`.

#1 `pretender, role player: a person who makes deceitful pretenses.`

#2 `actor, role player: a theatrical performer.`

Although, it is possible to find metaphoric cases in which the literal and figurative meaning belong both to the same category, the metaphoric relation is not hierarchical. The metaphoric link between the meanings is raised usually through inconsistency between the literal and the metaphoric meaning. For example, the meaning #1 of the term `role player` belongs to the concept `person`, while #2 is a role and thus these meanings are inconsistent and cannot be generalized to a common type.

3.5 Homonymy

From linguistic point of view [44], the meanings in a homonymy case have different etymological origins and they are not related. For example, the origin of meaning #1 of the term `bank` is Italian, while the second meaning is Norwegian.

#1 `depository financial institution, bank: a financial institution.`

#2 `bank: sloping land (especially the slope beside a body of water).`

From knowledge representation point of view, the etymology is not sufficient in all cases to capture homonymy [44]. For example: the following two meanings share the same term that refers to the famous French mathematician `Pascal`. Linguistically, both meanings are related since both of them are named after `Pascal`. Nonetheless, these meanings are in fact homonyms since they belong to two totally different categories: `unit of measurement` and `programming language`, respectively.

#1 `Pascal, Pa: a unit of pressure equal to one newton per square meter.`

#2 `Pascal: a programing language designed to teach programming.`

Some current researches suggested the perceived relatedness [44] as a criterion to identify homonymy cases such as the case of `animator`, or `pascal` in the previous examples.

Chapter 4

The Problem

The polysemy problem in WordNet has been addressed in many research papers and PhD dissertations. The state of the art approaches describe the problem in many ways such as the problem of the highpolysemous nature of wordNet, the problem of sense enumeration, the problem of redundancy, the problem of too-fine grained senses in WordNet, the problem of implicit relatedness, or the problem that WordNet does not differentiate between the different polysemy classes. All these descriptions are true but non of them is sufficient to describe the polysemy problem in WordNet completely. In fact they describe partial aspects of the problem not the problem itself.

In this approach, we classify the polysemy problem into two main problems:

1. **The problem of the highpolysemous nature of WordNet**: The highpolysemous nature of wordNet makes it very difficult to be used by NLP applications.

2. **The problem of unspecified information**: WordNet does not differentiate between the polysemy classes. Recognizing the polysemy class of a given polysemous term is essential for NLP.

The second problem is related to all polysemy classes in WordNet. The first problem on the other hand is not related to all polysemy classes in

WordNet. In particular, it is related to metonymy, specialization polysemy and compound noun polysemy.
Metonymy may be one of the main sources of the highpolysemous nature of WordNet. For example, WordNet contains 9 meanings for the term `book`, the first 7 meanings of them belong to the metonymy polysemy class.

#1 `book:a written work or composition that has been published (printed on pages bound together); "I am reading a good book on economics".`

#2 `book, volume: physical objects consisting of a number of pages bound together; "he used a large book as a doorstop".`

#3 `ledger, leger, account book, book of account, book: a record in which commercial accounts are recorded; "they got a subpoena to examine our books".`

#4 `book: a number of sheets (ticket or stamps etc.) bound together on one edge; "he bought a book of stamps".`

#5 `record, record book, book: a compilation of the known facts regarding something or someone; "Al Smith used to say, Let's look at the record"; "his name is in all the record books".`

#6 `book: a major division of a long written composition; "the book of Isaiah".`

#7 `script, book, playscript: a written version of a play or other dramatic composition; used in preparing for a performance.`

#8 `book: a collection of playing cards satisfying the rules of a card game.`

#9 `book, rule book: a collection of rules or prescribed standards on the basis of which decisions are made; "they run things by the book around here".`

In this approach, we do not consider the problem of metonymy. The state of the art approaches such as CORELEX [3] that we are going to describe in chapter 5 offered a good solution to the problem of metonymy. However, solving the polysemy problem in the case of metonymy reduces the the highpolysemous nature of WordNet partially. In fact, the problem remains unsolved for specialization polysemy and compound noun polysemy.

In the following, we describe the problem of the highpolysemous nature in specialization polysemy and compound noun polysemy.

4.1 The Problem of the highpolysemous Nature of WordNet

In the following we give an overview about the highpolysemous nature of WordNet in numbers. In Table 4.1, we give an overview about the nouns in WordNet. The table shows that WordNet contains 104290 nouns,

#Nouns	104290
#Synsets	74314
#Senses	130207

Table 4.1: Number of nouns, noun senses and noun synsets in WordNet 2.1

and 74324 synsets. Some nouns appear in several synsets creating 130207 senses. In Table 4.2, we compute the following averages. The average

#Noun per synset	1.4
#Noun per sense	0.8
#Synset per noun	0.71
#Sense per noun	$\approx$ 1.25

Table 4.2: Polysemy average in WordNet 2.1

noun number per synset is 1.4 and the average sense number per noun is about 1.25. These averages make the impression that WordNet is not highpolysemous. This is not true, WordNet is in fact highpolysemous as follows. In Table 4.3 and 4.4, we consider the polysemous nouns only.

#Polysemous Nouns	14530
#Polysemous synsets	29723
#Polysemous senses	59077

Table 4.3: Number of polysemous nouns, polysemous noun senses and polysemous noun synsets in WordNet 2.1

#Polysemous noun per polysemous synset	0.48
#Polysemous noun per polysemous sense	$\approx$ 0.25
#Polysemous synset per polysemous noun	2.0
#Polysemous sense per polysemous noun	$\approx$ 4.0

Table 4.4: Polysemy average in polysemous nouns in WordNet 2.1

According to Tables 4.3 and 4.4, a polysemous noun belongs in average to two synsets. The average of polysemous synsets per noun is 4. To make the highpolysemous nature problem clearer, we calculate in Table 4.5 the following percentages. That means, less than 14% of the nouns in wordNet

% of polysemous Nouns	13.93%
% of polysemous senses	45.37%
% of polysemous synsets	40%

Table 4.5: Polysemy average in polysemous nouns in WordNet 2.1

own more than 45% of the senses, and about 40% of the synsets.
In the following, we give an overview about the problem of the highpolysemous nature of wordNet in specialization polysemy and compound noun polysemy. We consider the highpolysemous nature in these polysemy classes as a result of the following problems.

a) *The problem of implicit relatedness*

b) *The problem of too fine-grained senses*

c) *The problem of redundancy*

d) *The problem of sense enumeration*

In the following, we discuss these problems.

4.1.1 The Problem of Compound Noun Polysemy

Compound noun polysemy may be the main resource of sense enumeration in WordNet. Sense enumeration means a misconstruction that results in wrong assigning of a synset to a term. Consider for example, the following synsets where `head` is synonymous to a compound noun.

#8 `fountainhead, headspring, head: the source of water from which a stream arise.`

#9 `head, head word:` *grammar* `the word in a grammatical constituent that plays the same grammatical role as the whole constituent.`

#13 `principal, school principal, head teacher, head: the educator who has executive authority for a school.`

#16 `promontory, headland, head, foreland: a natural elevation (especially a rocky one that juts out into the sea).`

#21 `headway, head: forward movement.`

#27 `read/write head, head: (computer science) a tiny electromagnetic coil and metal pole used to write and read magnetic patterns on a disk.`

#32 `drumhead, head: a membrane that is stretched taut over a drum.`

Using the term `head` to refer to any of the previous synsets is discourse dependent and can be understood only in a proper surrounding context. Notice that `head` is the preferred term in #9 only. The preferred terms in the other synsets are the compound nouns that correspond to more specific terms that denote the synsets precisely. For example, the preferred term in #27 is `read/write head`.

The term `head` is the most polysemous noun in WordNet. It has 33 senses. Notice that this type of sense enumeration in WordNet is not systematic. For example, in analogy to synset #13, the term `head` could be also synonymous to the terms in the following synsets:

#1 `department head: the head of a department`

#1 `head of household:the head of a household or family or tribe`

...

4.1.2 The Problem of Specialization Polysemy

Specialization polysemy in WordNet contributes to the highpolysemous nature of wordNet as follows.

The Problem of implicit Relatedness in Specialization Polysemy

The implicit relatedness in specialization polysemy is a hierarchical relation. Representing the hierarchical relation in specialization polysemy cases at lexical level rather than the semantic level is a kind of sense enumeration that leads to high polysemy and information lost. Which is the the more general meaning and which is the more specific meaning is encoded implicitly in the glosses. For example, what is the relation between #1 and #2 in the following? Notice that both meanings share the same common parent `body part`.

```
[#1] dorsum -- (the back of the body of a vertebrate or any analogous surface
   (as the upper or outer surface of an organ or appendage or part); "the
   dorsum of the foot")
```

```
[#2] back, dorsum -- (the posterior part of a human (or animal) body from the
   neck to the end of the spine; "his back was nicely tanned")
      => body part -- (any part of an organism such as an organ or extremity)
```

The Problem of too fine grained senses, Redundancy and Sense Enumeration in Specialization Polysemy

In the following, we briefly discuss the problems too fine grained senses, redundancy and sense enumeration of in specialization polysemy.

The problem of too fine grained senses
Many specialization polysemy cases in WordNet are too fine grained. For example, capturing the difference between the following meanings of the term `optimism` is very difficult.

#1 `optimism: the optimistic feeling that all is going to turn out well.`

#2 `optimism: a general disposition to expect the best in all things.`

The problem of redundancy
Many specialization polysemy cases in WordNet are redundant as in the following example.

#1 `calisthenics, callisthenics: the practice of calisthenic exercises; "calisthenics is recommended for general good health".`

#2 `calisthenics, callisthenics: light exercises designed to promote general fitness; "several different calisthenics were illustrated in the video".`

The problem of sense enumeration
Many specialization polysemy cases in WordNet are sense enumerations as in the following examples.

#10 `key:a list of answers to a test.`

#11 `key: a list of words or phrases that explain symbols or abbreviations.`

The illustrated problems in specialization polysemy contribute to useless increase of polysemy in WordNet such that the polysemy in WordNet becomes a challenging problem for NLP applications [45].

4.2 The Problem of Unspecified Information

The highpolysemous nature of WordNet is a part of the problem. The second part is that WordNet does not differentiate between the polysemy classes. For example differentiating between metaphoric polysemy and homonymy is not provided in WordNet [3].

Homonymy, metaphoric, and metonymy polysemy are essential in WordNet. Even after solving the polysemous high nature of wordNet in specialization polysemy, compound noun polysemy and metonymy, the problem of differentiating between the residual polysemy classes remains unsolved. Representing the polysemy at lexical level only without differentiating between them makes WordNet confusing for NLP. Consider for example the following thee meanings of `food`.

#1 `food, nutrient: any substance that can be metabolized by an organism to give energy and build tissue.`

#2 `food, solid food: any solid substance (as opposed to liquid) that is used as a source of nourishment; "food and drink".`

#3 `food, food for thought, intellectual nourishment: anything that provides mental stimulus for thinking.`

In this example, #1 and #2 belong to specialization polysemy. On the other hand #3 is metaphoric meaning of #1 and #2. After solving the problem of #1 and #2, the problem of determining the polysemy class of the resulting synset and #3 remains unsolved.

Of course Word sense disambiguation (WSD) [23] tools can be used to solve this problem. The accuracy of these tools is less than 80% in best cases [46]. The other problem is that deploying such tools in an NLP application is time consuming and affects the the usability of such tools as online applications.

Part II

State of the Art

Chapter 5

State of the Art

The approaches of polysemy can be classified in two main approaches. The first is polysemy reduction, where the focus is on complementary polysemy to produce more coarse-grained lexical resources of existing fine-grained ones such as WordNet [13]. The second type of polysemy approaches focuses on classifying polysemy into systematic or regular polysemy and homographs. These regular polysemy approaches including the approach presented in this thesis rely on Apresjan's definition of regular polysemy: "*A polysemous Term T is considered to be regular if there exists at least another polysemous T' that is semantically distinguished in the same way as T*" [47]. Based on this definition, CORELEX was introduced as ontology of systematic polysemous nouns extracted from WordNet. Other approaches, such as [27], were introduced to extract semantic relations between regular polysemous terms in WordNet. These approaches propose to enrich wordNet with semantic relations that correspond to the implicit relations between the complementary polysemous terms in WordNet [27] [28]. In the following, we summarize polysemy reduction approaches, CORELEX, and the most prominent semantic relations extraction approaches.

5.1 Polysemy Reduction Approaches

In polysemy reduction, the senses are clustered or merged such that each group contains related polysemous words. These groups are called homograph clusters [25]. Once the clusters have been identified, the senses in each cluster are merged. To achieve this task, several strategies have been introduced [13]. These strategies can be mainly categorized in semantic-based and probability-based strategies . Some approaches combine both strategies [48]. Although results of applications of these approaches are reported, these results are taken usually from applying them on sample data sets and there is no way to verify these results independently. Polysemy reduction approaches typically rely on the application of some detection rules such as: *If s_1 and s_2 are two synsets containing at least two words, and if s_1 and s_2 contain the same words, then s_1 and s_2 can be collapsed together into one single synset* [13]. However, there is no linguistic motivation behind this rule. Applying this rule may wrongly result in merging two different senses as in the following example.

#1 `smoke, smoking: a hot vapor containing fine particles of carbon`

#2 `smoke, smoking: the act of smoking tobacco or other substances.`

In general, polysemy reduction can neither predict the polysemy type occurring between the senses of polysemous words nor can deal with metonymy or metaphors. Polysemy reduction does not solve the polysemy problem in linguistic resource. Nevertheless, some rules such as the `common parent rule` [13] are linguistically motivated and can be adopted in solving part of the polysemy problem, namely the identification and merging of genuine redundant synsets.

Common Parent Rule in Polysemy Reduction Approaches

If s_1 and s_2 are two synsets with the same hypernym, and if s_1 and s_2

contain the same words then s_1 and s_2 can be collapsed together into single synset s_{12}.

5.2 CORELEX

CORELEX, the first systematic polysemy lexical database, follows the generative lexicon theory [49] that distinguishes between systematic (also known as regular or logic) polysemy and homographs. Systematic polysemous meanings are systematic and predictable while homonyms are not regular and not predictable. The polysemy type of the term `fish` in the following example is systematic since the meaning `food` can be predicted from the `animal` meaning and so these two meanings of `fish` belong to the systematic class `animal#food`.

#1 `fish: any of various mostly cold-blooded aquatic vertebrates usually having scales and breathing through gills; "the shark is a large fish"; "in the living room there was a tank of colorful fish".`

#2 `fish: the flesh of fish used as food; "in Japan most fish is eaten raw"; "they have a chef who specializes in fish".`

The two meanings of fish describe two related aspects of `fish`: `fish` as `animal` and `fish` as `food`. Two meanings of a polysemous word are systematic polysemous means that the meanings of this word are not homonyms and they describe different aspects of the same term. Following this distinction, CORELEX organizes the polysemous nouns of WordNet 1.5 into 126 systematic polysemy classes. These classes are combinations of 39 `basic types` that reside at the top level of WordNet hierarchy such as {`animal, food, attribute, state, artifact, ...`}. The idea is that metonymy cases can be underspecified to one of these classes. For example, the 7 senses of `book` that we have seen in chapter 4 can be underspecified to two senses `artifact` and `communication` [4].

Despite the effectiveness of the underspecification in CORLEX in metonymy, it is not suitable to solve the polysemy problem in other polysemy classes. The systematic polysemy classes in CORELEX have been determined in a top down fashion considering the patterns in the upper level of WordNet hierarchy only. The high level basic types in CORELEX patterns make them too coarse grained to extract useful semantic relations [27] [28] [4]. At the same time, there are hundreds of regular structural patterns that reside in the middle level and lower level of word-Net hierarchy that are not covered by the high level basic types. These patterns correspond to metaphoric [27] and specialization polysemy [4]. The underspecification method is not appropriate to CORELEX patterns that correspond to metaphoric polysemy. CORELEX patterns contain too many false positives [27] such as the following two meanings of the term `colt` that belong to the pattern `animal#artifact`

#1 `colt: a young male horse under the age of four.`

`colt: a kind of revolver.`

Some patterns correspond to homonymy. For example, according to our analysis, the pattern `animal#psychological feature` contains 105 homonymy cases such as the following meanings of the term `slider`.

`pseudemys scripta, slider, yellow-bellied terrapin: freshwater turtle of United States and South America.`

`slider: a fastball that curves slightly away from the side from which it was thrown.`

Another important point is related to the fine grained nature of WordNet, where the meanings of some CORELEX classes are very difficult to disambiguate, and indistinguishable even for humans [50] such as the pattern `attribute#state`. Consider the following two meanings of `pressure`.

#2 `pressure: a force that compels; "the public brought pressure to bear on the government".`

#4 `imperativeness, insistence, insistency, press, pressure: the state of demanding notice or attention; "the insistence of their hunger"; "the press of business matters".`

However, the construction of CORELEX was based on WordNet 1.5. In subsequent versions of WordNet, massive changes in the hierarchical structure of wordNet have been made [51]. These changes affect CORELEX classes such that there is a need to rebuild them. For example, `absorbency` that belongs to CORELEX pattern `attribute#state` has one meaning only in WordNet 2.1. Other words such as `abstemiousness` do not belong to this pattern anymore.

5.3 Semantic Relations Extraction Approaches

The semantic relations extraction approaches are regular polysemy approaches that attempt to extract implicit semantic relations between the polysemous senses via regular structural patterns. The basic idea in these approaches is that the implicit relatedness between the polysemous terms corresponds to variety of semantic relations. Extracting these relations and making them explicitly should improve wordNet [27]. These approaches refine and extend CORELEX patterns to extract the semantic relations. Beside the structural regularity, these approaches exploit also the synset gloss [4] and the cousin relationship [28] [27] in WordNet. For example, the approach described in [4] exploits synset glosses to extract `auto-referent` candidates. The approach described in [28] uses several rules, such as `ontological bridging` [28] to detect relations between the sense pairs.

Ontological Bridging rule

a sense pair $\langle s_1, s_2 \rangle$ for a word w can be bridged if s_1 has a hypernym that can be lexicalized as $M - H$ and s_2 has a hypernym that can be lexicalized as M.

An example for applying this rule is the following two meanings of the word `basketball`, where #1 is a transitive hyponym of `game`, and #2 is a hyponym of `game equipment`. In this case then, $M =$ `game` and $H =$ `game equipment`. Thus #2 denotes `the equipment used in the activity of` #1.

#1 `basketball, basketball game, hoops: a game played on a court by two opposing teams of 5 players ....`

#2 `basketball: an inflated ball used in playing basketball.`

In general, the extracted relations in these semantic relations extraction approaches are similar. For example, we find the relations `similar to` or `color of` in the results of the approach in [4]. The result in [28] contains relations such as `contained in`, `obtain from`. Similarly, the result in [27] contains relations such as `fruit of`, `tree of`.

The semantic relations extraction approaches are in general better than CORELEX in the following aspects. First of all, the discovered patterns in these approaches are more fine grained and enable to capture meaningful relations. These approaches classified the complementary polysemy into three sub classes: metonymy, metaphoric, and specialization polysemy, while CORELEX did not classify complementary polysemy. Another important point in these approaches is that these approaches considered the problem of false positives. However, these approaches did offer a solution to the highpolysemous nature of metonymy. They cover only few patterns of the specialization polysemy and metaphoric cases. They did not address the problem of too fine grained senses or compound noun polysemy.

Part III

Solution

Chapter 6

Proposed Solution

In the following, we present our proposed solution for the two problems described in chapter 4.

6.1 Solving the Problem of the highpolysemous Nature of WordNet

For solving the polysemy problem in metonymy, CORELEX and the semantic relations extraction approaches are possible solutions. The underspecification method in the first approach reduces the highpolysemous nature in Metonymy cases on the one hand , and enriching wordNet with semantic relations solves the unspecified information problem on the other hand. A hybrid solution that combines the advantages of both approaches may be an optimal solution. In the following, we present our solution to reduce the highpolysemous nature in compound noun polysemy and specialization polysemy.

6.1.1 Solving the Problem in Compound Noun Polysemy

We solve the problem of sense enumeration in compound noun polysemy by disambiguating the synsets that belong to this polysemy class. Disambiguating means that we remove the polysemous term that corresponds to the modified noun or noun modifier and keep the compound noun that defines the synset precisely. For example, #b refers to the synset after applying the disambiguation operation on #a as follows.

#a `fountainhead, headspring, head: the source of water from which a stream arises; "they tracked him back toward the head of the stream".`

#b `fountainhead, headspring: the source of water from which a stream arises; "they tracked him back toward the head of the stream".`

6.1.2 Solving the Problem in Specialization Polysemy

Solving Implicit Relatedness

The implicit relatedness in specialization polysemy is a hierarchical relation. For two synsets s_1, s_2 in a specialization polysemy case, the hierarchical relation can be one of the following:

- *Missing relation*: Corresponds to the cases where s_1 is a more general meaning of s_2 or vice versa.

- *Missing parent*: Corresponds to the cases where s_1 and s_2 are more specific meanings of a (missing) more general meaning synset.

We solve the implicit relatedness in both cases by transforming the implicit relation into explicit semantic relation as follows:

- *Solution to missing relation*: We add a new hierarchical relation that links the more specific synset to the more general synset as schematized in Figure 6.1.

- *Solution to missing parent*: we create a new parent and link both synsets to the missing parent as schematized in Figure 6.3.

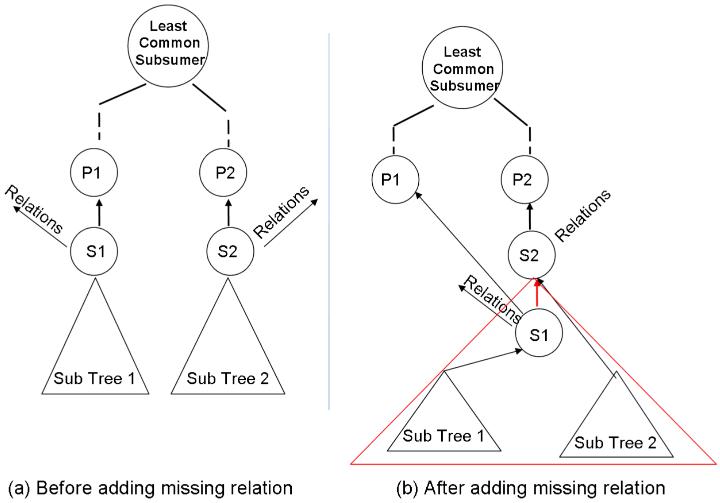

Figure 6.1: Adding a missing relation

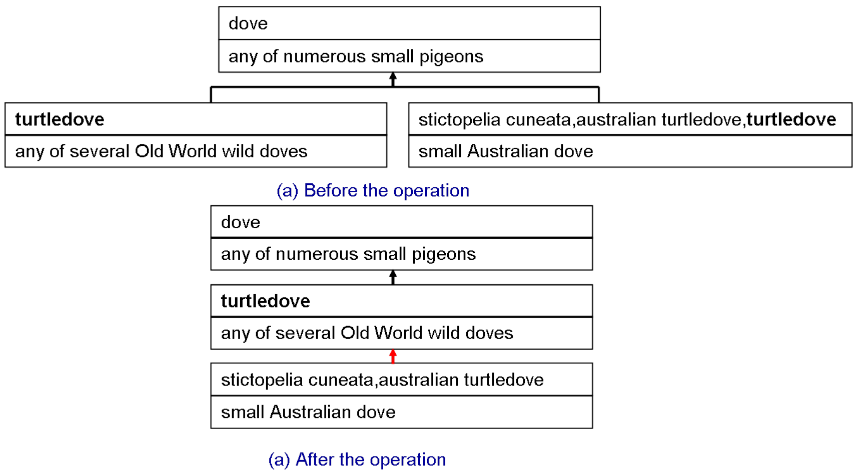

Figure 6.2: Example of adding a missing relation

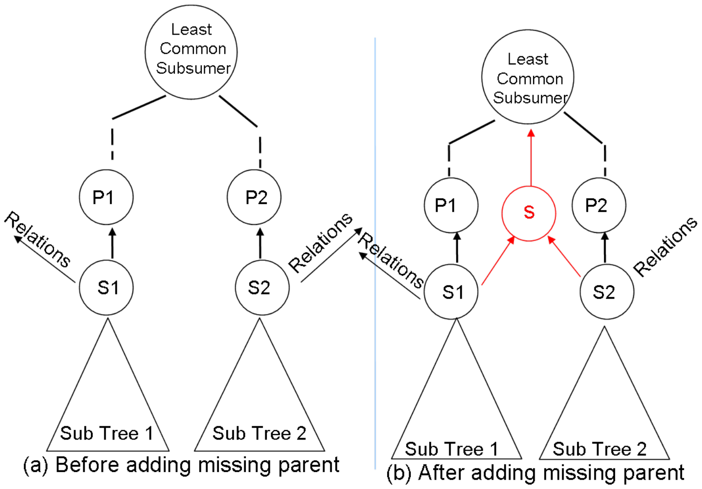

Figure 6.3: Adding a missing parent

An example of adding a missing relation is shown Figure 6.2. An example of adding a missing parent is shown Figure 6.4.

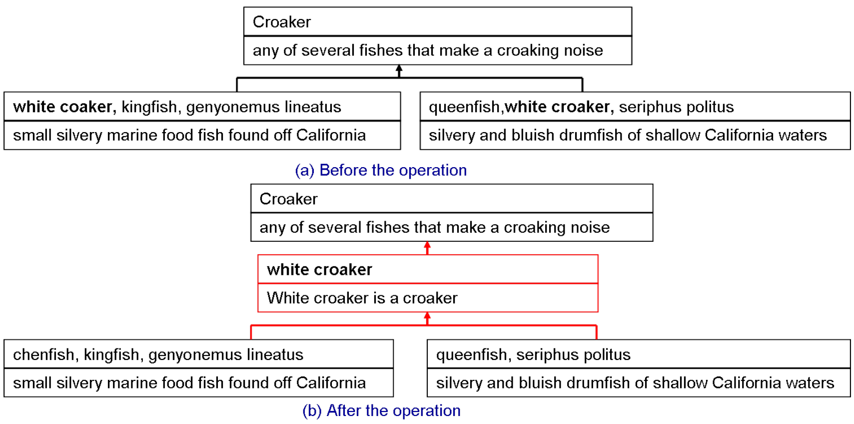

Figure 6.4: Example of adding a missing parent

Solving Redundancy, too fine grained Senses and Sense Enumeration

For solving redundancy, too fine grained senses and sense enumerations in specialization polysemy cases, we propose the merge operation as schematized in Figure 6.5. An example of a merge operation is shown Figure 6.6.

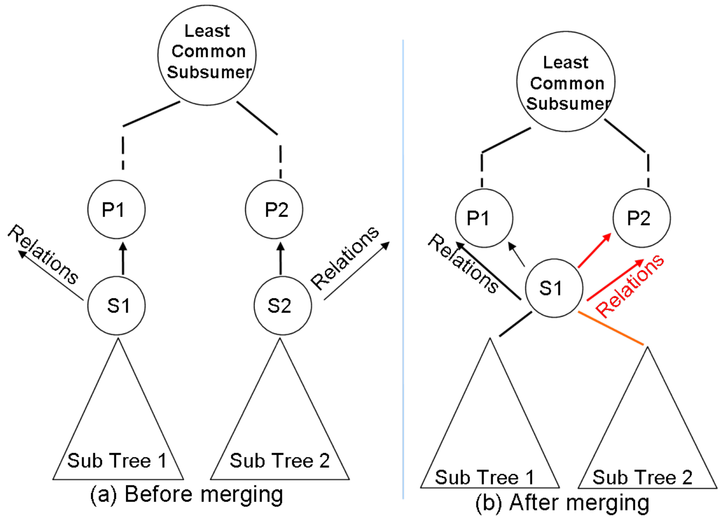

Figure 6.5: Merge operation

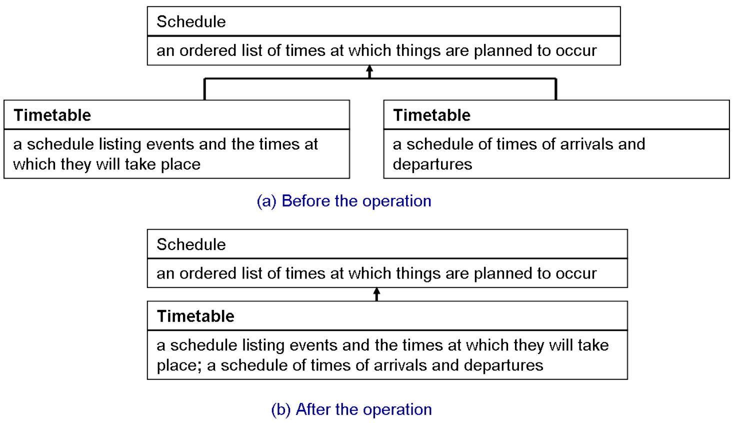

Figure 6.6: An example of merge operation

6.2 Solving the Problem of Unspecified Information in WordNet

We propose enriching WordNet with the following relations to denote the polysemy class in the case of homonymy and metaphoric polysemy.

- `is_homograph` to denote that two terms are homographs.
- `is_metaphor` to denote the metaphoric relation between the metaphoric meaning and literal meaning in a metaphoric polysemy case.

Chapter 7

Algorithm Overview

We divide the solution in two stages. In the first stage, we solve the problem of the highpolysemous nature in WordNet. In the second stage, we solve the problem of unspecified information to a subset of homonymy and metaphoric cases in WordNet. The input of our approach is the current structure of the noun synsets in WordNet. The output is the structure reorganized, where the reorganization is the result of (i) disambiguating compound noun polysemy cases, (ii) transforming the hierarchical relation from the lexical level into the semantic level, (iii) removing redundancy, too fine grained senses and sense enumerations in specialization polysemy cases, and (iv) explicitly denoting homonymy and metaphoric polysemy cases.
In the following, we briefly describe the two stages of our approach.

7.1 S1: Reducing the highpolysemous Nature of WordNet Algorithm

Reducing the highpolysemous nature of WordNet is performed in two phases.

7.1.1 S1.P1: Solving the Compound Noun Polysemy Problem Algorithm

In this phase, we solve the sense enumeration problem caused by compound noun polysemy cases. This is performed by a semi-automatic process that includes the following steps.

S1.P1.1 **Compound noun polysemy discovery**: Compound noun polysemy discovery is performed semi-automatically as follows.

1. **Compound noun candidates discovery**: This step is automatic and performed by deploying an algorithm that returns compound noun polysemy candidates.
2. **Manual validation**: This step is manual, where we exclude the false positives from the output of the algorithm in the previous step. For example, we exclude term abbreviations and specialization polysemy cases.

S1.P1.2 **Compound noun polysemy disambiguation**: In this step, we disambiguate the polysemous terms of the identified cases by removing the polysemous noun modifier and keeping the compound noun.

7.1.2 S1.P2: Solving the Specialization Polysemy Problem Algorithm

The algorithm for solving the specialization polysemy works in two steps.

S1.P2.1 Specialization polysemy discovery.

S1.P2.2 Specialization polysemy organization.

The input of the algorithm is the resulting WordNet after applying the operations in S1.P1. The output is the result removing redundancy, too fine grained senses and sense enumerations and transforming the implicit hierarchical relation between specialization polysemy synsets to explicit semantic relations.

In the following, we discuss these steps.

S1.P2.1 **Specialization polysemy discovery**

Specialization polysemy discovery works in the following three steps.

S1.P2.1.1 **Structural pattern discovery**: In this step, we deploy an algorithm for extracting the structural patterns. The input of the algorithm is the current structure of WordNet. The algorithm returns an associative array of structural patterns associated with their corresponding polysemy cases.

S1.P2.1.2 **Structural pattern classification**: In this step, we manually classify the structural patterns returned in the previous step. The output is four associative arrays of patterns associated with list of nouns. These four lists are:

1. *Specialization polysemy patterns*: This list contains the patterns whose corresponding cases are specialization polysemy candidates.
2. *Metaphoric patterns*: This list contains the patterns whose corresponding cases are metaphoric candidates.
3. *Homographs patterns*: This list contains homonymy patterns.
4. Singleton patterns: The patterns in this group are those patterns that have one polysemy case only and thus cannot be considered to be regular.

S1.P2.1.3 **Identifying false positives**: In this step, we manually process the polysemy cases in the four lists from the previous step. Our task is to decide the polysemy classes for the cases in the singleton patterns list and remove false positives form the other three lists. The outputs of this phase are three lists:

1. *Specialization polysemy instances*
2. *Metaphoric polysemy instances*
3. *Homonymy instances*

S1.P2.2 **Specialization polysemy organization**

Specialization polysemy organization is automatic and performed in two steps.

1. **Specialization polysemy sub classes discovery**: The input of this step is the list of specialization polysemy instances, the output of S1.P2.1.3 (item 1). Based on the synset patterns, these instances are divided automatically into the following three sub classes.

   (a) *Missing relation instances*: The synsets in the instances of this sub class indicate a missing hierarchical relation.

   (b) *Missing parent instances*: The synsets in the instances of this sub class are more specific meanings of a missing more general synset.

   (c) *Too fine grained, redundant, and sense enumeration instances*: The instances in this group are redundant or too fine grained senses or sense enumeration instances.

2. **Applying specialization polysemy operations**: The input of this step is the three lists of specialization polysemy that correspond to the sub classes returned in the previous step. In this step, we automatically apply the following operations according to the specialization polysemy sub class:

   (a) *Adding missing relation*: We apply this operation on the elements in the missing relation synsets sub class.

(b) *Adding a missing parent*: We apply this operation on the elements in the missing parent synsets sub class.

(c) *Synset merging*: We apply this operation on the elements in the too fine grained senses, redundant, and sense enumeration sub class.

7.2 S2: Solving the Problem of Unspecified Information in WordNet Algorithm

The task in this stage is to explicitly denote homonymy and metaphoric instances that were identified in steps S1.P2.1.2 and S1.P2.1.3. The input of the algorithm is WordNet structure after applying the specialization operations and the metaphoric and homonymy instances returned at the end of S1.P2.1.3 (items 2 and 3). The output is the resulting structure after denoting these instances explicitly as described below.

S2.P1 **Homonymy and metaphoric polysemy discovery**
This phase is included in phase S1.P1.1.

S2.P2 **Homonymy and metaphoric polysemy organization**
In this phase, we organize the metaphoric and homonymy instances by denoting these instances via the following semantic relations.

1. *is_homograph*: We use the relation is_homograph to denote homonymy between homonymy terms.
2. *is_metaphor*: In metaphoric instances, we use the relation is_metaphor to denote the metaphoric relation between the metaphoric meaning and literal meaning of a metaphoric term.

Chapter 8

WordNet Data Structures

In the following, we give formal definitions for the data structures used in our approach

8.1 Basic Data structures

Lemma, the basic unit in WordNet is defined in wordNet documentation as follows: *a lower case ASCII text of word as found in the WordNet database index files*. Usually the base form of a word or collocation. Based on this definition, we consider lemma as a single word or a collocation that corresponds to the orthographic string representation of natural language terms. A natural language term or simply a term belongs to a grammatical category; i.e., noun, verb, adjective or adverb. We define terms as follows.

Definition 1 *(Term).*

A term T is a quadruple $\langle$Lemma, Cat, T-Rank$\rangle$, where

a) Lemma is the term lemma, i.e., the orthographic string representation of the term;

b) Cat $\in$ {noun, verb, adjective, adverb} is the grammatical category of the term;

c) T-Rank is the term rank, i.e., a natural number >0.

d) T-Relations $\subset Term \times Term$ is a set of lexical relations.

T-Rank is used to reflect which is the preferred term of a synset. For example, `man` and `adult male` in the following synset correspond to the following term instances: $\langle$Lemma: "man", Cat: noun, T-Rank: 1$\rangle$ and $\langle$Lemma: "adult male", Cat: noun, T-Rank: 2$\rangle$.

#1 `man, adult male: an adult person who is male (as opposed to a woman).`

The set of T-Relations correspond the lexical relations in WordNet. For example, the lexical relation `antonym` holds between the terms `love` and `hate`. Another example, is the relation `is_homograph` that we propose to denote homonymy in WordNet.

In the following, we define wordNet synsets.

Definition 2 *(WordNet synset).*

A synset S is defined as $\langle$Cat, Terms, Label, Gloss, Relations, Genus, Differentia, S-Rank$\rangle$, where

a) Cat $\in$ {noun, verb, adjective, adverb } is the grammatical category of the synset ;

b) Terms is an ordered list of synonymous terms that have the same grammatical category as the synset grammatical category;

c) Label $\in$ Ts is the preferred term of the synset, i.e., the term whose T-Rank = 1;

d) Gloss is a natural language text that describes the synset;

e) Relations is a set of semantic relations that hold between synsets;

f) S-Rank is the synset rank, i.e., a natural number >0 that reflects the familiarity of the synset;

g) Genus is a synset that represents the genus in the synset gloss;

h) Differentia corresponds to one synset or more that represent the differentia in the synset gloss.

The synset #2 in the following example correspond to the synset instance in Figure 8.1:

#2 `woman, adult female: an adult female person (as opposed to a man); "the woman kept house while the man hunted".`

Cat: noun
Terms: {<Lemma: "woman", Cat: noun, T-Rank: 1>,
<Lemma: " adult female ", Cat: noun, T-Rank: 2>}
Label: <Lemma: "woman", Cat: noun, T-Rank: 1>
Gloss: "an adult female person (as opposed to a man)"
Relations : [...]
S-Rank: 2
Genus: $[\ldots]_{\text{female}}$
Differentia: $\{[\ldots]_{\text{adult}}\}$
woman

Figure 8.1: An example of synset instance

Genus and differentia in the synset definition correspond to the implicit encoded genus and differentia in the synset gloss. They are not formally defined in WordNet. Notice that the synset and its genus should belong to the same grammatical category. This is not required for differentia. For example, `ricotta` and its genus (`cheese`) in the following synset are nouns, while the differentia contains two adjectives `soft` and `Italian`.

#1 `ricotta: soft Italian cheese.`

The synset rank is relevant if one of the synset terms belongs to the terms of other synsets, i.e., the synset contains a polysemous term. In such cases, S-Rank reflects which is the the preferred sense of the polysemous term. Notice that the synset rank is relative to the polysemous term. Each polysemous synset is the preferred sense of one polysemous term at most. For example, all terms of the following synset are polysemous, but it is not the preferred sense of any of them.

`grinding, abrasion, attrition, detrition: the wearing down of rock particles by friction due to water or wind or ice.`

The set Relations correspond to the semantic relations used by WordNet to organize the relations between the synsets as explained in section 2.3.

8.2 WordNet Hierarchy

WordNet uses the relation hypernym and hyponym, the counter relation of hypernym to organize the hierarchical relations between the synsets. These relations denote the superordinate/subordinate relationship between synsets.

Definition 3 *(direct hypernym/hyponym relation).*

Let S = $\{s_1, s_2, \cdots, s_n\}$ the set of noun synsets in WordNet. Let R_{WN} be the set of wordNet relations. The relations hypernym/hyponym $\subseteq S \times S$ are defined as follows. For two synsets $s_k, s_l \in S$: s_k is a direct hypernym of s_l if $\langle s_k, hypernym, s_l \rangle \in R_{WN}$. s_l is a direct hyponym of s_k if s_k is direct hypernym of s_l.
For example, the relation direct hypernym/hyponym hold between `vehicle` and `wheeled vehicle` where `vehicle` is hypernym of `wheeled vehicle` and `wheeled vehicle` is hyponym of `vehicle`.

`vehicle: a conveyance that transports people or objects.`

`wheeled vehicle: a vehicle that moves on wheels and usually has a container for transporting things or people.`

The hypernym/hyponym relations correspond to superordinate/subordinate relations. The superordinate/subordinate relationship is transitive. In the following, we generalize the direct hypernym/hyponym relation to reflect the transitivity property, where we use the notion hypernym/hyponym instead of a direct hypernym/hyponym.

Definition 4 *(hypernym/hyponym relation).*

For two synsets s and $s^{'}$, s is a hypernym of $s^{'}$, if the following holds: s is a direct hypernym of $s^{'}$, or there exists a synsets $s^{''}$ such that s is a direct hypernym of $s^{''}$ and $s^{''}$ is a hypernym of $s^{'}$. s is a hyponym of $s^{'}$ if and only if $s^{'}$ is a hypernym of s.
For example, `vehicle` is a hypernym of `car`, because `vehicle` is direct hypernym of `wheeled vehicle` and `wheeled vehicle` is a direct hypernym of `car`.
Notation
We use the following symbols to denote hypernym/hyponym relations:

a) $s \prec s^{'}$ if s is a direct hypernym of $s^{'}$

b) $s \succ s^{'}$ if s is a direct hyponym of $s^{'}$

c) $s \prec^{*}$ if s is a hypernym of $s^{'}$

d) $s \succ^{*}$ if s is a hyponym of $s^{'}$

Using the direct hypernym relation, wordNet organizes noun-synsets in a hierarchy. We define the hierarchy of WordNet in noun-synsets as follows:

Definition 5 *(wordNet hierarchy).*

Let $S = \{s_1, s_2, ..., s_n\}$ be the set of noun-synsets in WordNet. WordNet hierarchy is defined as a connected and rooted digraph $\langle S, E \rangle$, where

a) `entity` $\in S$ is the single root of the hierarchy;

b) $E \subseteq S \times S$;

c) $(s_1, s_2) \in E$ if $s_1 \prec s_2$;

d) For any synset s $\neq$ `entity`, there exists at least one synset $s^{'}$ such that $s^{'} \prec s$.

In this definition, point (a) defines the single root of the hierarchy and point (d) defines the connectivity property in the hierarchy.

8.3 Semantic Definitions

The relation $\prec$ defines the hierarchical structure of WordNet but not enough to define its semantics. The relation defines the genus of a concept which is a part of the semantics of a concept. The differentia is usually implicit in the synset glosses. For example, $\prec$ defines the relation between `person` and `grammatical category` explicitly. The relation between `person` and `pronouns` or verb forms remains implicit.

```
person -- (a grammatical category of pronouns and verb forms; "stop talking
   about yourself in the third person")
     => grammatical category, syntactic category -- ((grammar) a category of
        words having the same grammatical properties)
```

In the following, we define a subset of the semantics of WordNet hierarchy that is relevant for our approach. Full definition of wordNet semantics is described in approaches such as [52] [53] [54] .
We define the semantics of WordNet using an Interpretation $I = \langle \Delta^I, f \rangle$, where Δ^I is an non empty set (the domain of interpretation) and f is an

interpretation function. In this definition, we define the semantics of a synset in terms of the genus and differentia of the synset glosses.

Definition 6 *(Semantics of WordNet Hierarchy).*

Let $WH = \langle S, E \rangle$ be wordNet hierarchy. We define an Interpretation of WH, $I = \langle \Delta^I, f \rangle$ as follows:

a) $entity^I = \Delta^I$

b) $\perp^I = \varnothing$

c) $\forall s \in S$: $s^I \subset \Delta^I$

d) $s^I = (s.genus)^I \sqcap (s.differentia)^I$

e) $(s_1 \sqcap s_2)^I = s_1^I \cap s_2^I$

f) $(s_1 \sqcup s_2)^I = s_1^I \cup s_2^I$

g) $s_1 \equiv s_2$ if $(s_1.genus)^I = (s_2.genus)^I$ and $(s_1.differentia)^I = (s_2.differentia)^I$

h) $s_1 \sqsubseteq s_2$ if $s_1^I \subseteq s_2^I$

In points a) and b), we define the empty and universal concepts. Point c) states that Δ^I is closed under the interpretation function f. In point d), we define the semantics of a synset as the conjunction of its genus and differentia. Notice that the synset genus is usually equal to its hypernym. In and e) and f), we define the conjunction and disjunction operations. In g) and h), we define synset equivalence and subsumption relations. These relations play an important role in specialization polysemy organization. Notice that in most cases $s_1 \prec^* s_2$ implies that $s_2 \sqsubseteq s_1$ and vice versa. For example, `social group` $\prec^*$ `family` and `family` $\sqsubseteq$ `social group`.

```
family, family unit -- (primary social group; parents and children; "he wanted
    to have a good job before starting a family")
       => kin, kin group, kinship group, kindred, clan, tribe -- (group of
           people related by blood or marriage)
           => social group -- (people sharing some social relation)
```

8.4 Polysemy Data Structures

A term is polysemous if it is found in the terms of more than one synset. A synset is polysemous if it contains at least one polysemous term. In the following, we define polysemous terms.

Definition 7 *(polysemous term).*

A term t = ⟨Lemma, Cat, T-Rank⟩ is polysemous if there is a term $t^{'}$ and two synsets s and $s^{'}$, $s \neq s^{'}$ such that

a) t ∈ s.Terms and $t^{'}$ ∈ $s^{'}$.Terms

b) t.Lemma = $t^{'}$.Lemma

c) t.Cat = $t^{'}$.Cat.

In definition 7, we exclude the syntactic ambiguous terms. In the following, we define polysemous synsets.

Definition 8 *(polysemous synset).*

A synset s is polysemous if any of its terms is a polysemous term.
It is possible for two polysemous synsets to share more than one term. Two polysemous synsets and their shared terms constitute a polysemy instance. In the following, we define polysemy instances.

Definition 9 *(polysemy instance).*

A polysemy instance is a triple $[\{T\}, s_1, s_2]$, where s_1, s_2 are two polysemous synsets that have the terms {T} in common.
For example, the term `bazaar` belongs to the following polysemy instances: $[\{bazaar, bazar\}, \#1, \#2]$, $[\{bazaar\}, \#1, \#3]$, and $[\{bazaar\}, \#2, \#3]$.

#1 `bazaar, bazar: a shop where a variety of goods are sold.`

#2 `bazaar, bazar: a street of small shops (especially in Orient).`

#3 `bazaar, fair: a sale of miscellany; often for charity.`

Notice that the polysemy instances c1 $= [\{T\}, s_1, s_2]$ and c2 $= [\{T\}, s_2, s_1]$ are considered to be one polysemy instance.

8.5 Structural Patterns Data Structures

We exploit the structural properties in WordNet hierarchy to identify the polysemy classes of the polysemy instances in WordNet. Our hypothesis is that polysemy instances that are similar in their structural properties or belong to the same structural pattern belong also to the same polysemy class. In the following, we illustrate the definitions that we use in our approach to define structural patterns.
In the following, we illustrate the essential structural definitions in our approach. We start with structural path that we define as follows.

Definition 10 *(structural path).*

Let $s, s^{'}$ be two synsets in wordNet. Let $s \prec^* s^{'}$. The structural path between s, and $s^{'}$ is defined as a sequence $(s_0, s_1), \cdots, (s_{n-1}, s_n)$ such that $s_0 = s, s_n = s^{'}$ and for any $i, 0 \leqslant i < n, s_i \prec s_{i+1}$.

According to the connectivity property of wordNet hierarchy in definition 5, any two synsets in wordNet have at least one common subsumer that we define as follows.

Definition 11 *(common subsumer).*

Let s_1, s_2, and s be synsets in wordNet. The synset s is a common subsumer of s_1 and s_2 if $s \prec^* s_1$ and $s \prec^* s_2$.
WordNet hierarchy is a digraph. This implies that it is possible for two synsets to have more than one common subsumer. To define the least common subsumer, we need to define the synset height in wordNet which we define as follows.

Definition 12 *(synset height).*

Let s be a synset in wordNet. Let $(s_0, s_1), \cdots, (s_{n-1}, s_n)$ be the structural path where $s_0 = entity$ and $s_n = s$. The synset height of s denoted as $\uparrow s \uparrow = n$.
In the following, we define the least common subsumer of two synsets in WordNet as follows.

Definition 13 *(least common subsumer).*

Let s_1, s_2, s be synsets in wordNet. Let $C = \{c_1, \cdots, c_n\}$ be the set of common subsumers of s_1 and s_2. The least common subsumer of s_1 and s_2 is defined as the synset $c_i \in C$ such that $\forall c_j \in C, i \neq j : \uparrow c_j \uparrow \ < \ \uparrow c_i \uparrow$.
Notice that any two synsets in WordNet have a least common subsumer.
In the following, we define structural patterns.

Definition 14 *(structural pattern).*

A structural pattern of polysemy instance $\mathrm{I} = [\ \{T\}, s_1, s_2]$ is a triple $P = \langle r, p_1, p_2 \rangle$, where

a) r is the least common subsumer of s_1 and s_2;

b) $p_1 \succ r$ and $p_2 \succ r$;

c) $p_1 \prec^* s_1$ and $p2 \prec^* s_2$

We call r *the pattern root* and p_1, p_2 *the pattern hyponyms.* For example, the structural pattern of the polysemy instance $[\ \{bazaar, bazar\}, s_1, s_2]$ is $\langle mercantile\quad establishment, marketplace, shop\rangle$ as shown in Figure 8.2. Notice that the patterns $p = \langle r, p_1, p_2\rangle$ and $q = \langle r, p_2, p_1\rangle$ are considered to

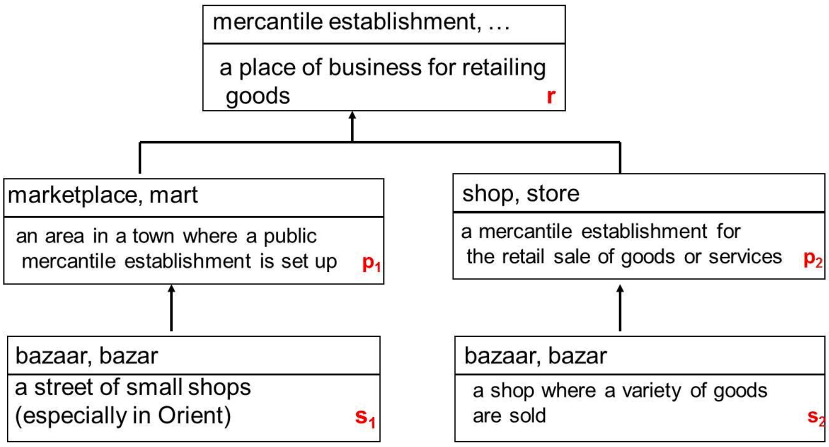

Figure 8.2: Example of a structural pattern

be one and the same pattern. We denote p and q through a pattern label which we define as follows.

Definition 15 *(pattern label).*

A patterns $P = \langle r, p_1, p_2\rangle$ (or $\langle r, p_2, p_1\rangle$) is denoted through the pattern label "$L_r\#\langle L_{p1}, L_{p2}\rangle$, where

a) L_r is the label of the synset r;

b) L_{p1} is the label of the synset p_1;

c) L_{p2} is the label of the synset p_2;

For example, the pattern label of the pattern in Figure 8.2 is "$mercantile\quad establishment\#\langle marketplace, shop\rangle$".

The pattern label of a pattern and all polysemy instances under that pattern constitute structural pattern class. We define structural pattern class as follows.

Definition 16 *(structural pattern class).*

For a pattern p, we define a structural pattern class $pc = \langle label, instances \rangle$, where

a) label is the pattern label of p;

b) instances is a list of all polysemy instances that belong to p.

8.6 Regular Structural Patterns

According to Apresjan's definition, regular structural patterns are those patterns whose corresponding structural pattern classes contain two polysemy instances at least. Definition 14 is good to discover all regular polysemy patterns at the upper and middle level in WordNet hierarchy. However, it is not suitable to capture all regular structural patterns at the lower level in WordNet hierarchy. The polysemy instances at the lower level correspond usually to the instances, in which the two synsets are direct hyponyms of the same common parent. The structural pattern of two synsets s_1, s_2 that share the same common parent has the form $\langle r, s_1, s_2 \rangle$. The number of the polysemy instances in such patterns is of course less than two instances and thus cannot be considered as a regular pattern.
To capture the structural regularity at the lower level in WordNet hierarchy, we define the common parent structural pattern. In the following, we define common parent structural pattern and the common parent structural pattern class, where we generalize the definition so that it corresponds to all instances in which r is the direct hypernym of at least one synset in a polysemy instance as illustrated in Figure 8.3.

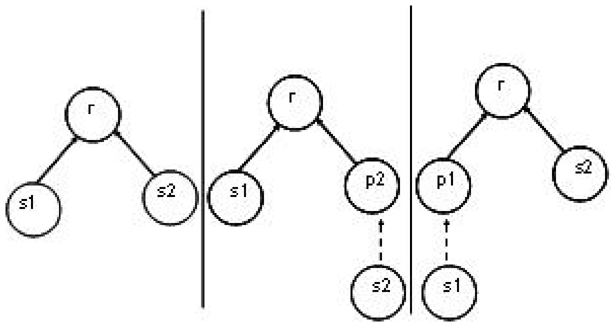

Figure 8.3: Common parent structural pattern

Definition 17 *(common parent structural pattern).*

A polysemy instance I = [$\{T\}, s_1, s_2$] belongs to the common parent structural pattern if its structural pattern p=$\langle r, p_1, p_2\rangle$ has one of the following forms $\langle r, s_1, s_2\rangle$, $\langle r, s_1, p_2\rangle$ or $\langle r, p_1, s_2\rangle$.
Since the instances of structural common parent classes are most usually singleton sets, we define the common parent structural pattern class that contains the polysemy instances of all common parent structural patterns.

Definition 18 *(common parent structural pattern class).*

We define the common parent structural pattern as $\langle label, instances\rangle$, where

a) label = "common parent";

b) instances is a list of all polysemy instances that belong to a common parent structural pattern.

Based on definition 18 and 18, we define regular structural patterns as follows.

Definition 19 *(regular structural pattern).*

A structural pattern is regular if the following holds:

a) It is a common parent structural pattern; or

b) The number of the instances of its structural pattern class $\geqslant 2$.

8.7 Type Compatible/Incompatible Structural Patterns

WordNet hierarchy represents a classification hierarchy where the synsets are the nodes in this hierarchy. Classification hierarchies should fulfill among other requirements the exclusiveness property [55] that we define as follows.

Definition 20 *(Exclusiveness property).*

Two synsets $s_1, s_2 \in S$ fulfill the exclusiveness property if $s_1^I \sqcap s_2^I = \bot^I$.

For example, `abstract entity` and `physical entity` fulfill the exclusiveness property. On the other hand `expert` and `scientist` do not fulfill this property because $expert^I \sqcap scientist^I \neq \bot^I$.

The exclusiveness property means that any two sibling nodes n_i, nj in the hierarchy are disjoint, i.e., $n_i^I \not\sqsubseteq n_j^I$ and $n_j^I \not\sqsubseteq n_i^I$. Analyzing the structural patterns in WordNet shows that the exclusiveness property is not always guaranteed in WordNet. For example, the pattern $\langle person, expert, scientist \rangle$ shown in Figure 8.4 does not fulfill the exclusiveness property because forcing this property would result in preventing a scientist to be an expert or an expert to be a scientist.

We use the exclusiveness property and the pattern root in a structural pattern to discover specialization polysemy candidates indirectly. The relation between the synsets in specialization polysemy is hierarchical. The hierarchical relation between the synsets in a specialization polysemy instance indicates that the exclusiveness property does not hold between synsets and thus between the structural pattern hyponyms. For example, the two synsets of the term `statistician` in Figure 8.4 constitute a specialization polysemy instance of the structural pattern $\langle person, expert, scientist \rangle$. We call the structural patterns that do not fulfill the exclusiveness property *type compatible structural patterns*. We call the polysemy instances that

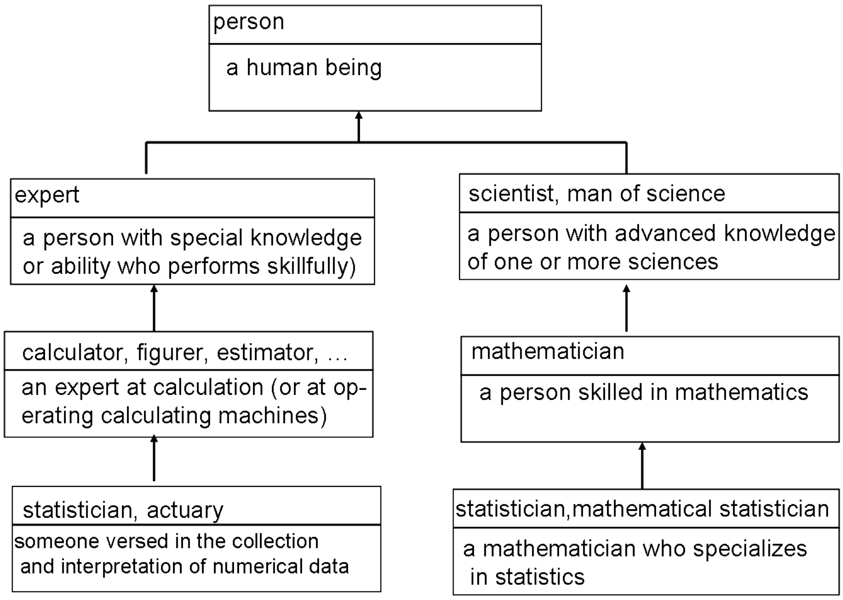

Figure 8.4: A specialization polysemy instance

belong to such patterns *type compatible polysemy instances*. On the other hand, there are many structural patterns in wordNet hierarchy that fulfill the exclusiveness property. For example, the pattern $\langle entity, physical\ entity, abstract\ entity\rangle$ fulfills the exclusiveness property because `physical entity` and `abstract entity` are disjoint. We call the structural patterns that fulfill the exclusiveness property *type incompatible structural patterns*. We call the polysemy instances that belong to such patterns *type incompatible polysemy instances*. Notice that it is possible for a polysemous term to have type compatible and type incompatible polysemy instances. For example, the polysemy instance $[\{acquirer\}, \#2, \#3]$ is a type compatible polysemy instance of the term `acquirer` since the synsets #2 and #3 do not fulfill the exclusiveness property because `credit card processing bank` is a `financial institution`. At the same time, the polysemy instances $[\{acquirer\}, \#1, \#2 >]$ and $]\{acquirer\}, \#1, \#3]$ are type incompatible polysemy instances because $\#1 \sqsubset physical \quad entity$, and $\#2 \sqsubset abstract \quad entity$ and $abstract \quad entity \sqcap physical \quad entity = \bot$. The same holds for the synsets #1 and #3.

#1 `acquirer: a person who acquires something (usually permanently).`

#2 `acquirer: the financial institution that dispenses cash in automated teller machines...`

#3 `merchant bank, acquirer: a credit card processing bank; merchants receive credit for credit card receipts less a processing fee.`

An important point here is that the polysemy instances of a structural pattern that fulfills the exclusiveness property are not necessarily homonymy instances. The exclusiveness property is not a requirement for metonymy and metaphoric polysemy. At the same time, not all type compatible structural patterns are specialization polysemy patterns. Type compatible structural patterns include also metaphoric structural patterns. Differentiating between specialization polysemy and metaphoric structural patterns is explained in details in chapter 11.

We turn now to the relation between the pattern root and the definition of type compatible/incompatible structural patterns. The pattern root of the structural pattern $\langle person, expert, scientist\rangle$ in the previous example is the root of many other structural patterns. Some of these patterns are type incompatible such as $\langle person, capitalist, enrollee\rangle$. However, the pattern root `person` is a hypernym of at least one structural pattern that does not fulfill the exclusiveness property.

In general, we can also observe that the common parent structural patterns do not fulfill the exclusiveness property. On the other hand, we find structural pattern roots in WordNet such that all the patterns that belong to these roots fulfill the exclusiveness property. We call such pattern roots type incompatible roots. Consider for example, the root of wordNet hierarchy `entity`. We define type incompatible roots as follows.

Definition 21 *(type incompatible roots).*

A synset r is a type incompatible root if the exclusiveness property holds between all its direct hyponyms.
Based on the previous definition, we define type incompatible structural patterns as follows.

Definition 22 *(type incompatible structural pattern).*

A pattern $p = \langle r, p1, p2 \rangle$ is type incompatible pattern, if the pattern root r belongs to type incompatible roots, otherwise p is type compatible.

In the following, we consider the set of the synsets that reside in the first and second level in WordNet hierarchy as a subset of the type incompatible roots in wordNet. This set contains the following synsets: {`entity, abstract entity, abstraction, physical entity, physical object` }. These synsets are not the only type incompatible roots in WordNet. We may find other type incompatible roots in the other levels of the hierarchy. Determining the synsets in the first and second level of the hierarchy as type incompatible roots is important because these roots enable us to automatically determine specialization polysemy candidates by excluding all type incompatible polysemy instances that belong to the structural patterns of these roots.
An important question here is why we have chosen level 2 and not level 3 or beyond to determine type incompatible roots? The answer is that the exclusiveness property is not guaranteed for all structural patterns whose roots reside in the third level and beyond. For example, the pattern root of the structural pattern $\langle substance, food, solid \rangle$ resides in the third level of wordNet hierarchy. It is clear that forcing the exclusiveness property would result in preventing food to be solid substance. Consider for example the following specialization polysemy instance $[\{cake\}, \#1, \#2]$ that

belongs to this pattern.

#1 `patty, cake: small flat mass of chopped food.`

#2 `cake: made from or based on a mixture of flour and sugar and eggs.`

Based on definition 22, we classify polysemy instances into type compatible and type incompatible as follows.

Definition 23 *(type incompatible polysemy instance).*

A polysemy instance is type incompatible if it belongs to a type incompatible pattern, otherwise it is type compatible.

Chapter 9

Compound Noun Polysemy Organization

In this approach, we consider that using a noun adjunct/modified noun to refer to its corresponding compound noun is similar to the use of anaphoric pronouns. In this sense, we may call a noun adjunct/modified noun that refers to a compound noun an *anaphoric term*.

Anaphoric pronouns and anaphoric terms are similar in the following aspects:

1. Anaphoric pronouns and anaphoric terms are usually used to avoid repetition of the same word.

2. Anaphoric pronouns and anaphoric terms are usually ambiguous.

3. Using and understanding of anaphoric pronouns and anaphoric terms depends on a term that precedes them.

4. Anaphoric pronouns and anaphoric terms usually need a disambiguation process to bind them to their corresponding referred term in the discourse.

In point 3, the discourse dependency of anaphoric terms means that an anaphoric term is used to refer to another (explicit or implicit) term in

the context that enables disambiguating the reference term. This is very important, because without (the explicit or implicit) referred term, the anaphoric term has no meaning or its meaning can not be disambiguated. We think that the referred term is the compound noun. That means using and understanding the reference term is dependent on a compound noun that can be understood from the discourse.
Similar to anaphoric pronouns in point 4, anaphoric terms need to be disambiguated. Anaphoric pronoun disambiguation is called *anaphoric resolution* which is a syntactic process that binds the pronouns to their corresponding referred terms. What is about the process of anaphoric term disambiguation? Is it different from the anaphoric pronoun disambiguation? Do we need to list all anaphoric term as synonyms to their corresponding compound nouns? Lets consider the following example.
The term `head` is not synonymous to `nail head` in WordNet. The term `nail head` has the following meanings in WordNet.

#1 `nailhead: something resembling the head of a nail that is used as an ornamental device.`

#2 `nailhead: flattened boss on the end of nail opposite to the point.`

Assuming that an NLP tool that uses wordNet as a lexicon is analyzing sentences like the following two sentences:

1 `John was playing with a nail. The head injured him.`

2 `A Turing machine has a read/write head. It uses the head to write on its tape.`

The term `head` appears in both sentences. The question now: Will the NLP tool fail to disambiguate the term `head` in the first sentence because it is not synonymous to `nail head` and succeed to disambiguate the second sentence? If the answer yes, wordNet needs a major improvement in which

noun adjuncts and modified nouns are added as synonyms to their corresponding compound nouns. For example, `head` should be synonymous to the compound nouns `nail head`, `spear head`, `department head`, $\cdots$ If the answer no, then we can remove all reference terms from wordNet without affecting the efficiency of the NLP tools that are based on WordNet.
In this approach, we argue that reference term disambiguation is similar to pronoun disambiguation. That means, removing the anaphoric terms in all compound noun polysemy cases reduces the sense enumerations in WordNet without affecting its efficiency as a lexical resource for NLP tools. In the following, we give formal definitions.

Definition 24 *(compound noun polysemous term).*

A term t is compound noun polysemous term of a term t' if t is the noun adjunct or the modified noun of t'.

Definition 25 *(compound noun polysemous synset).*

A synset s is compound noun polysemous if it contains a compound noun polysemous term.
The synsets in the previous example are compound noun polysemous. In the following, we define compound noun polysemy instance.

Definition 26 *(compound noun polysemy instance).*

A polysemy instance $I = [\{T\}, s_1, s_2]$ is compound noun polysemy instance if s_1 or s_2 is a compound noun polysemous synset.

9.1 Compound Noun Polysemy Discovery

In the following, we present `discoverCompoundNounPolysemyInstances` algorithm that we use to discover compound noun polysemy instances. The input of the function is wordNet hierarchy and the output is a list of compound noun polysemy instances.

```
10 WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 PolysemyInstance: struct of {terms: {Term}, s1: Synset, S2: Synset};
40 polyInstances: list of PolysemyInstance;
50 compoundNounPolyInstances: list of polysemyInstance;
60 function discoverCompoundNounPolysemyInstances(){
70 polyInstances := getPolyInstances(wH);
80 foreach polyInstance in polyInstances do {
90  if(isCompoundNounPolysemyInstance(polyInstance)) then {
100   compoundNounPolyInstances.add(polyInstance);}
110}
120 return compoundNounPolyInstances;}
```

Figure 9.1: Pseudo-code of the compound noun polysemy discovery algorithm

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `polyInstance`: A variable of type `PolysemyInstance`.

3. `polyInstances`: A list of `PolysemyInstance` in wordNet.

4. `compoundNounPolyInstances`: A list of `PolysemyInstance` to store compound noun polysemy instances in WordNet.

The input/output of the function:

- The input: `wH`.
- The output: `compoundNounPolyInstances`.

The function works as follows:

1. The function retrieves the polysemy instances in wordNet via the function `getPolyInstances` (1ine 70) which is described in figure 10.2.
2. The function iterates over the retrieved polysemy instance and performs the following (line $80-100$):
   (a) It tests if the polysemy instance is a compound noun polysemy instance by calling the test function `isCompoundNounPolysemyInstance` (line 90) which is described in figure 9.2.
   (b) It stores compound noun polysemy instances in the list `compoundNounPolyInstances` (line 100).
3. The function returns the list `compoundNounPolyInstances`, the output of the function (line 120).

In the following, we present the function `isCompoundNounPolysemyInstance` . The function uses the following data structures:

1. `Term`: A term as defined in definition 1.
2. `Synset`: A synset as defined in definition 2.
3. `PolysemyInstance`: A polysemy instance as defined in definition 9.

The function uses the following variables:

1. `p`: A variable of Type `PolysemyInstance`.
2. `s1, s2`: Variables of Type `Synset`.
3. `terms`: A list of `Term`.

```
10 Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
20 Synset: struct of {terms: {Term}, label: Term, gloss: string,
        relations: {Relation}, s-rank: integer};
30 PolysemyInstance: struct of {terms: {Term}, s1: Synset, S2: Synset};
40 terms: list of Term;
50 s1, s2: Synset;
60 function isCompoundNounPolysemyInstance(PolysemyInstance p){
70 terms := p.terms; s1:= p.s1; s2:=p.s2;
80 return (isCompoundNounPolysemousSynset(terms,s1))
|| isCompoundNounPolysemousSynset(terms,s2));}
```

Figure 9.2: Pseudo-code for testing compound noun polysemy instances

The input/output of the function:

- The input: `p`.
- The output: `true` or `false`.

The function tests if a polysemy instance is a compound noun polysemy instance according to definition 26. A polysemy instance is a compound noun polysemy instance if one of its synsets is compound noun polysemous in respect to the polysemy instance terms. The function calls the function `isCompoundNounPolysemousSynset` which is described in the Figure 9.3.
In the following, we present the function `isCompoundNounPolysemousSynset`.
The function uses the following data structures:

1. `Term`: A term as defined in definition 1.
2. `Synset`: A synset as defined in definition 2.

The function uses the following variables:

1. `s`: A variable of Type `Synset`.
2. `terms`: A list of `Term`.

```
10 Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
20 synsetTerms : a list of Term;
30 Synset: struct of {terms: {Term}, label: Term, gloss: string,
        relations: {Relation}, s-rank: integer};
40 function isCompoundNounPolysemousSynset(terms: list of Term, Synset s){
50 synsetTerms := s.terms;
60 foreach term in terms do{
70  foreach term1 term in synsetTerms do{
80   if(isCompoundNounPolysemousTerm(term,term1))then{
90       return true;}}
100 return false;}
```

Figure 9.3: Pseudo-code for testing compound noun polysemy synsets

3. `synsetTerms`: A list of `Term`.

The input/output of the function:

- The input: `terms`, `s`.
- The output: `true` or `false`.

The function tests if a polysemy synset is a compound noun polysemous according to definition 25. The test is performed via the function `isCompoundNounPolysemousTerm` as presented in Figure 9.4.

```
10 Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
20 function isCompoundNounPolysemousTerm(Term t1, Term t2){
30 return is_prefix(t1.lemma,t2.lemma) || is_suffix(t1.lemma, t2.lemma);}
```

Figure 9.4: Pseudo-code for testing if a term is compound noun

The function uses the following data structures:

1. `Term`: A term as defined in definition 1.

The function uses the following variables:

1. `t1`, `t2`: Variables of Type `Term`.

The input/output of the function:

- The input: `t1`,`t2`.
- The output: `true` or `false`.

The function tests if a term is compound noun polysemous in respect to another term according to definition **??**. The test is performed as a string operation using the string functions `is_prefix` and `is_suffix` that test if a string is a prefix or a suffix of another string.

9.2 Compound Noun Polysemy Manual Validation

The input of this phase is the output of the algorithm `discoverCompound-NounPolysemyInstances`. The task of this phase is to exclude false positive instances. False positive instances here belong to the following groups:

1. **Term abbreviations**: Since the algorithm in the previous step uses the string function to test compound noun polysemy, the algorithm returns polysemy instances that include term abbreviations as compound noun polysemy instances. For example, the term `mil` is abbreviation of the terms `milliliter` and `millilitre` in the following synset.

   \# `milliliter, millilitre, mil, ml, cubic centimeter, cubic centimetre, cc: a metric unit of volume equal to one thousandth of a liter.`

2. **Specialization polysemy instances**: Polysemy instances that indicate a hierarchical relation do not belong to the compound noun polysemy instances. For example, the following two synsets of the term `laver`:

   \#1 `red laver, laver: edible red seaweeds.`

#2 `sea lettuce, laver: seaweed with edible translucent crinkly green fronds.`

3. **Metonymy polysemy instances**: Metonymy polysemy instances are excluded. For example, the following two synsets of the term `cherry`

   #2 `cherry, cherry tree: any of numerous trees and shrubs producing a small fleshy round fruit with a single hard stone; many also produce a valuable hardwood.`

   3 `cherry: a red fruit with a single hard stone.`

4. **Nouns with ing inflected forms**: Polysemy instances that correspond to ing inflected form are excluded. For example, the following synset of the term `feel`

   # `spirit, tone, feel, feeling, flavor, flavour, look, smell: the general atmosphere of a place or situation and the effect that it has on people; "the feel of the city excited him"; "a clergyman improved the tone of the meeting"; "it had the smell of treason".`

5. **Nouns with alternative forms** Terms with alternative forms such as `ful` are excluded. For example, the following synset of the term `bottle`

   # `bottle, bottleful: the quantity contained in a bottle.`

6. **Missing adjunct noun/modified noun synset**: In some cases, a synset of the adjunct noun or the modified noun is missing. Such cases are excluded. For example, non of the 6 synsets of the term `party` can be considered as a general meaning of the term `political party` in the following synset.

   # `party, political party: an organization to gain political power.`

9.3 Compound Noun Polysemy Disambiguation

The input of this step is a list of compound noun polysemy instances. The task in this phase is to disambiguate these instances. Disambiguating here means removing the noun adjunct or the modified noun from the synset terms of these instances. In the following, we present the algorithm compoundNounPolysemyDisambiguation.

```
WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
wH: WordNetHierarchy;
Term: struct of {lemma: string; cat: grammatical category; t-rank: integer};
Synset: struct of {terms: {Term}, label: Term, gloss: String,
        relations: {Relation}, s-rank: integer};
PolysemyInstance: struct of {terms: {Term}, s1: Synset, S2: Synset};
compoundNounPolyInstances: list of polysemyInstance;
function compoundNounPolysemyDisambiguation(){
 foreach polyInstance in compoundNounPolyInstances do{
  Synset s1 := polyInstance.s1;
  Synset s2 := polyInstance.s2;
   {Term} terms := polyInstance.terms;
  if(isCompoundNounPolysemousSynset(terms,s1)) then {
    disambiguate(terms, s1);}
   if(isCompoundNounPolysemousSynset(terms, s2)) then {
   disambiguate(terms, s2);}
}
```

Figure 9.5: Pseudo-code for compound noun polysemy disambiguation

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Term`: A term data structure as defined in definition 1.
3. `Synset`: A synset data structure as defined in definition 2.
4. `PolysemyInstance`: A polysemy Instance as defined in definition 9.

The function uses the variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `compoundNounPolyInstances`: a list of compound noun polysemy instances in wordNet.
3. `polyInstance`: A variable of type `PolysemyInstance`.
4. `s1, s2`: Variables of type `Synset`.
5. `terms`: A variable that represents the polysemous terms in `PolysemyInstance`.

The input/output of the function:

- The input: `compoundNounPolyInstances`.
- The output: No output, the operations are performed on `wH`.

The function works as follows:
It iterates on each of the input polysemy instances in `compoundNounPolyInstances` (line 90-160).

a. If the first and/or the second synset of the current operated polysemy instance is compound noun polysemous according to definition **??** (line 130) and (line 150).

b. Compound noun polysemous terms according to definition 24 are disambiguated (line 140) and (line 160). This operation is performed via the function `disambiguate` which is described in Figure 9.6.

In the following, we present the function disambiguate .
The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Term`: A term data structure as defined in definition 1.

```
WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
wH: WordNetHierarchy;
Term: struct of {lemma: string; cat: grammatical category; t-rank: integer};
synsetTerms: list of term;
Synset: struct of {terms: {Term}, label: Term, gloss: String,
        relations: {Relation}, s-rank: integer};
function disambiguate(polyTerms: list of Term, Synset s){
 synsetTerms := synset.terms;
 foreach term in polyTerms do {
   foreach term1 != term in synsetTerms do{
    if(isCompounNounPolysemousTerm(term, term1)) then{
  s.terms := s.terms\{term};}
 }}
}
```

Figure 9.6: Pseudo-code for the disambiguation operation

3. `Synset`: A synset data structure as defined in definition 2.

The function uses the variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `compoundNounPolyInstances`: a list of compound noun polysemy instances in wordNet.

3. `polyTerms`, `synsetTerms`: A list of `Term`.

4. s: A variable of type Synset.

The input/output of the function:

- The input:`polyTerms`, `s`.

- The output: No output, the operations are performed on `wH`.

The function works as follows:

1. The function iterates over the input terms and checks if any of them is a compoun noun polysemous term according to definition 24.
2. Discovered compound noun polysemous terms are removed from the terms of the input synset s (line 120).

For example, the result of applying the function on head and the synset #8 is the synset #8':

#8 `fountainhead, headspring, head: the source of water from which a stream arise.`

#8' `fountainhead, headspring: the source of water from which a stream arise.`

Chapter 10

The Pattern Discovery Algorithm

In the Figure 10.1, we present the algorithm discoverStructuralPatterns that we use to compute the type compatible patterns. The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `PolysemyInstance`: A polysemy Instance as defined in definition 9.

3. `StructuralPattern`: Structural pattern as defined in definition 14.

4. `PatternLabel`: Pattern label as defined in definition 15.

5. `StructurlPatternClass`: Structural pattern class as defined in definition 16.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `polyInstances`: A list of the polysemy instances in wordNet.

3. `pattern`: A variable of type `StructuralPattern`.

4. `label` : A variable of type `PatternLabel`;

5. `patternClass`: A variable of type `StructuralPatternClass`.

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstances: list of polysemyInstance;
StructuralPattern: struct of {r: Synset, p1: Synset, p2: Synset};
pattern: StructuralPattern;
PatternLabel: string;
label : PatternLabel;
StructurlPatternClass: struct of {label: PatternLabel,
  instances:{PolysemyInstance}};
patternClass: StructuralPatternClass;
 patternsClasses: HashMap of PatternLabel x StructurlPatternClass;
function discoverStructuralPatterns (){
polyInstances := getPolyInstances(wH);
foreach polyInstance in polyInstances do{
  pattern := getStructuralPattern(polyInstance);
  if(isTypeCompatiblePattern(pattern)) then{
  label := getPatternLabel(polyInstance, pattern);
   if(!patternsClasses.containskey(label)) then {
     patternClass := new StructurlPatternClass(label, nil);
     patternClass.instances.add(polyInstance);
     patternsClasses.add(patternClass);
} else{
     patternClass := patternsClasses.get(label);
     patternClass.instances.add(polyInstance);}
  }
  }
return patternsClasses;}
```

Figure 10.1: Pseudo-code of the pattern discovery top level algorithm

6. `patternsClasses`: A hash map to store the structural pattern classes.

The input/output of the function:

- The input: `wH`.
- The output: `patternsClasses`.

The function works as follows:

1. The function retrieves the polysemy instances in wordNet via the function `getPolyInstances` (line 120) which is described in Figure 10.2.
2. The function iterates over all retrieved polysemy instances (line 140 - 260) and perform the following:
   (a) It retrieves the structural pattern of each polysemy instance by calling the function `getStructuralPattern` (line 150) which is described in Figure 10.3.
   (b) It checks if the structural pattern of the polysemy instance is type compatible structural pattern according to definition 22 via the function `isTypeCompatiblePattern` (line 160) which is described in Figure 10.4.
   (c) If the structural pattern of the polysemy instance is type compatible, it adds it to its corresponding structural pattern class (line 170 - 260) as follows:
      i. The function retrieves the structural pattern label via the function `getPatternLabel` (line 170) which is described in Figure 10.10.
      ii. It checks, if the structural pattern is computed for the first time, i.e. a new pattern.
      iii. If the pattern is new:

A. it creates a structural pattern class for it (line 190);

B. it adds the polysemy instance to the structural pattern class (line 200);

C. stores the structural pattern in `patternsClasses` under the pattern label (line 210).

iv. If the pattern is not a new pattern:

A. it retrieves the structural pattern class of the pattern from `patternsClasses` (line 230);

B. adds the polysemy instance to the structural pattern (line 240);

3. The function returns the hash map `patternsClasses` that contains structural patterns of the type compatible polysemy instances (line 270).

10.1 Polysemy Instances Discovery Algorithm

In the following, we present the function `getPolysemyInstances` that returns a list of all polysemy instances in WordNet as defined in definition 9. The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Term`: A term as defined in definition 1.

3. `Synset`: A synset as defined in definition 2.

4. `PolysemyInstance`: A polysemy instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `polyTerms`: A list of polysemous terms as defined in definition 7.

```
WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
wH: WordNetHierarchy;
Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
polyTerms, terms: list of Term;
Synset: struct of {terms: {Term}, label: Term, gloss: string,
      relations: {Relation}, s-rank: integer};
polySynsets: list of Synset;
PolysemyInstance: struct of {terms: {Term}, s1: Synset, S2: Synset};
polyInstances: list of PolysemyInstance;
function getPolysemyInstances(WordNetHierarchy wH)
 polyTerms := getPolysemousTerms(wH);
foreach term in polyTerms do {
  polySynsets = getPolysemousSynsets(term, wH);
   foreach s1 in polySynsets do {
        foreach s2 != s1 in polySynsets do {
      terms := getComonTerms(s1.terms,s2.terms);
        PolysemyInstance polyInstance;
        polyInstance:=new PolysemyInstance(terms, s1, s2);
       if (!polyInstances.contains(polyInstance)) then{
           polyInstances.add(polyInstance);}
}
}
return polyInstances;
}
```

Figure 10.2: Pseudo-code for computing polysemy instances

3. `polySynsets`: A list of polysemous synsets as defined in definition 8.
4. `polyInstances`: A list of the polysemy instances in wordNet as defined in definition 9.

The input/output of the function:

- The input: `wH`.
- The output: `polyInstances`.

The function computes the polysemy instances in wordNet by constructing the polysemy instances of each polysemous term. The number of these instances is proportional to the number of synsets in which a term is participating. The number of polysemy instances of a term with n meanings is equal to $\sum_{i=1}^{i=n-1} i = \frac{n*(n-1)}{2}$ polysemy instances. Because of the many-to-many relationship between terms and synsets in WordNet, a polysemy instance may belong to more than one polysemous term. For example, the term `alteration` has the following three meanings.

#1 `change, alteration, modification: an event that occurs when something passes from one state or phase to another.`

#2 `alteration, modification, adjustment: the act of making something different.`

#3 `revision, alteration: the act of revising or altering.`

The function computes the following three polysemy instances for this term. $[\{alteration, modification\}, \#1, \#2]$, $[\{alteration\}, \#1, \#3]$, and $[\{alteration\}, \#2, \#3]$. The first polysemy instance is also a polysemy instance of the term `modification` and shall not be considered again by computing the polysemy instances for this term.

The function works as follows:

1. The function retrieves the polysemous terms in WordNet via the function `getPolysemousTerms` (line 100). The function `getPolysemousTerms` returns a list the polysemous noun terms in wordNet according to definition 7.

2. The function iterates over all the retrieved polysemous terms to compute the polysemy instances of each term (line 110 - 210).

   (a) It retrieves the polysemous synsets of the polysemous term. (line 120) via the function `getPolysemousSynsets`. The function returns a list of the polysemous synsets of a polysemous term according to definition 8.

   (b) It iterates over the retrieved polysemous synsets to constructs the polysemy instances for them (line $130-210$).

      i. For two polysemous synsets of a polysemous term, we construct a polysemy instance (line 140 - 170).

      ii. The function stores the constructed polysemy instance to the list `polyInstances` (line 180 -190) as follows:

         A. The function tests, if the polysemy instance already added to the list `polyInstances` since it is possible for a polysemy instance to belong to more than one term as explained above.

         B. new polysemy instances are added to the list `polyInstances`.

3. The function returns `polyInstances`, the output of the function (line 220).

10.2 Structural Patterns Discovery Algorithm

In the following, we present the function `getStructuralPattern` that computes the structural pattern of a polysemy instance. The function uses the fol-

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
  {Relation}, s-rank: integer};
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
StructuralPattern: struct of {r: Synset, p1: Synset, p2: Synset};
polyInstanceStructuralPattern: StructuralPattern;
function getStructuralPattern (PolysemyInstance polyInstance){
Synset s1 := polyIsntance.s1;
Synset s2 := polyInstance.s2;
Synset r := getLeastCommonSubsumer(s1,s2);
Synset p1 := getStructuralPatternHyponym(s1,r);
Synset p2 := getStructuralPatternHyponym (s2,r);
polyInstanceStructuralPattern := new StructuralPattern(r,p1,p2);
return polyInstanceStructuralPattern;}
```

Figure 10.3: Pseudo-code for computing structural pattern

lowing data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Synset`: A synset data structure as defined in definition 2.
3. `PolysemyInstance`: A polysemy Instance as defined in definition 9.
4. `StructuralPattern`: Structural pattern as defined in definition 14.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `polyInstance`: A variable of type `PolysemyInstance`.
3. `s1,s2,r,p1,p2`: Variables of type `Synset`.
4. `polyInstanceStructuralPattern`: A variable of type `StructuralPattern`.

The input/output of the function:

- The input: `polyInstance`.
- The output: `polyInstanceStructuralPattern`.

The function works as follows:

1. The function retrieves the least common subsumer of the polysemy instance synsets s1 and s2 by calling the function `getLeastCommonSubsumer` (line100) which described in Figure 10.5.
2. Then, it retrieves the structural pattern hyponyms p1 and p2 by calling the function `getStructuralPatternHyponym` (line 110, line 120) which is described in Figure 10.9.
3. Then, it constructs the structural pattern of the input polysemy instance (line 130).
4. Finally, the constructed structural pattern is returned (line 140).

The functions `getLeastCommonSubsumer` and `getStructuralPatternHyponym` are described in Figure 10.5 and Figure 10.9. In the following, we describe the function `isTypeCompatiblePattern`. The function uses the following data struc-

```
StructuralPattern: struct of {r: Synset, p1: Synset, p2: Synset};
TypeIncompatibleRoot: Synset;
typeIncompatibleRoots: list of TypeIncompatibleRoot;
function isTypeCompatiblePattern(StructuralPattern pattern){
foreach s in typeIncompatibleRoots do{
 if (pattern.r = s) then {
    return true;}
 }
return false;
}
```

Figure 10.4: Pseudo-code for testing if a structural pattern is type compatible

tures:

1. `StructuralPattern`: Structural pattern as defined in definition 14.

2. `TypeIncompatibleRoot`: Type incompatible root as defined in definition 22.

The function uses the following variables:

1. `pattern`: A variable of type `StructuralPattern`.

2. `typeIncompatibleRoots`: A list of type incompatible roots.

The input/output of the function:

- The input: `pattern`.

- The output: `true` or `false`.

The function works as follows: The function checks if a structural pattern is a type compatible pattern type compatible (as defined in definition 22) by testing if the pattern root r belongs to the list of type incompatible roots according to definition 21.
In the following, we describe the function `getLeastCommonSubsumer` that computes the least common subsumer of two synsets as defined in definition 13. The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Synset`: A synset data structure as defined in definition 2.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `commonSubsumers`: A list of `Synset`.

3. `leasCommonSubsumer`: A variable of type `Synset`.

The input/output of the function:

```
10 WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
40 commonSubsumers: list of Synset;
50 leasCommonSubsumer: Synset;
60 function getLeastCommonSubsumer(Synset s1, Synset s2){
70  commonSubsumers := getCommonSubsumers(s1,s2);
80  sortSynsetsBySynsetHeight(commonSubsumers);
90 leasCommonSubsumer := commonSubsumers.get(0);
100 return leasCommonSubsumer;}
```

Figure 10.5: Pseudo-code for computing least common subsumer

- The input: `s1, s2`.
- The output: `leasCommonSubsumer`.

WordNet hierarchy is a connected, and single rooted digraph. Connected means that any two synsets in the hierarchy have at least one common subsumer. Single rooted means that the hierarchy has a single root that subsumes all other synsets. Digraph means that the hierarchy is not a tree and it is possible for two synsets to have more than one common subsumer. Accordingly, any two synsets in the hierarchy have a least one common subsumer. This common subsumer may be a common parent of the two synsets in best case, or the root of the hierarchy in worst case. The function works as follows:

1. The function retrieves the common subsumers of the synsets s1 and s2 by calling the function `getcommonSubsumers` (lines 70) which is described in Figure 10.6.

2. Then, it sorts the common subsumers according to 10 using the function `sortSynsetsBySynsetHeight` (line 80) which is described in Figure 10.8.

3. The least common subsumser is the synset in the first position in the list `commonSubsumers` (line 90).

4. The function returns `leasCommonSubsumer`, the output of the function (line 100).

In the following, we describe the function `getCommonSubsumers` that computes the common subsumers of two synsets according to definition 11. The

```
10 WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
40 synsetHypernyms1: list of Synset;
50 synsetHypernyms2: list of Synset;
60 commonSubsumers: list of Synset;
70 function getCommonSubsumers(Synset s1, Synset s2){
80  getSynsetHypernyms(s1,synsetHypernyms1);
90  getSynsetHypernyms(s2,synsetHypernyms2);
100 commonSubsumers := getIntersection(synsetHypernyms1, synsetHypernyms2);
110 return commonSubsumers;}
```

Figure 10.6: Pseudo-code for computing common subsumers

function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Synset`: A synset data structure as defined in definition 2.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `synsetHypernyms2, synsetHypernyms2`: A list of `Synset`.

3. `commonSubsumers`: A list of `Synset`.

The input/output of the function:

- The input: `s1, s2`.
- The output: `commonSubsumers`.

The function works as follows:

1. The function retrieves the synset hypernyms of the synsets s1 and s2 by calling the function `getSynsetHypernyms` (line 80 and line 90) which is described in Figure 10.9.
2. The common subsumers in the list are those synsets that belong to the hypernyms of s1 and the hypernyms of s2 are stored in `commonSubsumers`. (line 100)
3. The functions returns `commonSubsumers`, the output of the function.

In the following, we describe the function `getSynsetHypernyms`. The function

```
10 WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
40 synsetDirectHypernyms: list of Synset;
50 function getSynsetHypernyms(Synset s, synsetHypernyms: list of Synset){
60 synsetDirectHypernyms := getSynsetDirectHypernyms(s);
60 foreach dierectHypernym in synsetDirectHypernyms do{
70 synsetHypernyms.add(dierectHypernym);
80 getSynsetHypernyms(dierectHypernym, synsetHypernyms);
90 }
100}
```

Figure 10.7: Pseudo-code for computing synset hyponyms

uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Synset`: A synset data structure as defined in definition 2.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `s`: A variable of type `Synset`.
3. `dierectHypernym`: A variable of type `Synset`.
4. `synsetDirectHypernyms`: A list of `Synset`.

The input/output of the function:

- The input: `s, synsetHypernyms`.
- The output: no output, the function stores the results in `commonSynsetHypernyms`.

The function is a recursive function that computes the synset hypernyms according to definition 4. It works as follows:

1. The function retrieves the direct hypernyms of the input synset s by calling the function `getSynsetDirectHypernyms` (line100) that computes the synset direct hypernyms according to definition 3.
2. Recursively, the function computes the direct hypernyms of the retrieved direct hypernym of the synsets.
3. The function stops computing when it reaches the synset `entity`, the single root of wordNet hierarchy.

In the following, we describe the function `sortSynsetsSynsetHeight`. The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Synset`: A synset data structure as defined in definition 2.

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
  {Relation}, s-rank: integer};
function sortSynsetsBySynsetHeight (synsets: list of Synset){;
foreach s1 in synsets do{
foreach s2 != s1 in synsets do{
if(getSynsetHeight(s1) > getSynsetHeight(s2)) then{
swap(s1, s2);
}
  }
}
}
```

Figure 10.8: Pseudo-code for Sorting synsets

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `synsets`: A list of `Synset`.

The input/output of the function:

- The input: `synsets`.

- The output: no output, a call by reference sorting function.

The function sorts the synsets in a list of synsets using the function `getSynsetHeight` that computes the synset height according to definition 12.
In the following, we describe the function `getStructuralPatternHyponym`.
The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Synset`: A synset data structure as defined in definition 2.

The function uses the following variables:

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
 {Relation}, s-rank: integer};
structuralPatternHyponym: Synset;
function getStructuralPatternHyponym(Synset s, Synset r){
Synset structuralPatternHyponym := s;
while(!isDirectHypernym(r, structuralPatternHyponym) do{
structuralPatternHyponym := getDirectHypernym(structuralPatternHyponym);
}
return structuralPatternHyponym;}
```

Figure 10.9: Pseudo-code for constructing unique pattern label

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `structuralPatternHyponym, s, r`: Variables of type `Synset`.

The input/output of the function:

- The input: `s, r`.

- The output: `structuralPatternHyponym`.

The function computes the pattern hyponym p1 or p2 in respect to the pattern root as defined in definition 14 and works as follows:

1. The input variable `s` is assigned to `structuralPatternHyponym`.

2. As long as the input variable `r` is not the direct hypernym of `structuralPatternHyponym`, it is assigned to its direct hypernym. The function `isDirectHypernym` is used to test if `r` is a direct hypernym of `structuralPatternHyponym`, the function `getDirectHypernym` is used to retrieve the direct hypernym of `structuralPatternHyponym` according to definition 3.

3. The loop stops when `r` is the direct hypernym of `structuralPatternHyponym`. Notice that `r` is a hypernym of `s`. That means `structuralPatternHyponym`

is equal to `s` itself in case of common parent structural pattern or a hypernym of `s` otherwise.

4. The functions returns `structuralPatternHyponym`, the output of the function.

In the following, we describe the function `getPatternLabel`. The function

```
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations: {Relation}, s-rank: integer};
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstance: PolysemyInstance;
StructuralPattern: struct of {r: Synset, p1: Synset, p2: Synset};
PatternLabel: string;
label: PatternLabel;
function getPatternLabel(PolysemyInstance polyInstance, StructuralPattern pattern){
if(isCommonParentStructuralPattern(polyInstance, pattern)) then{
label := "common parent";
    }else{
  Synset r,p1, p2;
    r  := pattern.r;
    p1 := pattern.p1;
    p2 := pattern.p2;
 String rootLabel := r.label;
 String labelPart1, labelPart2;
 if(p1.label < p2.label) then {
 labelPart1 := p1.label;
 labelPart2 := p2.label;}
 else {
 labelPart1 := p2.label;
 labelPart2 := p1.label;}
label := rootLabel ."#<".labelPart1.",".labelPart2.">";
return label; }
```

Figure 10.10: Pseudo-code for constructing pattern label

uses the following data structures:

1. `Synset`: A synset data structure as defined in definition 2.
2. `PolysemyInstance`: A polysemy Instance as defined in definition 9.
3. `StructuralPattern`: Structural pattern as defined in definition 14.
4. `PatternLabel`: Pattern label as defined in definition 15.

The function uses the following variables:

1. `polyInstance`: A variable of type `PolysemyInstance`.
2. `pattern`: A variable of type `StructuralPattern`.
3. `p1,p2`: A variables of type `Synset`.
4. `label`: A variable of type `PatternLabel`.

The input/output of the function:

- The input: `polyInstance, pattern`.
- The output: `label`.

The function works as follows:

1. The function checks if the structural pattern belongs to common parent structural patterns as defined in definition 17 using the function `isCommonParentStructuralPattern` (line 80) which is described in Figure 10.11.
2. If the structural pattern belongs to common structural patterns, the function returns the label "common parent" (line 90).
3. Otherwise, the function constructs the pattern label as defined in definition 15 as follows (line 100-220):
   (a) The function retrieves the structural pattern root `r` and the structural pattern parts `p1, p2` (line 100-140).

(b) Then, it constructs the pattern label (line 150 - 220) by concatenating the labels of `r`, `p1`, and `p2` based on the lexicographic order of `p1` and `p2` . Using the lexicographic order enables us to consider the patterns $\langle r, p_1, p_2 \rangle$ and $\langle r, p_2, p_1 \rangle$ as one and the same structural pattern.

4. The function returns the constructed pattern label (line 240).

In the following, we describe the function `isCommonParentStructuralPattern`. The function tests if a structural pattern is a common parent structural

```
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstance: PolysemyInstance;
StructuralPattern: struct of {r: Synset, p1: Synset, p2: Synset};
function isCommonParentStructuralPattern( StructuralPattern pattern,
   PolysemyInstance polyInstance){
return pattern.p1 = polyInstance.s1 || pattern.p2 = polyInstance.s2;
}
```

Figure 10.11: Pseudo-code for testing common parent structural patterns

pattern as defined in definition 17. The function uses the following data structures:

1. `Synset`: A synset data structure as defined in definition 2.
2. `PolysemyInstance`: A polysemy Instance as defined in definition 9.
3. `StructuralPattern`: Structural pattern as defined in definition 14.

The function uses the following variables:

1. `PolyInstance`: A variable of type `PolysemyInstance`.
2. `pattern`: A variable of type `StructuralPattern`.

3. `s1,s2,p1,p2`: Variables of type `Synset`.

The input/output of the function:

- The input: `polyInstance, pattern`.
- The output: `true` of `false`.

Chapter 11

Pattern Classification and False Positives Identification

In this chapter, we describe The steps S1.P2.1.2 and S1.P2.1.2 of our approach. The input of this step is the output of the pattern discovery algorithm in S1.P2.1.1 which is a hash map that contains the structural pattern labels as keys and the structural pattern classes as values. We divide the structural pattern classes into regular and non regular structural pattern classes according to definition 19, where the common parent and the structural pattern classes that contain two instances at least are considered to be regular.

11.1 Pattern Classification

Our task in this step is to classify the regular structural pattern classes into specialization polysemy, metaphoric, and homonymy structural patterns. Non regular structural patterns are handled in next section. Metonymy polysemy cases and metonymic structural patterns belong to type incompatible patterns and they were excluded in the pattern discovery algorithm. For example, Figure 11.1 illustrates a polysemy instance of the term `news paper` where one synset refers to `news paper` as `an artifact` and the second synset refers to `news paper` as an `organization`. The structural pattern of this polysemy instance is $\langle entity, physical\ entity, abstract\ entity\rangle$ and hence it belongs to type incompatible structural patterns because its root belongs to type incompatible roots as defined in definition 21.

Notice that the excluded polysemy instances include also homonymy and metaphoric polysemy instances.

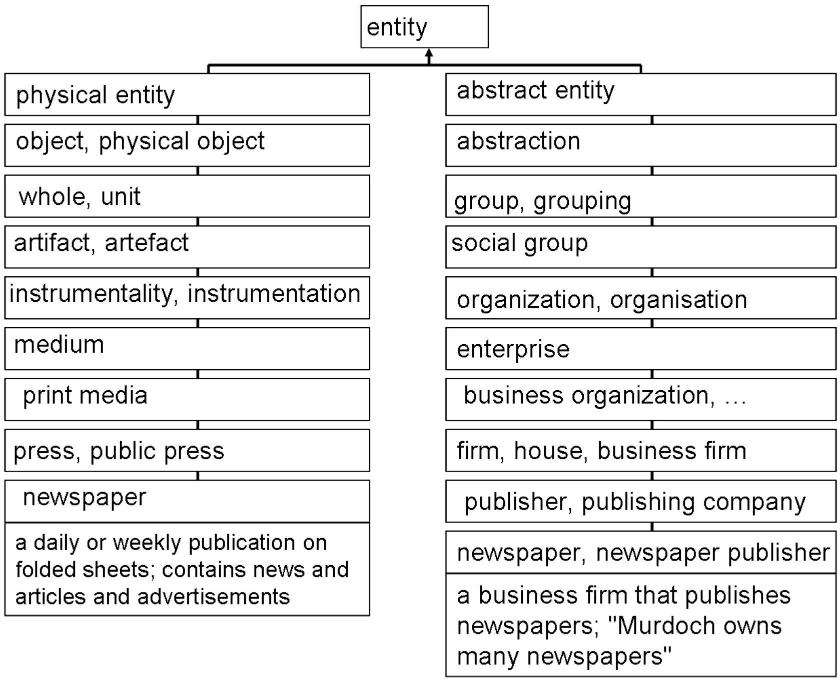

Figure 11.1: Example of a metonymy polysemy instance

In the following, we describe the pattern classification of the other polysemy types. This step is fully manual and based on the (implicit) semantics of the genus and differentia in the synset glosses.

11.1.1 Metaphoric Patterns

Metaphoric polysemy instances may belong to type incompatible or type compatible structural patterns. For example, the metaphoric polysemy instance of the term `ocean` in Figure 11.2 belongs to the structural pattern $\langle entity, physical\ entity, abstract\ entity\rangle$, while the metaphoric polysemy instance of the term `gold digger` in Figure 11.3 belongs to the pattern $\langle person\ , worker, female\rangle$.

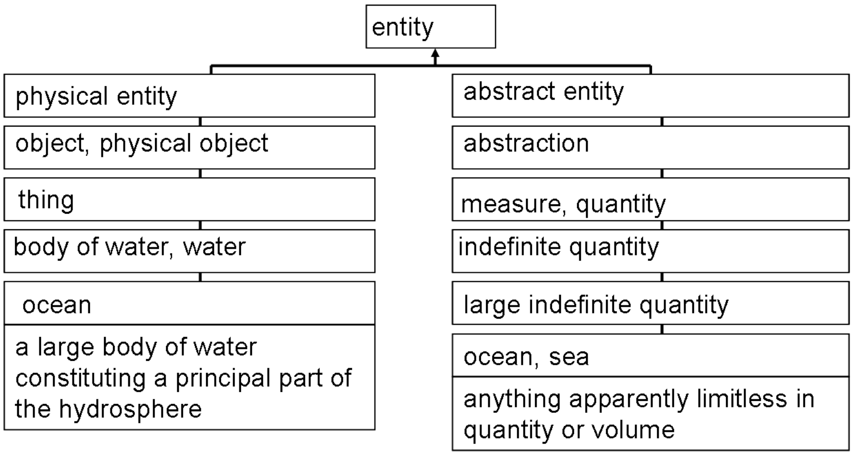

Figure 11.2: Example for type incompatible metaphoric polysemy instances

In this section, we are concerned with identifying metaphoric patterns that were identified by the algorithm as type compatible patterns. The exclusiveness property that we have used to identify type incompatible and type compatible patterns does not help to identify metaphoric structural patterns. Identifying metaphoric patterns is based on the distinction be-

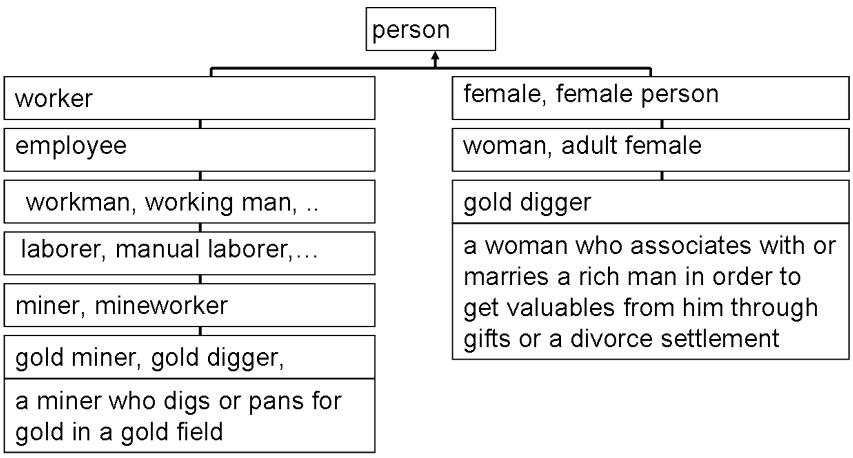

Figure 11.3: Example for type compatible metaphoric polysemy instances

tween the literal meaning and the figurative meaning. Our idea is that it is not possible for a literal and the figurative meaning to be collectively exhaustive [55]. In the following, we define the collectively exhaustive property.

Definition 27 *(Collectively Exhaustiveness property).*

Two synsets $s_1, s_2 \in S$ are collectively exhaustive if there exists or it is possible to find a synset s such that $s^I = s_1^I \sqcup s_2^I$ and s_1, s_2 fulfill the exclusiveness property.

For example, `abstract entity` and `physical entity` fulfill the collectively exhaustiveness property because $entity^I = abstract \quad entity^I \sqcup physical \quad entity^I$. On the other hand `worker` and `female` do not fulfill this property because `worker` corresponds to a role and `female` to a concept. This is because `person` is a direct hypernym of the concept `organism` and the role `causal agent`. Notice that `female worker` and `male worker` (WordNet does not contain these two synsets) can be considered collectively exhaustive because `female worker` and `male worker` are roles and can be hyponyms of the role `worker` and `female worker` and `male worker` fulfill the collectively exhaustiveness property ($worker^I = female \quad worker^I \sqcup male \quad worker^I$).

In the cases, where both pattern hyponyms are concepts, metaphoric patterns do not fulfill the collectively exhaustiveness property as follows. The metaphoric relation between the literal meaning (the source meaning) and the (figurative meaning) is analogy relation. This means, one of the pattern hyponyms in a metaphoric pattern is a subset of the other hyponym as in illustrated in Figure 11.4. That means the pattern hyponyms do not fulfill the exhaustiveness property because either $p_1 \sqsubset p_2$ or $p_2 \sqsubset p_1$.

In the following, we define metaphoric patterns as follows.

Definition 28 *(Metaphoric structural pattern).*

A pattern $p = \langle r, p_1, p_2 \rangle$ is metaphoric if p_1 and p_2 do not fulfill the collectively exhaustiveness property.

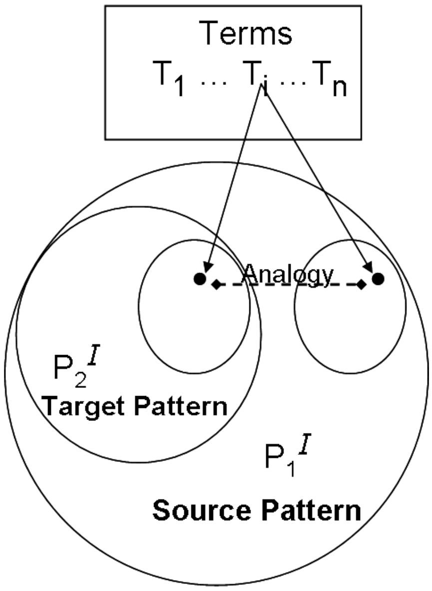

Figure 11.4: Metaphoric pattern schema

Notice that the schema in Figure 11.4 is for type compatible metaphors. For type incompatible metaphors, the relation holds between the attributes of both concepts, not the concepts themselves. For example, the analogy between `indefinite quantity` and `body of water` in Figure 11.2 is between the attribute `size` of both concepts and the analogy may be that both concepts are limitless is size. However, we are concerned in this thesis with type compatible metaphors.

In the following we give examples for identified metaphoric patterns. The pattern $\langle organism, animal, person \rangle$ is metaphoric. Although both synsets share the same hypernym `organism`, they are not collectively exhaustive because $person^I \sqsubset animal^I$.

```
animal, animate being, beast, brute, creature, fauna -- (a living organism
   characterized by voluntary movement)
person, individual, someone, somebody, mortal, soul -- (a human being; "there
   was too much for one person to do")
     => organism, being -- (a living thing that has (or can develop) the
        ability to act or function independently)
```

The polysemy instances that belong to this pattern are 326 instances. Consider for example the following instance.

#1 `fox: alert carnivorous mammal with pointed muzzle and ears and a bushy tail; most are predators that do not hunt in packs.`

#2 `dodger, fox, slyboots: a shifty deceptive person.`

Another example is the pattern $\langle attribute, property, trait \rangle$. Although, both synset share the same hypernym `attribute`, they are not collectively exhaustive because $trait^I \sqsubset property^I$ ($trait^I = property^I \sqcap person^I$).

```
property -- (a basic or essential attribute shared by all members of a class;
   "a study of the physical properties of atomic particles")
trait -- (a distinguishing feature of your personal nature)
      => attribute -- (an abstraction belonging to or characteristic of an
         entity)
```

The polysemy instances that belong to this pattern are 111 instances. Consider for example the following instance.

#1 `softness:the property of giving little resistance to pressure and being easily cut or molded.`

#2 `gentleness, softness, mildness: acting in a manner that is gentle and mild and even-tempered; "his fingers have learned gentleness"; "suddenly her gigantic power melted into softness for the baby"; "even in the pulpit.`

11.1.2 Specialization Polysemy Patterns

Based on the hierarchical relation between specialization polysemy instances, we identify specialization patterns as schematized in Figure 11.5 We define specialization polysemy patterns as follows.

Definition 29 *(specialization polysemy structural pattern).*

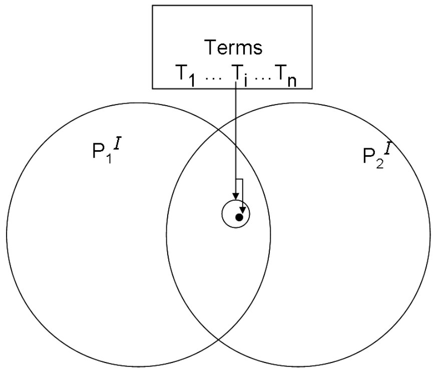

Figure 11.5: Specialization Polysemy Pattern Schema

A pattern $p = \langle r, p_1, p_2 \rangle$ is a specialization polysemy pattern if a) and b) hold

a) p_1 and p_2 do not fulfill the exclusiveness property.

b) p is not a metaphoric pattern.

In the following we give examples for identified specialization polysemy patterns. All instances that belong to the common parent structural patterns are classified as specialization polysemy instances. The polysemy instances that belong to this pattern are 2879 instances. Consider for example the following instance.

#1 `capital, working capital: assets available for use in the production of further assets.`

#2 `capital: wealth in the form of money or property owned by a person or business and human resources of economic value.`

Another example is the pattern $\langle act, action, activity \rangle$. The polysemy instances that belong to this pattern are 406 instances. Consider for example the following instance.

```
employment, work -- (the occupation for which you are paid; "he is looking for
   employment"; "a lot of people are out of work")
```

```
      => occupation, business, job, line of work, line -- (the principal
         activity in your life that you do to earn money; "he's not in my
         line of business")
         => activity -- (any specific behavior; "they avoided all
            recreational activity")
employment, engagement -- (the act of giving someone a job)
      => action -- (something done (usually as opposed to something said);
         "there were stories of murders and other unnatural actions")
            => act, human action, human activity -- (something that people
               do or cause to happen)
```

Anther example is the pattern $\langle artifact, instrumentality, structure\rangle$. The polysemy instances that belong to this pattern are 127 instances. Consider for example the following instance

#6 `foot: a support resembling a pedal extremity; "one foot of the chair was on the carpet".`

#7 `foundation, base, fundament, foot, groundwork, substructure, understructure: lowest support of a structure; "it was built on a base of solid rock"; "he stood at the foot of the tower".`

Another example, is the pattern $\langle animal, invertebrate, larva\rangle$. The polysemy instances that belong to this pattern are 17 instances. Consider for example the following instance. Notice that the pattern hyponyms correspond to the phases of a a larva.

#1 `ailanthus silkworm, Samia cynthia: large green silkworm of the cynthia moth.`

#2 `cynthia moth, Samia cynthia, Samia walkeri: large Asiatic moth introduced into the United States; larvae feed on the ailanthus.`

11.1.3 Homonymy Patterns

Based on the exclusiveness property and metaphoric definition, we identify homonymy patterns as schematized in Figure 11.6

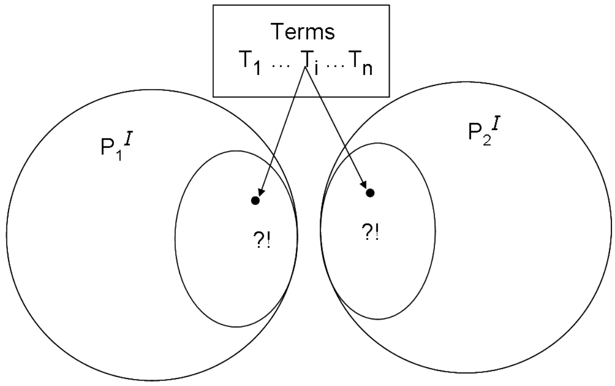

Figure 11.6: Homonymy Pattern Schema

The pattern discovery algorithm in S1.P2.1.1 is based on type incompatible root as defined in definition 21 to exclude all polysemy instances that are for sure not specialization polysemy instances. In the previous sections , we have seen that the results of the algorithm contains metaphoric patterns. In that section, we have explained that the exclusiveness property is not sufficient to identify metaphoric patterns because metaphoric patterns may belong to type incompatible or type compatible patterns. In this section we define homonymy patterns based on exclusiveness property and the definition of metaphoric patterns as follows.

Definition 30 *(Homonymy structural pattern).*

A pattern $p = \langle r, p_1, p_2 \rangle$ is homonymy pattern if a) and b) hold.

a) p_1 and p_2 fulfill the exclusiveness property.

b) p is not a metaphoric pattern.

In the following we give examples for identified homonymy patterns. The pattern $\langle organism, person, plant \rangle$. The polysemy instances that belong to this pattern are 40 instances. Consider for example the following instance.

#1 `spinster, old maid: an elderly unmarried woman.`

#2 `zinnia, old maid, old maid flower: any of various plants of the genus Zinnia cultivated for their variously and brightly colored flower heads.`

Another example is the pattern $\langle organism, animal, plant\rangle$. The polysemy instances that belong to this pattern are 41 instances. Consider for example the following instance.

#1 `red fox, Celosia argentea: weedy annual with spikes of silver-white flowers.`

#2 `red fox, Vulpes fulva: New World fox; often considered the same species as the Old World fox.`

Another example is the pattern $\langle vertebrate, bird, mammal\rangle$. The polysemy instances that belong to this pattern are 13 instances. Consider for example the following instance.

#3 `griffon, wire-haired pointing griffon: breed of medium-sized long-headed dogs with downy undercoat and harsh wiry outer coat; originated in Holland but largely developed in France.`

#4 `griffon vulture, griffon, Gyps fulvus: large vulture of southern Europe and northern Africa having pale plumage with black wings.`

11.2 False Positives Identification

This step corresponds to S1.P2.1.2 in our approach. Our task here is to process the four lists returned at the end of the pattern classification and remove false positives. These lists correspond to metaphoric polysemy list, specialization polysemy list, homonymy list, and a list of non regular (singleton patterns) list. This task is performed also manually due to the implicit and missing information in synset glosses. Our procedure to determine the polysemy class of a polysemy instance is based on the three definitions in the previous section, where we process the polysemy instances

instance by instance to determine the the relation between the synsets of the polysemy instances.
If a polysemy instance does not belong to the polysemy class it was assigned to (false positive instance) in the previous step, we assign it to its corresponding polysemy class.
In the following, we give examples for false positives. The common parent structural pattern which was assigned to the specialization polysemy class contains 180 false positive polysemy instances, 98 of them were identified as homonymy instances such as the following instance.

#1 `cardholder: a person who holds a credit card or debit card.`

#2 `cardholder: a player who holds a card or cards in a card game.`

Metaphoric false positives (82 instances) were also identified in the common parent class such as the following instance.

#1 `game plan: (figurative) a carefully thought out strategy for achieving an objective in war or politics or business or personal affairs; "newscasters speculated about the President's game plan for an invasion".`

#2 `game plan: (sports) a plan for achieving an objective in some sport.`

Another example is the pattern $\langle organism, animal, person \rangle$ which was assigned to the metaphoric polysemy class contains 326 polysemy instances, 74 of them were identified as homonyms such as the following instance.

#2 `Minnesotan, Gopher: a native or resident of Minnesota.`

#3 `ground squirrel, gopher, spermophile: any of various terrestrial burrowing rodents of Old and New Worlds; often destroy crops.`

Chapter 12

Specialization Polysemy Organization

In this chapter, we discuss the steps in the phase S1.P2.2 of our approach.

12.1 Specialization Polysemy Sub-classes Discovery

In this section, we explain the step S1.P2.2.1 in our Approach. In chapter 4, we have classified the specialization polysemy problem into the following problems.

a) *The problem of implicit relatedness*

b) *The problem of too fine-grained senses*

c) *The problem of redundancy*

d) *The problem of sense enumeration*

In this section, discuss the specialization polysemy sub-class discovery algorithm that classify the specialization polysemy instances into three classes as follows.

- *Implicit relatedness sub-classes*: Based on the nature of implicit relatedness, we differentiate between two classes.

i) *Missing relation sub-class*

ii) *Missing parent sub-class*

- *Too fine grained senses, redundant, and sense enumerations sub-class* This sub class contains the polysemy instances that corresponds to the problems b) to d) that we call

iii) *Merge sub-class*

In the following, we discuss the three classes.

12.1.1 Implicit Relatedness Sub-classes Discovery

The implicit relatedness sub-class discovery is based on the more general/-more specific meaning relation which corresponds to a semantic relation that we define as follows.

Definition 31 ***(more general/more specific meaning.)***

For synsets s_1, s_2,

a) s_1 is said to be a more general meaning of s_2 if $s_2^I \sqsubset s_1^I$.

b) s_1 is a more specific meaning of s_2 if s_2 is a more general meaning of s_1.

In the following we define two classifications of specialization polysemy instances. The first one is based on the more general meaning definition and the other is based on the relation between the synset terms.

Definition 32 ***(semantic classification of specialization polysemy)***

Let $I = [Ts, s_1, s_2]$ be a specialization polysemy instance. The more general meaning between s_1 and s_2 can be one of the following cases.

a) s_1 is a more general meaning of s_2 and s_2 is not a more general meaning of s_1: Such cases correspond to the polysemy instances where $s_2^I \equiv (s_1.genus)^I$.

b) s_1 is not a more general meaning of s_2 and s_2 is not a more general meaning of s_1 but s_1 and s_2 are more specific meaning of another synset. Such cases correspond to the polysemy instances where $(s_1.genus)^I \equiv (s_2.genus)^I$ and $(s_1.differentia)^I \not\equiv (s_2.differentia)^I$

c) s_1 is a more general meaning of s_2 and s_2 is a more general meaning of s_1. Such cases correspond to the polysemy instances where $(s_1.genus)^I \equiv (s_2.genus)^I$ and $(s_1.differentia)^I \equiv (s_2.differentia)^I$

The groups a), b), and c) define the possible relation between the synsets in a specialization polysemy instance. In the following, we explain our criteria to identify the polysemy instances of each of the groups a) and b). We discuss the group c) in section 12.1.2.

Definition 33 ***(synonymity classification of specialization polysemy)***

Let $I = [Ts, s_1, s_2]$ be a specialization polysemy instance. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. The relation between the terms of the synsets can be one of the following cases.

a') $Ts_1 \subset Ts_2$ and $Ts_2 \not\subset Ts_1$.

b') $Ts_1 \not\subset Ts_2$ and $Ts_2 \not\subset Ts_1$ and $Ts_1 \cap Ts_2 \neq \emptyset$

c') $Ts_1 = Ts_2$

According to the polysemy definition in WordNet, a term is polysemous, if there is at least two contexts, where it is used to refer to a meaning in one context and it refers to another meaning in an the other context. On the other hand, a term is monosemous, if it refers to the same meaning in all contexts. Now, since the relation between the synsets of a specialization polysemy instances are hierarchical, the monosemous synonyms in such synsets can be used to indicate which synset is the more general meaning and/or which is the more specific one as follows.

Definition 34 ***(more general/more specific term)***

Let $I = [Ts, s_1, s_2]$ be a specialization polysemy instance. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let $T_{ms} = (Ts_1 \cup Ts_2) \backslash Ts$. A term $t \in s_1 \cup s_2$ is said to be

i) more general meaning term, if $t \in Ts$

ii) more specif term, if $t \in T_{ms}$

For example, `victor` is a more general term in the polysemy instance $[\{victor\}, \#1, \#2]$, while `master` and `winner` are more specific terms ($T_{ms} = \{master, winner\}$).

#1 `victor, master, superior: a combatant who is able to defeat rivals.`

#2 `winner, victor: the contestant who wins the contest.`

Based on the more general/more specific term definition, we discuss in the following two sub sections how we identify the instances of the group a) as those the instances in a') and the instances in b) as those instances in b'). The relation between the instances in c) and c') is explained in section 12.1.2.

Missing Relation sub-class

Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let Ts be the set of more general terms in I and T_{ms} the set of more specific terms. Let

I belong to the cases in definition 33 item a'). The fact that $Ts_1 \subset Ts_2$ and $Ts_2 \not\subset Ts_1$ implies $Ts_2 \cap T_{ms} = \emptyset$, $T_{ms} \subset Ts_2$, and $Ts_1 = Ts$. This means that all the terms in Ts_1 may refer to the more general synset in some context and at the same time it may refer to the more specific one in some other context. On the other hand, only a subset of Ts_2 can refer to the more specific synset only. That means s_1 is a more general meaning of s_2 and s_2 is not a more general meaning of s_1 which describes the cases in definition 32 item a).
Because the more general meaning between s_1 and s_2 is missing in WordNet, we call this specialization polysemy sub-class the missing relation sub-class that we define as follows.

Definition 35 *(missing relation synsets sub class).*

Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. I belongs to the missing relation synsets sub class if the following hold.

a) $Ts_1 \subset Ts_2$;

b) $Ts_2 \not\subset Ts_1$.

An example for missing relation polysemy instance is $[\{turtledove\}, \#1, \#2]$.

```
[#1] Australian turtledove, turtledove, Stictopelia cuneata -- (small
   Australian dove)
[#2] turtledove -- (any of several Old World wild doves)
      => dove -- (any of numerous small pigeons)
```

Missing Parent sub-class

Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let Ts be the set of more general terms in I and T_{ms} the set of more specific terms. Let I belong to the cases in definition 33 item b'). The fact that $Ts_1 \not\subset Ts_2$, $Ts_2 \not\subset Ts_1$, and $Ts_1 \cap Ts_2 \neq \emptyset$ implies $Ts_1 \cap T_{ms} \neq \emptyset$ and $Ts_2 \cap T_{ms} \neq \emptyset$.

This means that the terms in Ts may refer to the more general synset in some context and at the same time it may refer to the more specific one in some other context. On the other hand, a subset of Ts_1 may refer to s_1 but not to s_2 and a subset of Ts_2 may refer to s_2 but not to s_1. That means s_1 is not a more general meaning of s_2 and s_2 is not a more general meaning of s_1 but s_1 and s_2 are more specific meaning of another synset which describes the cases in definition 32 item b).
Because the more general meaning of s_1 and s_2 is missing in WordNet, we call this specialization polysemy sub-class the missing parent sub-class that we define as follows.

Definition 36 ***(missing parent synsets sub class)***

. Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let Ts be the set of more general terms in I and T_{ms} the set of more specific terms. I belongs to the missing relation synsets sub class if the following hold.

a) $Ts_1 \not\subset Ts_2$;

b) $Ts_2 \not\subset Ts_1$;

c) $Ts_1 \cap Ts_2 \neq \emptyset$.

An example for missing relation polysemy instance is $[\{kestrel\}, \#1, \#2]$.

```
[#1] sparrow hawk, American kestrel, kestrel, Falco sparverius -- (small North
   American falcon)
[#2] kestrel, Falco tinnunculus -- (small Old World falcon that hovers in the
   air against a wind)
      => falcon -- (diurnal birds of prey having long pointed powerful wings
         adapted for swift flight)
```

12.1.2 Too Fine Grained, Redundant, and Sense Enumeration Sub-class Discovery

In chapter 4, we have described the problems of fine grained senses, redundancy and sense enumerations in specialization polysemy. In that chapter, we have also described the sense enumeration problem as a problem in compound noun polysemy. The difference between sense enumeration in compound polysemy and specialization polysemy is as follows. In compound noun polysemy, the sense enumeration appears when the noun modifier or the modified noun is synonymous to its corresponding compound noun. That means the synset contains in addition to the polysemous modified noun or noun modifier at least another one synonym which is the compound noun itself. The sense enumeration in specialization polysemy appears when we use the same terms to refer to two different synsets such that one synset refers to a general concept and the other refers to a special case. For example, the synset #1 of the term `timetable` is a general meaning while the synset #2 is a special case of `timetable` that is considered to be denoted by the term.

#1 `timetable: a schedule listing events and the times at which they will take place.`

#2 `timetable: a schedule of times of arrivals and departures.`

A more appropriate term to denote #2 may be `departures/arrivals time table`. We think that the problem of sense enumeration in WordNet can be explained by the problem of missing terms that we have discussed in chapter 2.

Specialization polysemy instances contain also too fine grained senses and redundancy such as the following examples.

#1 `hope:a specific instance of feeling hopeful; "it revived their hope of winning the pennant".`

#2 `hope: the general feeling that some desire will be fulfilled; "in spite of his troubles he never gave up hope".`

Understanding the difference between the two meanings of `hope` is very difficult. Specialization polysemy instances contain also redundancy.

#1 `comedienne: a female actor in a comedy.`

#2 `comedienne: a female comedian.`

The sense enumeration, too fine grained senses and redundancy in specialization polysemy have in common that the synsets in the polysemy instances are denoted by the same terms. For this reason we consider them to belong to the same specialization polysemy sub class as follows.
Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let Ts be the set of more general terms in I and T_{ms} the set of more specific terms. Let I belong to the cases in definition 33 item c'). The fact that $Ts_1 = Ts_2$ implies $Ts_1 \cap T_{ms} = \emptyset$ and $Ts_2 \cap T_{ms} = \emptyset$. This means that the terms in Ts may refer to s_1 and s_2 in all contexts. On the other hand,$T_{ms} = \emptyset$. That means s_1 is a more general meaning of s_2 and s_2 is a more general meaning of s_1 which describes the cases in definition 32 item c).
Because all these cases indicate adding a non appropriate synset , we call this specialization polysemy sub-class the redundancy sub-class that we define as follows.

Definition 37 ***(redundant synsets sub class)***

Let $I = [Ts, s_1, s_2]$. Let $Ts_1 = s_1.Terms$, $Ts_2 = s_2.Terms$. Let Ts be the set of more general terms in I and T_{ms} the set of more specific terms. I belongs to the missing relation synsets sub class if $Ts_1 = Ts_2$;
The three examples, previously discussed are examples for polysemy instances of the redundant synsets sub class.

12.2 Applying Specialization Polysemy Operations

In the following, we describe the automatic operations applied on the identified polysemy instances according to the specialization polysemy sub class.

12.2.1 Organization of the Polysemy Instances in the Missing Relation Sub-class

For missing relation cases, we apply the operation as shown in Figure 12.1. The function `organizeMissingRelationInstance` works as follows. Based on the

```
10 WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Term: struct of {lemma: string; cat: grammatical category; t-rank: integer};
40 pTerms: list of Term;
50 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
    {Relation}, s-rank: integer};
60 s1,s2: Synset;
70 PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
80 function organizeMissingRelationInstance(PolysemyInstance polyInstance)
90 {Term List pTerms := polyInstance.terms;
100  s1 := polyInstance.s1; s2 := polyInstance.s2;
110 if(|s1.terms| < |s2.terms|) then{
120   s2.terms := s2.terms\pTerms;
130   s2.relations := s2.relations U {<s2, hypernym, s1>};
140    return s2;}
150 else{
160   s1.terms := s1.terms\terms;
170   s1.relations := s1.relations U {<s1, hypernym, s2>};
180    return s1;}
```

Figure 12.1: Pseudo code for adding a missing relation

terms of the synsets in the input `polysemyInstance`:

1 The function determines the more general meaning synset (line 110).

2 If s1 is the more general meaning synset, then:

a) disambiguate the more specific synset s2 (line 120).

b) add the missing hypernymy relation between the more specific synset s2 and the more general synset s1 (line 130).

3 If s2 is the more general meaning synset, then:

a) disambiguate the more specific synset s1 (line 160).

b) add the missing hypernymy relation between the more specific synset s1 and the more general synset s2 (line 170).

Applying the missing relation operation on the polysemy instance in Figure 12.2 is shown in Figure 12.3.

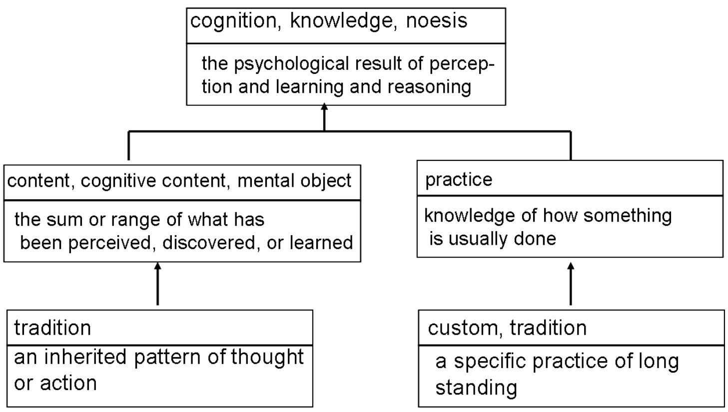

Figure 12.2: An example for a missing relation specialization polysemy instance

12.2.2 Organization of the Polysemy Instances in the Missing Parent Sub-class

For missing parent cases, we add a new (missing) parent as shown in Figure 12.4. The function `organizeMissingParentInstance` organizes the polysemy instances in the missing parent group in the following way.

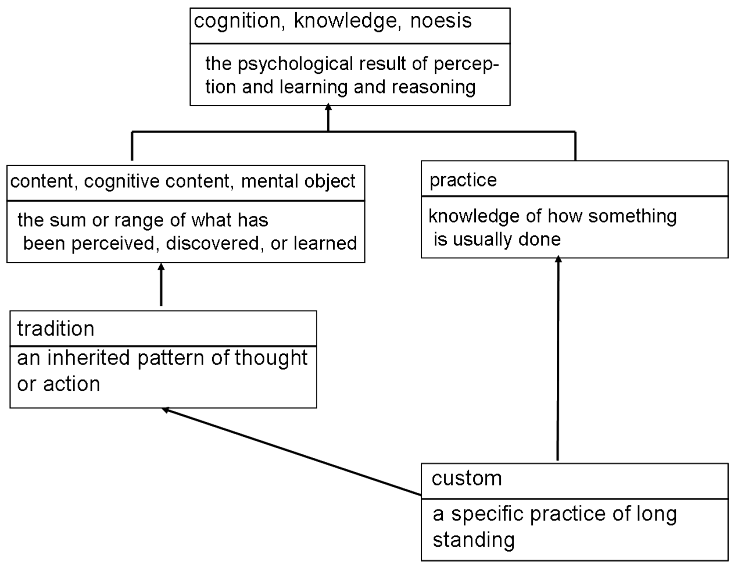

Figure 12.3: An example for adding a missing relation

1 The function create the new parent `mParent` for the synsets of the input polysemy Instance.

a) The function retrieves the least common subsumer of s1 and s2 (line 120).

b) The function retrieves the preferred term of `mParent` via the function `getPrefferedTerm` which is the preferred term in the shared polysemous terms in the input polysemy instance (line 130).

c) The function retrieves the preferred term of the least common subsumer via the function `getPrefferedTerm` (line 140).

d) The terms of `mParent` are the shared polysemous terms in the input polysemy instance (line 150).

e) The parent of `mParent` is the least common subsumer of the synsets in the input polysemy instance (line 160).

f) The gloss of `mParent` is constructed automatically with the following form: `pTerm is clsTerm`, where clsTerm is the preferred term of the least common subsumer of s1 and s2 (line 170).

```
10 WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Term: struct of {lemma: string; cat: grammatical category; t-rank: integer};
40 pTerm,clsTerm: Term
50 pTerms: list of Term;
60 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
70 s1,s2,mParent,commonLeastSubsumer: Synset;
80 PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
90 function organizeMissingParentInstance(PolysemyInstnace polyInstance){
100  s1 := polyInstance.s1; s2:=polyInstance.s2;
110 pTerms polyInstance.terms;
120 commonLeastSubsumer := getLeastCommonSubsumer(s1,s2);
130 pTerm  := getPrefferedTerm(pTerms);
140 clsTerm := getPrefferedTerm(commonLeastSubsumer);
150 mParent.terms := pTerms;
160 mParent.relations := {<mParent,hypernymy, commonLeastSubsumer>};
170 mParent.gloss := pTerm. " is a ".term1;
180 s1.relations := s1.relations U {<s1,hypernymy, mParent>};
190 s2.relations := s2.relations U {<s2,hypernymy, mParent>};;
200 s1.terms := s1.terms\pTerms;
210 s2.terms := s2.terms\pTerms;
220 return mParent;}
```

Figure 12.4: Pseudo code for adding a missing parent

2 The synsets s1 and s2 are connected to mParent via the hypernym relation (line 180 and 190).

3 The synsets s1 and s2 are disambiguated by removing the polysemous terms from both synsets (line 200 and 210).

Applying the missing parent operation on the polysemy instance in Figure 12.5 is shown in Figure 12.6.

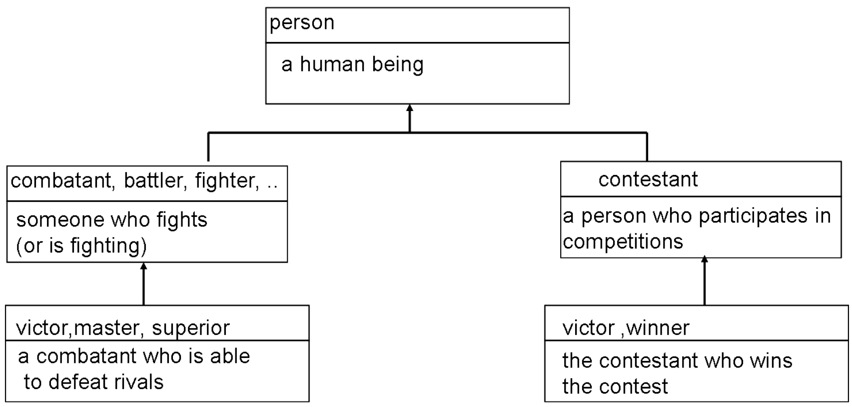

Figure 12.5: An example for a missing parent specialization polysemy instance

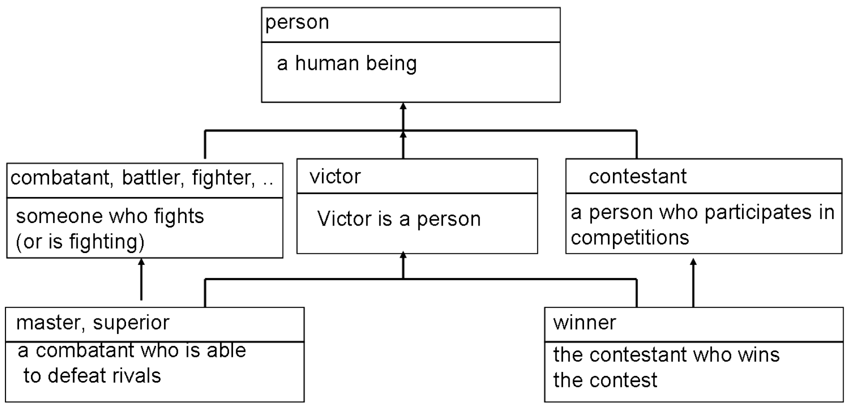

Figure 12.6: An example for adding a missing parent

12.2.3 Organization of the Polysemy Instances in the Redundant Synsets Sub-class

A specialization polysemy case considered as a merge case if it belongs to a redundant synsets pattern. For merge cases, we apply the following operation. The function `organizeRedundantInstance` implements the merge

```
10 WordNetHierarchy: struct of {S: {Synset}, E: {(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
   {Relation}, s-rank: integer};
40: s1,s2: Synset;
50 PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
60 function organizeRedundantInstance(PolysemyInstance polyInstance)
70 s1 := polyInstance.s1; s2 := polyInstance.s2;
80 if (s1.s-rank < s2.s-rank) then
90  {s1.gloss := s1.gloss." or ".s2.gloss;
100   s1.relatoins := s1.relations U s2.relations
110   removeSynset(s2);
120   removeRedundantRelations(s1.relations);
130   return s2;}
140 else{s2.gloss := s2.gloss." or ".s1.gloss;
150   s2.relations := s2.relations U s1.relations
160  removeSynset(s1);
170  removeRedundantRelations(s2.relations);
180  return s1;}
190}
```

Figure 12.7: Pseudo code for merge operation

operation as follows. Based on the preferred sense (synset rank) of the synsets in the input `polysemyInstance`:

1 The function determines preferred synset of the polysemous terms(line 110).

2 If s1 is preferred synset, then:

a) The gloss of s1 is modified such that it is concatenated to the gloss of s2 (line 90).

b) The relations of s1 are modified such that they include also the relations of s2 (line 100).

c) The synset s2 is removed from wordNet (line 110).

d) Redundant relations are removed from s1 (line 120).

3 If s2 is preferred synset, then:

a) The gloss of s2 is modified such that it is concatenated to the gloss of s1 (line 140).

b) The relations of s2 are modified such that they include also the relations of s1 (line 150).

c) The synset s1 is removed from wordNet (line 160).

d) Redundant relations are removed from s2 (line 170).

Applying the merge operation on the polysemy instance in Figure 12.8 is shown in Figure 12.9.

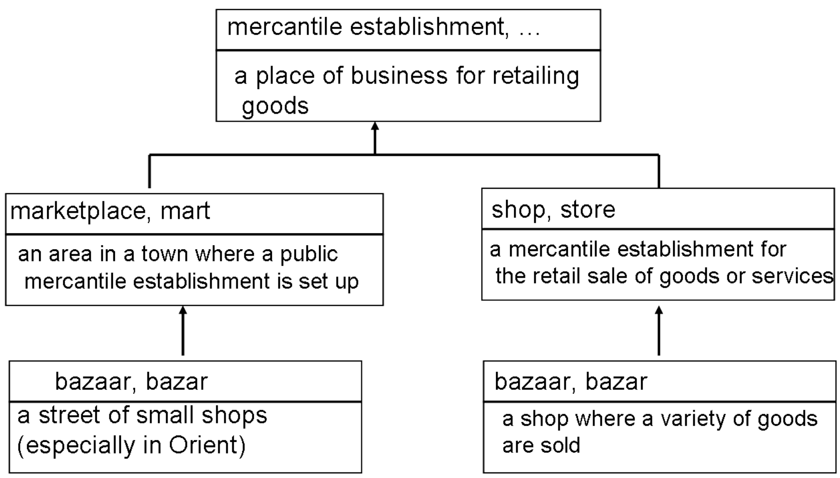

Figure 12.8: An example for redundant specialization polysemy instance

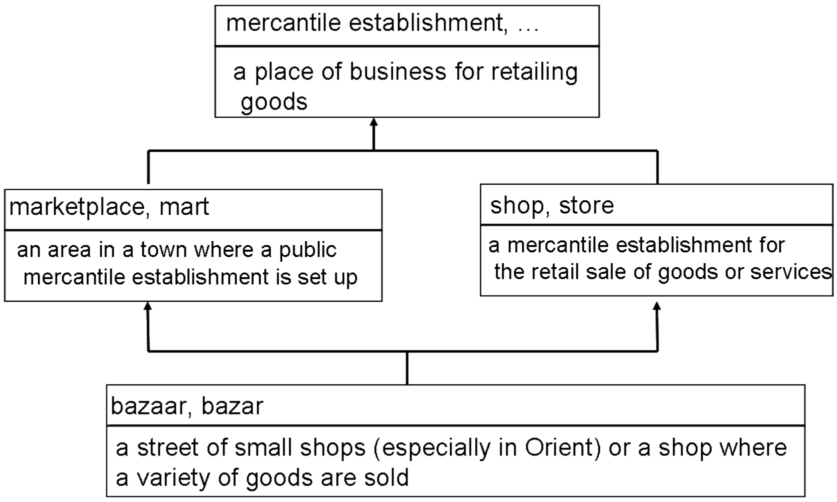

Figure 12.9: An example for a merge operation

12.3 Specialization Polysemy Organization Rules

The relation between terms and synsets in WordNet is many to many. This means that it is possible for a term, or a synset to participate in more than one polysemy relation or operation of the same type (e.g., missing parent operation). Considering such cases is very important, since the specialization polysemy operations make changes in the hierarchical structure and the synset synonyms. An example for changes in the hierarchical structure is the creating of a new synset in a missing parent operation. Changes in the synset synonyms affect the criteria for determining the polysemy operations between the synsets in specialization polysemy cases. The relation between specialization polysemy synsets is a binary relation and the specialization polysemy operations are applied pair wise. Before applying the specialization polysemy operations, we arrange the the specialization polysemy instances In the following, we explain the criteria that we are using to arrange the instances. In Figure 12.10, we see an extreme case of correlation between specialization polysemy instances. In this example, we can see the following correlated terms and synsets:

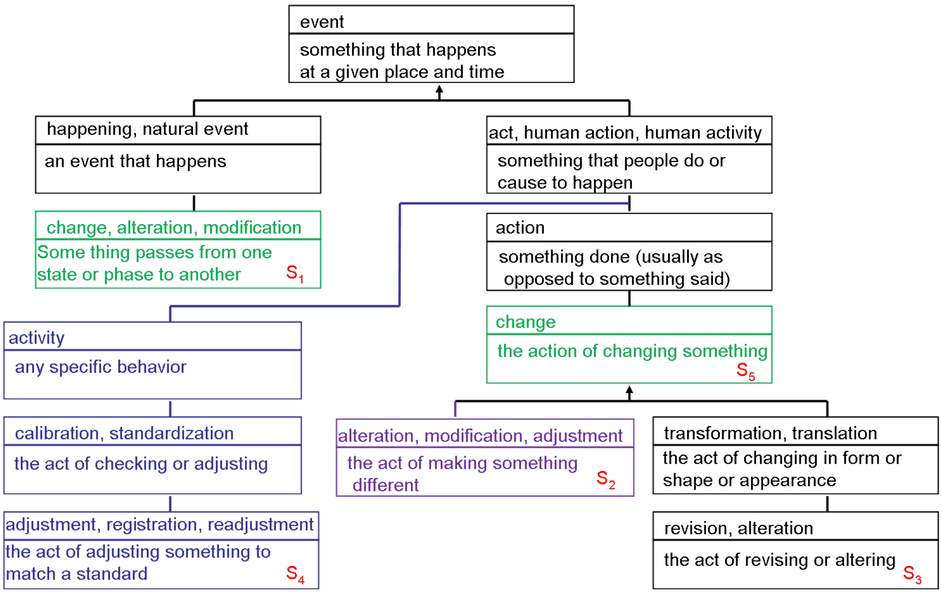

Figure 12.10: Example for correlated polysemy instances

The terms `alternation` and `modification` are found in s_1 and s_2. The term `alternation` in s_1, s_2, and s_3. At the same time we find the term `change` in s_1 and s_5 and the term `adjustment` in s_2 and s_4. The synset s_2 participates in two polysemy instances. The instance $[\{alteration, modification\}, s_1, s_2]$ corresponds to a missing parent operation and the instance $[\{adjustment\}, s_2, s_4]$ that corresponds to another missing parent operation. To handle such cases, we propose the following rules:

i *Synset level rule*: We apply the operations in a top down manner. For example, following this rule, we apply the operation on the polysemy instance $[\{change\}, s_1, s_5]$ before the polysemy instance $[\{alteration, modification\}, s_1, s_2]$.

ii *Number of polysemous terms rule*: We order the operations according to the number of polysemous terms in polysemy instances. Following this rule, we apply the operation on the polysemy instance $[\{alteration, modification\}, s_1, s_2]$ before the operation on the polysemy instance $[\{alteration\}, s_2, s_3]$. The operations on the polysemy instances $[\{alteration\}, s_2, s_3]$ and the operation on $[\{alteration\}, s_2, s_4]$ have the same priority.

iii *Resulting changes rule*: in case a synset is participating in more than one operation, the type of operation may change according to resulting changes from previous operations. For example, the operation on the polysemy instance $[\{alteration\}, s_2, s_4]$ is a missing parent operation. The result of the operation on $[\{alteration, modification\}, s_1, s_2]$ that shall be applied before, leads to changing the operation from a missing parent operation to a missing relation operation.

iv *Relation redundancy rule*: A hyponym relation between two synsets is redundant as illustrated in Figure 12.11

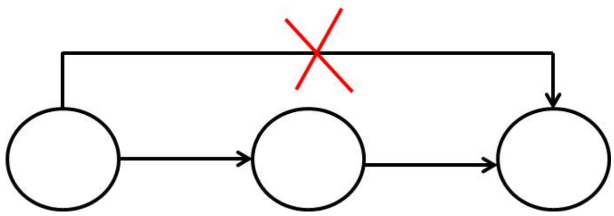

Figure 12.11: Redundant hyponym relation

In Figure 12.12, we show the final result of applying the operations on the synsets in Figure 12.10.

In Figure 12.12, the red colored lines and synsets are newly added. We apply the operations in the following order:

1. Missing relation operation on s_1, s_5 (according to the synset level rule). This affects s_1 and s_5 in the following way. We connect s_1 to the synset `happening`. The synset s_1 now is a hyponym of s_5 and the term `change` is removed from s_1.

2. The operation on the synsets s_1 and s_2 has changed now to a missing relation instead of the original operation missing parent.

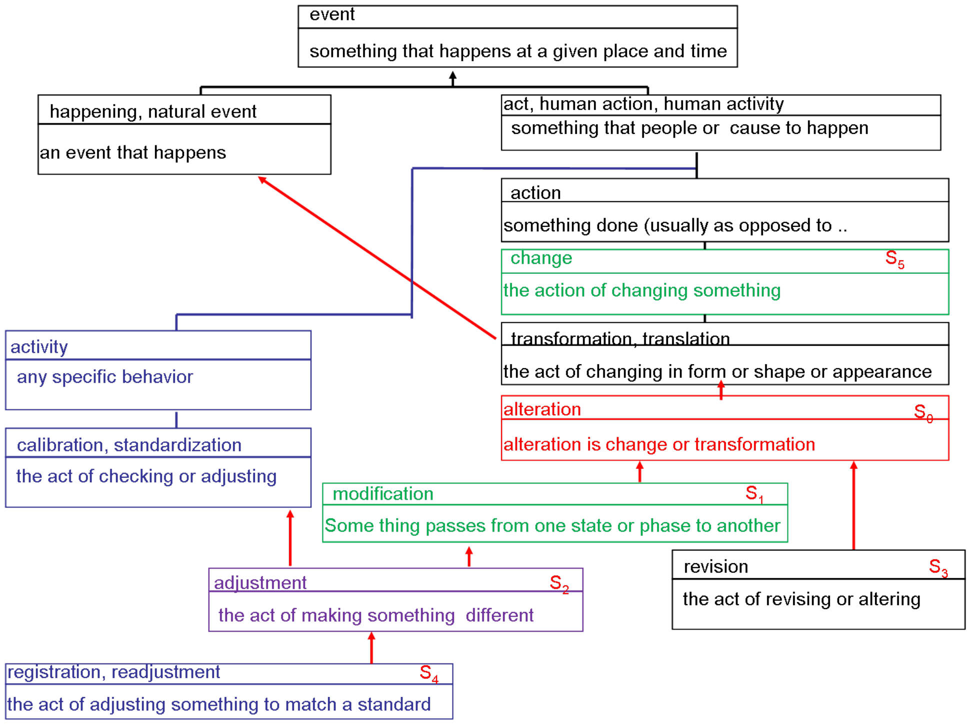

Figure 12.12: Solving correlated Polysemy instances

3. We apply the missing relation operation on s_1, s_2 (according to the number of shared terms rule) . The synset s_2 is connected to s_1. The relation between s_2 and the synset change is removed due to the relation redundancy rule. The terms `alteration` and `modification` are removed from s_2.

4. The operation on s_2, s_4 has changed to missing relation instead of the original missing parent. There is no change in the operation s_2, s_3.

5. Missing parent operation on s_2, s_3. This leads to creating a new synset s_0. The synset s_0 has the term `alteration` only. The synsets s_2 and s_3 are connected to s_0. The relation between s_3 and `transformation` is removed due to the relation redundancy rule.

6. Missing relation operation on s_2, s_4. The term `adjustment` has been removed from s_4.

12.3.1 Specialization Polysemy Organization Top level Algorithm

In the Figure 12.13, we show the specialization polysemy organization top level algorithm `organizeSpecializationPolysemy`.

```
Hierarchy: struct of {N:list of Synset; E: list of <Synset,Synset>};
WH: WordNet Hierarchy
PolysemyInstance: struct of {ts: list of Term; s1: Synset; S2: Synset};
specPolyInstances: list of PolysemyInstance;
function organizeSpecializationPolysemy(){
orderInstancesBySynsetLevel(specPolyInstances);
foreach polyInstance in specPolyInstances do{
coRrelaredInstances: list of PolysemyInstance;
coRrelaredInstances := getCoRrelaredInstances(polyInstance,
 coRrelaredInstances, specPolyInstances);
    orderInstancesBySharedTerms(coRrelaredInstances);
    applySpecializationPolysemyOperations(coRrelaredInstances);
 }
}
```

Figure 12.13: Pseudo code for the specialization polysemy organization top level algorithm

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.

2. `polyInstance`: A variable of type `PolysemyInstance`.

3. `coRrelaredInstances`, `specPolyInstances`: list of `PolysemyInstance`.

The input/output of the function:

- The input: `specPolyInstances`.

- The output: No output, the operations are performed on `wH`.

The function works as follows:

1. The function orders the polysemy instances according to the `Synset level rule` using the function `orderInstancesBySynsetLevel` which is described in 12.14 (line 60).

2. The function iterates over the specialization polysemy instances and performs the following (lines 70 - 120):

   a) To apply the `number of shared terms rule`, we use the function `getCoRrelaredInstances` which is described in ??. This function computes the correlated polysemy instances of each polysemy instance (i.e, all instances that share one or more terms with `polyInstance`).

   b) The correlated polysemy instances are stored in `coRrelaredInstances`.

   c) The function orders the polysemy instances in `coRrelaredInstances` according to the `number of shared terms rule` using the function `orderInstancesBySharedTerms` which is described in 12.15.

   d) The function applies the polysemy operations on the polysemy instances in the list `coRrelaredInstances` using the function `applySpecializationPolysemyOperations` which is described in ??.

In Figure 12.14, we present the function `orderInstancesBySynsetLevel`.

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Synset`: A synset data structure as defined in definition 2.

3. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
 {Relation}, s-rank: integer};
s1a,s1b,s2a,s2b: Synset;
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
specPolyInstances: list of PolysemyInstance
function orderInstancesBySynsetLevel(specPolyInstances){
foreach polyInstance1 in specPolyInstances do{
 s1a := polyInstance1.s1;
 s1b := polyInstance1.s2;
 root1 := getLeastCommonSubsumer(s1a,s1b);
 foreach polyInstance2 != polyInstance1 in specPolyInstances do{
 s2a  :=  polyInstance2.s1;
 s2b := polyInstance2.s2;
 root2 := getLeastCommonSubsumer(s2a,s2b);
 if(getSynsetHeight(root1) < getSynsetHeight(root2)) then {
 swap(polyCase1,polyCase2);
 }
 }
}
```

Figure 12.14: Pseudo code for orderInstancesBySynsetLevel algorithm

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `s1a,s1b,s2a,s2b`: A variables of type `Synset`.
3. `polyInstance`: A variable of type `PolysemyInstance`.
4. `specPolyInstances`: A list of `PolysemyInstance` in wordNet.

The input/output of the function:

- The input: `specPolyInstances`.
- The output: No output, call by reference function where the ordering is performed on the input.

The function `orderInstancesBySynsetLevel` is a sorting algorithm that sorts the polysemy instances based on the distance between the least common subsumer of the polysemy instances synsets.

In Figure 12.15, we show the function `orderInstancesBySharedTerms` that sorts the polysemy instances based on the number of polysemous terms in a polysemy instance.

The function uses the following data structures:

1. `Term`: A term as defined in definition 1.
2. `PolysemyInstance`: A polysemy instance as defined in definition 9.

The function uses the following variables:

1. `polyInstance1, polyInstance2`: A variable of type `PolysemyInstance`.
2. `polyInstances`: A list of `PolysemyInstance`.
3. `terms1,terms2`: A list of Term.

The input/output of the function:

- The input: `polyInstances`.

```
Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
terms1,terms2: list of Term;
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstances: list of PolysemyInstance;
 polyInstance1,polyInstance2: PolysemyInstance;
function orderInstancesBySharedTerms(polyInstances){
foreach polyInstance1 in polyInstances do{
terms1  :=polyInstance1.terms;
foreach polyInstance2!=polyInstance1 in polyInstances do{
  terms2 = polyInstance2.terms ;
  if(|terms2| < |terms2|) then{
  swap(polyInstance1,polyInstance2);
    }
 }
}
}
```

Figure 12.15: Pseudo code for orderInstancesBySharedTerms algorithm

- The output: No output, call by reference function where the ordering is performed on the input.

The function `orderInstancesBySharedTerms` is a sorting algorithm that sorts the polysemy instances based on the shared polysemous terms between the polysemy instances.
In Figure 12.16, we show the recursive function `getCoRrelatedInsatnces` that computes the correlated polysemy instances.
The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.

2. `Term`: Term as defined in definition 1.

3. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
term: Term;
terms1,terms2: list of Term;
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
 polyInstance, polyInstance1,polyInstance2: PolysemyInstance;
correlatedPolyInstances, specPolyInstances: list of PolysemyInstance;
function getCoRrelaredInstances(polyInstance, correlatedPolyInstances,
   specPolyInstances){
 terms1 = polyCase.terms;
 foreach term in terms1 do{
     foreach polyInstance1 in specPolyInstances do{
    terms2 = polyCase1.terms;
    if(terms2.contains(term)) then {
        if(!dependentPolyInstances.contains(polyInstance1)) then{
       dependentPolyInstances.add(polyInstance1);
       getCoRrelaredInstances(polyInstance1, dependentPolyInstances,
   specPolyInstances);
     }
      }
  }
  }
}
```

Figure 12.16: Pseudo code for getCoRrelaredInstances algorithm

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `term`: A variable of type `Term`.
3. `terms1, terms2`: list of `Term`.
4. `polyInstance, polyInstance1`: Variables of type `PolysemyInstance`.
5. `correlatedPolyInstances, specPolyInstances`: list of `PolysemyInstance`.

The input/output of the function:

- The input:`polyInstance, correlatedPolyInstances, specPolyInstances`.
- The output: No output, call by reference function where the function stores the correlated polysemy instances in the list `correlatedPolyInstances`.

The function computes the correlated polysemy instances of a polysemy instances. Two polysemy instances are correlated if they share one or more polysemous terms.

In Figure 12.17, we describe `applySpecializationPolysemyOperations` that is used to apply the polysemy operations as described 12.2.

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstance: PolysemyInstance;
correlatedPolyInstances: list of PolysemyInstance;
function applySpecializationPolysemyOperations(correlatedPolyInstances){
 foreach polyseyInstance in correlatedPolyInstances do{
applyOperation(polyInstance);
}
```

Figure 12.17: Pseudo code for applySpecializationPolysemyOperations algorithm

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `polyInstance`: Variables of type `PolysemyInstance`.
3. `correlatedPolyInstances`: list of `PolysemyInstance`.

The input/output of the function:

- The input: `correlatedPolyInstances`.
- The output: No output, the operations are performed on `wH`.

The function iterates over the input polysemy instances to apply the polysemy operation by calling the function `applyOperation` which is described in Figure 12.18.

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Synset`: A synset data structure as defined in definition 2.
3. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `polyInstance`: A variable of type `PolysemyInstance`.
3. `s1,s2, result`: A variables of type `Synset`.
4. `operatedSynsets`: A hash map of (`Synset` X `Synset`).

```
WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
wH: WordNetHierarchy;
Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
 {Relation}, s-rank: integer};
s1,s2,result: Synset;
operatedSynsets: list of (Synset * Synset);
PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
polyInstance: PolysemyInstance;
function applyOperation(polyInstance){
     s1 := polyInstance.s1;
     s2 := polyInstance.s2;
    if(isRedundantInstance(polyInstance)) then {
    result :=organizeRedundantInstance(polyInstance);
     if(result=s1) then {
       operatedSynsets[s1]=s2;}else{
       operatedSynsets[s2]= s1;}
  } else{
  if(isMissingRelationInstance(polyInstance)) then {
    result := organizeMissingRelationInstance(polyInstance);
     if(result=s1) then {
       operatedSynsets[s1]=s2;}else{
       operatedSynsets[s2]= s1;}
    }else{
    if(isMissingParentInstance(polyInstance)) then {
       result :=organizeMissingParentInstance(polyInstance);
       operatedSynsets[s1]:= parent;
     operatedSynsets[s2]:= parent;
      }else{
      applyPropagatedOperation(polyInstance);
     }
   }
}
```

Figure 12.18: Pseudo code for applyOperation algorithm

The input/output of the function:

- The input: `polyInstance`.
- The output: No output, the operations are performed on `wH`.

The function `applyOperation` works as follows.

1. It uses the functions `isMissingRelationInstance`, `isMissingParentInstance`, and `isRedundantInstance` to decide the polysemy operation on the input according to definitions 35, 36, and 37 respectively.
2. It uses the hash map `operatedSynsets` to store the operated synsets. We need this to keep track on changes in the operated synsets of the polysemy instances that participate in more than one polysemy operation.
3. If no operation is applicable on the input polysemy instance due to changes from previous operations, the function calls `applyPropagatedOperation` which is described in Figure 12.19.

The function uses the following data structures:

1. `WordNetHierarchy`: WordNet hierarchy as defined in definition 5.
2. `Term`: A term data structure as defined in definition 1.
3. `Synset`: A synset data structure as defined in definition 2.
4. `PolysemyInstance`: Polysemy Instance as defined in definition 9.

The function uses the following variables:

1. `wH`: An object that corresponds to the current instance of `WordNetHierarchy`.
2. `terms`: A list of Term.
3. `polyInstance`, `newPolyInstance`: Variables of type `PolysemyInstance`.

```
10 WordNetHierarchy: struct of {S:{Synset}, E:{(Synset, Synset)}};
20 wH: WordNetHierarchy;
30 Term: struct of {lemma: string, cat: grammatical category, t-rank: integer};
40 terms: list of Term;
50 Synset: struct of {terms: {Term}, label: Term, gloss: String, relations:
    {Relation}, s-rank: integer};
60 s1,s2: Synset;
70 operatedSynsets: list of (Synset * Synset);
80 PolysemyInstance: struct of {terms: {Term}; s1: Synset; S2: Synset};
90 polyInstance: PolysemyInstance;
100 function applyPropagatedOperation(polyInstance){
110  s1 := polyInstance.s1;
120  s2 := polyInstance.s2;
130  if(!operatedSynsets.contains(s1) || !operatedSynsets.contains(s2)) then {
140    if(operatedSynsets[s1]!=null) then {
150      s1 := operatedSynsets[s1];
160    }else{
170      if(operatedSynsets[s2]!=null) then {
180        s2 := operatedSynsets[s2];
190      }
200    }
210  }
220  terms = getComonTerms(s1.terms, s2.terms);
230  if(s1!=null && s2!=null && terms!=null) then{
240    PolysemyInstance newPolyInstance := new PolysemyInstance(terms,s1,s2);
250    applyOperations(newPolyInstance);
260  }
170}
```

Figure 12.19: Pseudo code for applyPropagatedOperation algorithm

4. `s1,s2`: A variables of type `Synset`.

5. `operatedSynsets`: A hash map of (`Synset` X `Synset`).

The input/output of the function:

- The input: `polyInstance`.

- The output: No output, the operations are performed on `wH`.

The function `applyPropagatedOperation` checks which synset is not operated and which one is already operated. Both synsets are already operated means that the polysemy operation is full ended and there is no need for further processing. If only one synset in is operated it searches for its corresponding synset in the global list `operatedSynsets`. If there is such synset in `operatedSynsets`, a new polysemy instance is constructed and the corresponding polysemy operation is applied on the new polysemy instance. Otherwise, the function stops.

12.4 Solving the Problem of Unspecified Information in WordNet Algorithm

Solving Homonymy

We propose the lexical relation is_homograph to denote that terms are homographs. We represent this relation as ⟨`source_category, is_homograph, target_category`⟩ , where `source_category` and `target_category` belong to the same grammatical category. This relation is bidirectional. For example, this relation holds between the term term `saki` in #1 and the term term `saki` in #2.

#1 `sake, saki, rice beer: Japanese alcoholic beverage made from fermented rice.`

#2 `saki: small arboreal monkey of tropical South America with long hair and bushy nonprehensile tail.`

Solving Metaphoric Polysemy

We propose the semantic relation is_metaphor to denote the metaphoric relation between the metaphoric meaning and literal meaning of a metaphoric polysemy case. We represent this relation as ⟨`source_category, is_metaphor, target_category`⟩ , where `source_category` and `target_category` belong to the same grammatical category. This relation is not bidirectional. In the cases, where this relation is applicable, we need to specify the literal meaning and the metaphoric meaning. For example, ⟨`#2, is_metaphor, #1`⟩ denotes the metaphoric relation between the following two meanings of `coolness`.

#1 `chilliness, coolness, nip: the property of being moderately cold.`

#2 `coolness, imperturbability: calm and unruffled self-assurance.`

Part IV

Results

Chapter 13

Results and Evaluation

In the following, we present the results and the evaluation of our approach.

13.1 Solving Compound Noun Polysemy Results

In the folloiwng, we present the results of S1.P1 in our approach.

13.1.1 Compound Noun polysemy Discovery Algorithm Results

Table 13.1 shows the results of the compound noun polysemy discovery algorithm that returned 3407 possible compound noun polysemous terms. These terms belong to 4918 synsets. The total number of compound noun polysemous instances is 15651 instances.

#Compound noun polysemous terms	3407
#Compound noun polysemous synsets	4918
#Compound noun polysemous instances	15651

Table 13.1: Results of Compound Noun Polysemy Discovery algorithm

13.1.2 Manual Validation Results

Table 13.2 shows the results of the manual validation process where 1905 terms are classified to be compound noun polysemous terms. These terms belong to 2547 synsets. These synsets belong to 11008 compound polysemy instances.

#Compound noun polysemous terms	1905
#Compound noun polysemous synsets	2547
#Compound noun polysemous instances	11088

Table 13.2: Results of compound noun polysemy after validation

13.1.3 Disambiguation Algorithm Results

In the following, we present the final result of S1.P1 of our Approach. In Table 13.3, we give an overview about the nouns in the resulting WordNet. The table shows a reduction in the number to 127260 senses. There is no

#Nouns	104290
#Synsets	74314
#Senses	127260

Table 13.3: Number of nouns, noun senses and noun synsets in WordNet 2.1 After S1.P1

change in the number of terms or synsets. In Table 13.4, we compute the following averages. The average sense number per noun is about 1.22. In

#Noun per synset	1.4
#Noun per sense	0.82
#Synset per noun	0.71
#Sense per noun	$\approx$ 1.22

Table 13.4: Polysemy average in WordNet 2.1 After S1.P1

Table 13.5 and 13.6, we consider the polysemous nouns only.

In Table 13.7, we show the percentage of the polysemous nouns, senses, and synsets in the resulting WordNet.

#Nouns	13820
#Synsets	27420
#Senses	52573

Table 13.5: Number of polysemous nouns, polysemous noun senses and polysemous noun synsets in WordNet 2.1 After S1.P1

#Noun per synset	0.50
#Noun per sense	$\approx$ 0.26
#Synset per noun	1.98
#Sense per noun	$\approx$ 3.8

Table 13.6: Polysemy average in polysemous nouns in WordNet 2.1 After S1.P1

13.2 Solving Specialization Polysemy Results

In the following, we present the results of S1.P2 of our approach.

13.2.1 Pattern Discovery Algorithm Results

The number of polysemy instances computed by the polysemy instances discovery algorithm is 41306 polysemy instances. Based on the type compatibility criterion, the algorithm classified these instances into 25333 type incompatible polysemy instances and 15973 type compatible polysemy instances. The type incompatible polysemy instances belong to 73 regular structural patterns and 7 single ton patterns. In the following table, we present the distribution of type incompatible instances. The type compatible patterns are discussed in the next section.

Specialization polysemy instances do not belong to these patterns accord-

% of polysemous nouns	13.25%
% of polysemous senses	40.36%
% of polysemous synsets	36.9%

Table 13.7: Polysemy average in polysemous nouns in WordNet 2.1 After S1.P1

Pattern root	# patterns	#polysemy instances
entity	4	11292
physical entity	10	2872
abstract entity	0	0
abstraction	22	9603
physical object	37	1566
Total	73	25333

Table 13.8: Type incompatible polysemy excluded by the algorithm

ing to the exclusiveness property. The polysemy classes of the identified type incompatible polysemy instances may be homonymy, metaphoric or metonymy.

13.2.2 Pattern Classification Results

The algorithm returned 15973 type compatible polysemy candidates. After the pattern classification, 1396 instances have been classified to belong to type incompatible polysemy and have been excluded. These instances belong to 7 patterns. Table 13.9 shows the excluded patterns with the number of their corresponding polysemy instances. The total polysemy

Structural pattern	#polysemy instances
⟨psychological feature, cognition, event⟩	985
⟨whole, artifact, natural object⟩	201
⟨communication, message, written communication⟩	110
⟨cognition, content, process⟩	79
⟨communication, message, signal⟩	14
⟨thing, body of water, part⟩	4
⟨thing, part, unit⟩	3
Total	1396

Table 13.9: Type incompatible polysemy excluded by the pattern classification

instances after excluding the type incompatible instances are 14577 instances. These instances are divided in two groups as follows. 12988 of these instances belong to 1028 regular type compatible patterns and 1569 instances belong to single tone patterns. The regular patterns are grouped into 67 groups according to the number of the polysemy instances that belong to these patterns as shown in Table 13.10. Notice that the number of the pattern with two, three or four instances is 711 patterns (about 69.1% of the patterns).

#Polysemy instances	#patterns	#Polysemy instances	#patterns
2	433	40 - 50	10
3	178	50 - 100	10
4	100	100 - 200	4
5 - 10	179	200 - 300	4
11 - 20	78	400 - 500	3
20−30	16	500 - 1000	3
30−40	9	⩾1000	1

Table 13.10: Regular type compatible structural patterns statistics

#polysemy class	#patterns	#instances
Specialization polysemy	823	9902
Metaphoric Polysemy	134	1697
Homonymy	71	1389
Total	1028	12988

Table 13.11: Classification of the regular structural patterns

13.2.3 Removing False Positives Results

In Table 13.12, we show the results removing false results, where we see that the average false positives is about 17%. In Table 13.14, we show the results of the singleton pattern classification. In Table 13.13, we show validations for sample patterns.

#polysemy class	#instances	#false positives	percentage
Specialization polysemy	9902	1740	17.57%
Metaphoric Polysemy	1697	175	10.3%
Homonymy	1389	295	21.1%
Total	12988	2210	17%

Table 13.12: False Positives in Pattern Classification

Structural pattern	Polysemy class	#polysemy instances	#false positives
Common parent	Spec. Polysemy	2879	180
$\langle event, act, happening\rangle$	Spec. Polysemy	707	348
$\langle act, action, activity\rangle$	Spec. Polysemy	429	23
$\langle organism, animal, person\rangle$	Metaphoric	326	74
$\langle event, act, group\quad action\rangle$	Spec. Polysemy	345	71
$\langle attribute, quality, state\rangle$	Spec. Polysemy	315	0
$\langle attribute, property, state\rangle$	Metaphoric	329	0
$\langle attribute, quality, trait\rangle$	Spec. Polysemy	132	44
$\langle vascular\quad plant, herb, woody\quad plant\rangle$	Spec. Polysemy	56	0
$\langle woody\quad plant, shrub, tree\rangle$	Spec. Polysemy	36	0

Table 13.13: Sample pattern validation

Polysemy class	#instances
Specialization polysemy	1128
Metaphoric Polysemy	205
Homonymy	236
Total	1569

Table 13.14: Single Ton polysemy Classification

13.2.4 Polysemy Operations Algorithm Results

In the following, we present the final result of S1.P2 of our Approach, we show some statistics about nouns in the resulting WordNet. In Table 13.3, we give an overview about the nouns in WordNet after applying our algorithm. The table shows that WordNet a reduction in the number to

#Nouns	104290
#Synsets	74712
#Senses	119312

Table 13.15: Number of nouns, noun senses and noun synsets in WordNet 2.1 After S1.P2

119312 senses. There is no change in the number of terms. On the other hand there is increase in the number of synsets synsets. In Table 13.16, we compute the following averages. The average sense number per noun is

#Noun per synset	1.395
#Noun per sense	0.87
#Synset per noun	0.716
#Sense per noun	$\approx$ 1.144

Table 13.16: Polysemy average in WordNet 2.1 After S1.P2

about 1.22. In Table 13.17 and 13.18, we consider the polysemous nouns only.

#Polysemous nouns	10998
#Polysemous synsets	21456
#Polysemous senses	35433

Table 13.17: Number of polysemous nouns, polysemous noun senses and polysemous noun synsets in WordNet 2.1 after S1.P2

In Table 13.19, we show the percentage of the polysemous nouns, senses, and synsets in the resulting WordNet.

#Polysemous noun per Polysemous synset	0.512
#Polysemous noun per polysemous sense	≈ 0.31
#Polysemous synset per polysemous noun	1.95
#Polysemous sense per polysemous noun	≈ 3.22

Table 13.18: Polysemy average in polysemous nouns in WordNet 2.1 After S1.P2

% of polysemous Nouns	10.54%
% of polysemous senses	29.7%
% of polysemous synsets	28.7%

Table 13.19: Polysemy average in polysemous nouns in WordNet 2.1 After S1.P2

13.2.5 Solving the problem of unspecified information in WordNet Algorithm

In the following, we present the results of S2 in our approach. Table 13.20 shows the discovered homonymy and polysemy instances.

Table 13.21 shows the discovered metaphoric polysemy instances.

Table 13.22 shows the discovered type incompatible polysemy instances. We think that the majority of these instances belong to the metonymy polysemy. However, they include also homonymy and metaphoric polysemy instances. For example, the term `book` that we have discussed in Chapter 4 has in the resulting WordNet the following three senses.

```
# book:  a number of sheets (ticket or stamps etc.)  bound together on one edge;
  "he bought a book of stamps" OR a written work or composition that has been
  published (printed on pages bound together); "I am reading a good book on economics".

# book:  a major division of a long written composition; "the book of Isaiah".

# book:  a collection of playing cards satisfying the rules of a card game.
```

#senses	#nouns	#polysemy instances	#homonymy instances	%
2	7818	6729	980	14.56%
3	1991	5222	264	5%
4	679	3570	165	4.62%
5	269	2320	109	4.7%
6	108	1417	67	4.72%
7	66	1183	48	4.5%
8	23	576	26	4.5%
9	17	522	23	4.4%
10	15	613	28	4.56%
11	4	220	3	1.36%
12	6	342	6	1.75%
13	1	66	1	1.5%
21	1	153	4	2.16%
Total	10998	22933	1724	7.51%

Table 13.20: Discovered homonymy Instances in WordNet

#senses	#nouns	#polysemy instances	#metaphoric instances	%
2	7818	6729	700	10.4%
3	1991	5222	433	8.29%
4	679	3570	199	5.57%
5	269	2320	122	5.26%
6	108	1417	64	4.5%
7	66	1183	51	4.3%
8	23	576	19	3.3%
9	17	522	17	3.25%
10	15	613	31	5%
11	4	220	16	7.27%
12	6	342	4	1.16%
13	1	66	1	1.5%
21	1	153	0	0%
Total	10998	22933	1657	7.22%

Table 13.21: Discovered metaphoric Instances in WordNet

#senses	#nouns	#polysemy instances	#type incompatible instances	%
2	7818	6729	5049	75%
3	1991	5222	4525	86.65%
4	679	3570	3206	89.8%
5	269	2320	2089	90%
6	108	1417	1286	90.75%
7	66	1183	1084	90.63%
8	23	576	509	88.36%
9	17	522	482	92.33%
10	15	613	554	90.37%
11	4	220	201	91.3%
12	6	342	332	97%
13	1	66	66	100%
21	1	153	153	100%
Total	10998	22933	19552	85.25%

Table 13.22: Discovered type incompatible instances in WordNet

13.3 Evaluation

For the manual validation described in Chapter 11 and the evaluation process described in this section, we have developed a special user interface 13.1. This user interface provides the local view of the polysemy instances. For each polysemy instance, the user can view also the polysemy type of the displayed polysemy instance and the polysemy operation (applicable for specialization polysemy instances). The user can then agree with the suggested polysemy type/ polysemy operation or he can choose one of the provided alternative polysemy types. If the user can not decide, he can choose "No decision". To evaluate our approach, 3797 type compatible polysemy instances have been evaluated by two evaluators. In Table 13.23, we report the statistics of the evaluation, where we show the following:

a *Total agreement*: Measures the number of polysemy instances where both evaluators agrees with our approach (corresponds to first column

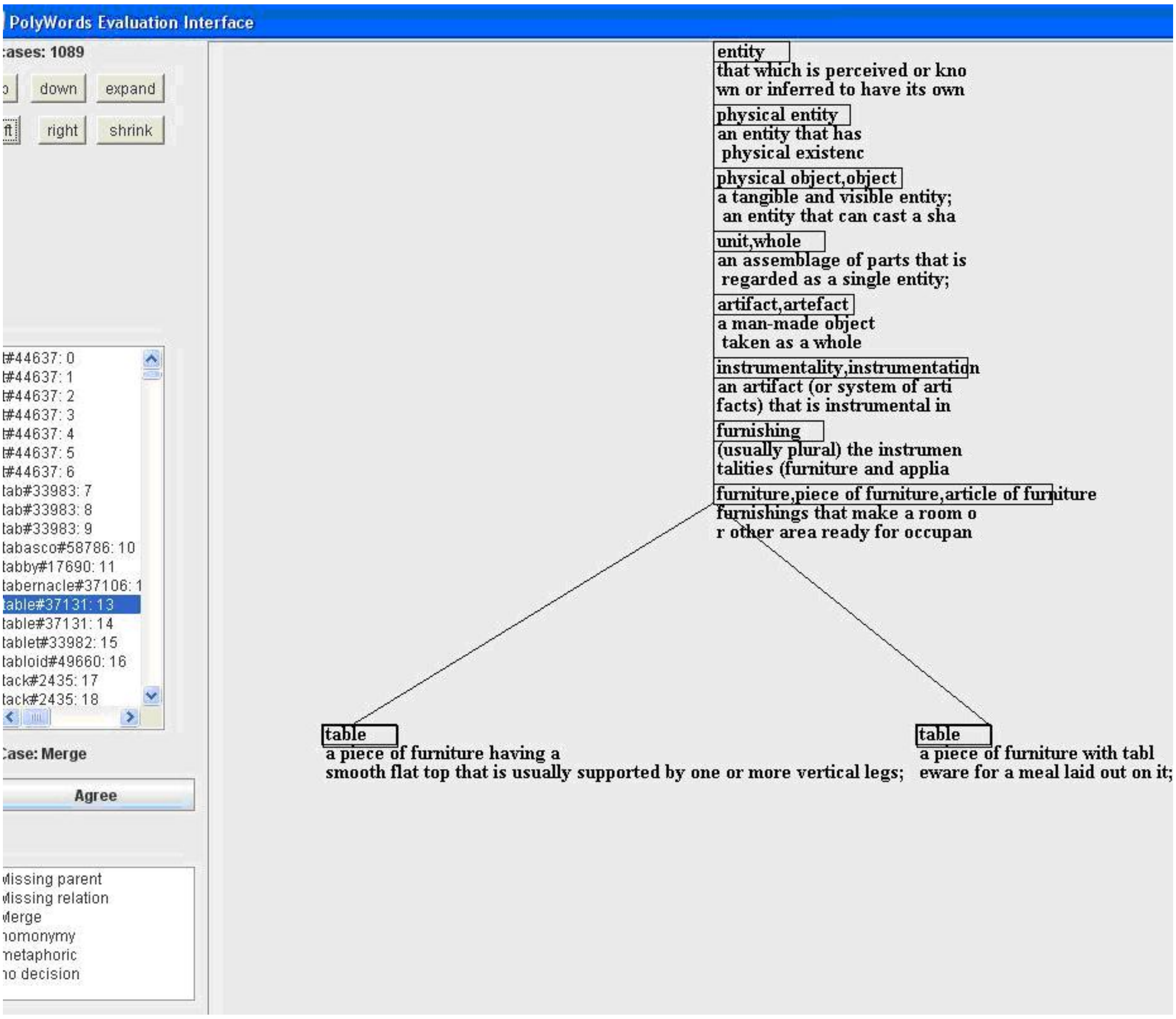

Figure 13.1: Polysemy Evaluation Interface

in the table).

b *Partial agreement* Measures the number of polysemy instances where the at least one of the evaluators agrees with our approach (corresponds to second and third columns in the table).

c *Disagreement* Measures disagreement between the approach and the evaluators (corresponds to last three columns in the table).

In the following tables, a refers to our approach, e_1, e_2 refer to evaluator1 and evaluator 2 respectively. In another evaluation, 1020 cases have been

$e_1 = e_2 = a$	$a = e_1$	$a = e_2$	$a = e_1 \wedge a! = e_2$	$a = e_2 \wedge a! = e_1$	$a \neq e_1 \neq e_2$
3665 (96.5%)	3621 (95.3%)	3600 (94.8%)	77 (2.1%)	55 (1.5%)	9 (0.23%)

Table 13.23: Polysemy average in polysemous nouns in WordNet 2.1

evaluated by another two evaluators, where we measured the agreement on the polysemy classification and the specialization polysemy operation. In Table 13.24, we report the statistics of the evaluation, where the column polysemy type refers to homonymy, metaphoric, metonymy, or specialization polysemy and the column polysemy operation refers to creating missing parent, adding missing relation, or merging operation. Note that, polysemy operation is applicable in case of specialization polysemy.

	Polysemy Classification Agreement	Polysemy Operation Agreement
$a = e_1$	979 (96%)	924 (90.5%)
$a = e_2$	945 (92.5%)	855 (84%)
$a = e_1 \vee a = e_2$	1006 (98.5%)	978 (96%)

Table 13.24: Polysemy average in polysemous nouns in WordNet 2.1

Chapter 14

Conclusion and Future Work

In this thesis, we have introduced an organizational approach for solving the polysemy where have reduced the high polysemy in compound noun and specialization polysemy in the case of nouns. In Addition, we have identified a subset of homonymy and metaphoric polysemy instances and denoted them explicitly in WordNet. The main idea of our approach is too much implicit information in a lexical resource is a source of noise rather than a source of knowledge.
In this approach, we have solved the following problems.

1. **The problem of the highpolysemous nature of WordNet**: We have solved this problem partially and could reduce the polysemy in wordNet in the case of nouns from 1.25 to 1.14 sense per noun, where the manual treatment in two phases of the approach guarantees the quality of the approach results. By solving the compound noun polysemy and specialization polysemy, the polysemy problem in WordNet is reduced to the metonymy problem modulo small portion of metaphoric and homonymy instances instead of the polysemy problem in five polysemy classes.

2. **The problem of unspecified information**: We have identified 15% of the polysemy instances in the resulting WordNet that belong to

homonymy and metaphoric polysemy and thus decreased the polysemy problem in a future solution to the metonymy problem.

The main contributions of this work are at two levels:
At the conceptual level, we have provided a new foundation towards the problem of polysemy. At the implementation level, we improved the quality of WordNet to maximize the accuracy of NLP and knowledge-based applications, especially in the field of the semantic search.

14.0.1 Future Work

In this thesis, we did not solve the polysemy problem in metonymy and consider solving the problem as future work, where we propose refining CORELEX as follows.

1. *Solve the high ambiguous polysemy problem*

   i Rebuild CORELEX classes;

   ii Populate the classes with corresponding polysemy instances;

   iii Classify the patterns into metonymy, metaphoric, and homonymy;

   iv Discover and handle false positives;

   v apply the underspecification method on the resulting metonymy classes;

2. *Solve the unspecified information problem*

   i Denote metaphoric and homonymy cases as described in S2 of our approach;

   ii Link the metonymy instances via the following semantic relation: `has_aspect`: to denote the relation between the meanings in a metonymy polysemy instance, where this relation holds between the base meaning of a term and the derived meanings of that term. To set up the relation we need to determine the base meaning and then relate the other derived meanings to it.

Bibliography

[1] I. L. Falkum, "The semantics and pragmatics of polysemy: A relevance-theoretic account," *PhD thesis, University College London*, 2011.

[2] C. Stokoe, "Differentiating homonymy and polysemy in information retrieval," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, (Stroudsburg, PA, USA), pp. 403–410, Association for Computational Linguistics, 2005.

[3] P. Buitelaar, "Corelex: Systematic polysemy and underspecification," *PhD thesis,Brandeis University, Department of Computer Science*, 1998.

[4] L. Barque and F.-R. Chaumartin, "Regular polysemy in wordnet.," *JLCL*, vol. 24, no. 2, pp. 5–18, 2009.

[5] W. Peters, "Extraction of implicit knowledge from wordnet," in *Proceedings of Ontolex2002 Workshop on Ontologies and Lexical Knowledge Bases*, 2002. Preceding LREC2002, Las Palmas, 2002.

[6] W. Peters and I. Peters, "Lexicalised systematic polysemy in wordnet.," in *LREC*, European Language Resources Association, 2000.

[7] A. A. Freihat, F. Giunchiglia, and B. Dutta, "Approaching regular polysemy in wordnet," *In Proceedings of the 5th International Conference*

on Information, Process, and Knowledge Management (eKNOW), pp. 63–69, nov 2013.

[8] S. N. Kim and T. Baldwin, "Word sense and semantic relations in noun compounds," *ACM Trans. Speech Lang. Process.*, vol. 10, pp. 9:1–9:17, July 2013.

[9] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, Nov. 1995.

[10] W. B. Dolan, "Word sense ambiguation: clustering related senses," in *Proceedings of COLING94*, pp. 712–716, 1994.

[11] N. Tomuro, "Semi-automatic induction of systematic polysemy from wordnet," in *In: Proceedings ACL-98 Workshop on the Use of WordNet in NLP*, pp. 108–114, 1998.

[12] R. Snow, S. Prakash, D. Jurafsky, and A. Y. Ng, "Learning to merge word senses.," in *EMNLP-CoNLL*, pp. 1005–1014, ACL, 2007.

[13] R. Mihalcea and D. I. Moldovan, "Ez.wordnet: Principles for automatic generation of a coarse grained wordnet.," in *FLAIRS Conference* (I. Russell and J. F. Kolen, eds.), pp. 454–458, AAAI Press, 2001.

[14] A. A. Freihat, F. Giunchiglia, and B. Dutta, "Solving specialization polysemy in wordnet," *International Journal of Computational Linguistics and Applications*, vol. 4, jan-june 2013.

[15] V. P. Peters W., Peters I., "Automatic sense clustering in eurowordnet," *In Proceedings of the International Conference on Language Resources and Evaluation*, pp. 409–416, 1998.

[16] J. Utt and S. Padó, "Ontology-based distinction between polysemy and homonymy," in *Proceedings of the Ninth International Conference*

on Computational Semantics, IWCS '11, (Stroudsburg, PA, USA), pp. 265–274, Association for Computational Linguistics, 2011.

[17] D. TufiŞ, R. Ion, and N. Ide, "Fine-grained word sense disambiguation based on parallel corpora, word alignment, word clustering and aligned wordnets," in *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, (Stroudsburg, PA, USA), Association for Computational Linguistics, 2004.

[18] P. Buitelaar, "A lexicon for underspecified semantic tagging," *CoRR*, vol. cmp-lg/9705011, 1997.

[19] K. Jiamjitvanich and M. Yatskevich, "Reducing polysemy in wordnet.," in *OM* (P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt, N. F. Noy, and A. Rosenthal, eds.), vol. 551 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2008.

[20] R. Navigli, "Meaningful clustering of senses helps boost word sense disambiguation performance," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, (Stroudsburg, PA, USA), pp. 105–112, Association for Computational Linguistics, 2006.

[21] R. Mandala, T. Tokunaga, and H. Tanaka, "Complementing wordnet with roget's and corpus-based thesauri for information retrieval.," in *EACL*, pp. 94–101, The Association for Computer Linguistics, 1999.

[22] F. Giunchiglia, U. Kharkevich, and I. Zaihrayeu, "Concept search: Semantics enabled syntactic search," in *Proceedings of the Workshop on Semantic Search (SemSearch 2008) at the 5th European Semantic Web Conference (ESWC 2008) , June 2, 2008, Tenerife, Spain*

(S. Bloehdorn, M. Grobelnik, P. Mika, and D. T. Tran, eds.), vol. 334 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2008.

[23] P. Edmonds and E. Agirre, "Word sense disambiguation.," *Scholarpedia*, vol. 3, no. 7, p. 4358, 2008.

[24] M. I. Mihalcea R., "Ez.wordnet: Principles for automatic generation of a coarse grained wordnet," *FLAIRS Conference*, pp. 454–458, 2001.

[25] V. F. Gonzalo J., Chugur I., "Sense clusters for information retrieval: Evidence from semcor and the eurowordnet interlingual index," *ACL-2000 Workshop on Word Senses and Multi-linguality, Association for Computational Linguistics*, pp. 10–18.

[26] A. A. Freihat, F. Giunchiglia, and B. Dutta, "Regular polysemy in wordnet and pattern based approach," *International Journal On Advances in Intelligent Systems*, jan 2013.

[27] P. W., "Detection and characterization of figurative language use in wordnet," *PhD thesis, Natural Language Processing Group, Department of Computer Science, University of Sheffield*, 2004.

[28] T. Veale, "A non-distributional approach to polysemy detection in wordnet."

[29] T. Veale, "Pathways to creativity in lexical ontologies," in *In Proceedings of the 2nd Global WordNet Conference*, 2004.

[30] M. Palmer, H. T. Dang, and C. Fellbaum, "Making fine-grained and coarse-grained sense distinctions, both manually and automatically.," *Natural Language Engineering*, vol. 13, no. 2, pp. 137–163, 2007.

[31] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: an on-line lexical database," *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, 1990.

[32] G. A. Miller and F. Hristea, "Wordnet nouns: Classes and instances.," *Computational Linguistics*, vol. 32, no. 1, pp. 1–3, 2006.

[33] A. Gangemi, N. Guarino, and A. Oltramari, "Conceptual analysis of lexical taxonomies: The case of WordNet top-level," 2001.

[34] M. Piasecki, S. Szpakowicz, and B. Broda, *A Wordnet from the Ground Up*. Oficyna Wydawnicza Politechniki Wroclawskiej, 2009.

[35] V. Verdezoto, Nervo, "Towards semi-automatic methods for improving wordnet," in *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, (Stroudsburg, PA, USA), pp. 275–284, Association for Computational Linguistics, 2011.

[36] M. Ciaramita and M. Johnson, "Supersense tagging of unknown nouns in wordnet," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 168–175, Association for Computational Linguistics, 2003.

[37] D. I. Moldovan and A. Novischi, "Word sense disambiguation of wordnet glosses.," *Computer Speech & Language*, vol. 18, no. 3, pp. 301–317, 2004.

[38] S. M. Harabagiu, G. A. Miller, and D. I. Moldovan, "WordNet 2 – a morphologically and semantically enhanced resource," in *Proc. SIGLEX 1999*, 1999.

[39] R. Navigli, P. Velardi, A. Cucchiarelli, F. Neri, and R. Cucchiarelli, "Extending and enriching wordnet with ontolearn," 2004.

[40] D. Fass, "Metonymy and metaphor: what's the difference.," in *COLING*, pp. 177–181, 1988.

[41] M. Lapata and A. Lascarides, "A probabilistic account of logical metonymy," *Comput. Linguist.*, vol. 29, pp. 261–315, June 2003.

[42] V. Evans and J. Zinken, "Figurative language in a modern theory of meaning construction: A lexical concepts and cognitive models approach."

[43] D. Gentner, B. F. Bowdle, P. Wolff, and C. Boronat, "Metaphor is like analogy," pp. 199–253, MIT Press, 2001.

[44] K. Glover, "Polysemy methods & materials models & meaning," *Master thesis, University of Essex*, 2005.

[45] I. Mani, "A theory of granularity and its application to problems of polysemy and underspecification of meaning.," in *KR* (A. G. Cohn, L. K. Schubert, and S. C. Shapiro, eds.), pp. 245–257, Morgan Kaufmann, 1998.

[46] G. E. Bakx, "Machine learning techniques for word sense disambiguation," *PhD thesis, Universitat Politecnica de Catalunya*, 2006.

[47] A. J., "Regular polysemy," *Linguistics*, pp. 5–32, 1974.

[48] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surv.*, vol. 41, pp. 10:1–10:69, Feb. 2009.

[49] J. Pustejovsky, "The generative lexicon," *Computational Linguistics*, vol. 17, 1991.

[50] N. Tomuro, "Tree-cut and a lexicon based on systematic polysemy," in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, (Stroudsburg, PA, USA), pp. 1–8, Association for Computational Linguistics, 2001.

[51] V. P. Eneko A., Magnini B., "Semeval-2007 task 01: Evaluating wsd on cross-language information retrieval," *Proceedings of the Fourth Inter-*

national Workshop on Semantic Evaluations (SemEval-2007)Chairs, 2007.

[52] J. Alvarez, "Integrating the wordnet ontology into a description logic system."

[53] S. Rudolph, "Foundations of description logics," in *Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011* (A. C. Polleres A., dÁmato C., ed.), vol. 6848 of *LNCS*, pp. 76–136, Springer, 2011.

[54] T. D. Breaux, A. I. Anton, and J. Doyle, "Semantic parameterization: A process for modeling domain descriptions," *ACM Transactions on Software Engineering Methodology*, vol. 18, no. 2, 2009.

[55] A. Marradi, "Classification, typology, taxonomy," *Quality & Quantity: International Journal of Methodology*, vol. 24, no. 2, pp. 129–157, 1990.

Appendix A

Specialization Polysemy Patterns

ability#creativity,intelligence	ability#faculty,intelligence	artifact#instrumentality,line
accomplishment#attainment,deed	act#action,activity	act#action,communication
act#action,distribution	act#action,hindrance	act#action,nonaccomplishment
act#action,rejection	act#action,speech act	act#activity,communication
act#activity,distribution	act#activity,inactivity	act#activity,judgment
act#activity,nonaccomplishment	act#activity,rejection	act#activity,speech act
act#communication,distribution	act#communication,speech act	act#distribution,speech act
act#hindrance,rejection	act#judgment,speech act	act#nonaccomplishment,rejection
act#nonaccomplishment,speech act	act#rejection,speech act	action#accomplishment,arrival
action#accomplishment,change	action#accomplishment,choice	action#accomplishment,playing
action#aggression,change	action#arrival,change	action#change,choice
activity#aid,occupation	activity#aid,operation	activity#aid,practice
activity#aid,work	activity#aid,worship	activity#attempt,control
activity#attempt,diversion	activity#attempt,work	activity#behavior,practice
activity#behavior,wrongdoing	activity#ceremony,occupation	activity#ceremony,work
activity#control,occupation	activity#control,work	activity#creation,diversion
activity#creation,occupation	activity#creation,preparation	activity#creation,procedure
activity#creation,representation	activity#creation,wrongdoing	activity#diversion,game
activity#diversion,music	activity#diversion,occupation	activity#diversion,practice
activity#diversion,work	activity#diversion,wrongdoing	activity#game,practice
activity#occupation,work	activity#operation,turn	activity#operation,work
activity#operation,wrongdoing	activity#practice,use	activity#practice,work
activity#practice,wrongdoing	activity#protection,work	activity#provision,work
activity#role,work	activity#sensory activity,work	activity#training,work
activity#turn,wrongdoing	activity#use,work	activity#use,wrongdoing
activity#work,wrongdoing	alga#brown algae,seaweed	animal#chordate,larva
animal#chordate,young	animal#invertebrate,larva	aquatic bird#swan,wading bird
area#room,storage space	arrangement#array,formation	arthropod#arachnid,insect
artifact#article,instrumentality	artifact#block,building material	artifact#building material,covering
artifact#building material,facility	artifact#building material,instrumentality	artifact#building material,structure
artifact#building material,surface	artifact#commodity,covering	artifact#commodity,creation
artifact#commodity,decoration	artifact#commodity,fabric	artifact#commodity,instrumentality
artifact#commodity,strip	artifact#commodity,structure	artifact#covering,creation
artifact#covering,decoration	artifact#covering,fabric	artifact#covering,facility
artifact#covering,instrumentality	artifact#covering,opening	artifact#covering,sheet
artifact#covering,structure	artifact#covering,way	artifact#creation,decoration
artifact#creation,fabric	artifact#creation,facility	artifact#creation,instrumentality
artifact#creation,line	artifact#creation,sheet	artifact#creation,structure
artifact#decoration,fabric	artifact#decoration,facility	artifact#decoration,instrumentality
artifact#decoration,line	artifact#decoration,strip	artifact#decoration,structure
artifact#enclosure,facility	artifact#enclosure,instrumentality	artifact#enclosure,structure
artifact#excavation,instrumentality	artifact#fabric,instrumentality	artifact#fabric,line
artifact#fabric,strip	artifact#fabric,structure	artifact#facility,instrumentality
artifact#facility,opening	artifact#facility,sheet	artifact#facility,structure
artifact#facility,way	artifact#fixture,instrumentality	artifact#fixture,structure

artifact#instrumentality,opening
artifact#instrumentality,padding
artifact#instrumentality,plaything
artifact#instrumentality,strip
artifact#instrumentality,structure
artifact#instrumentality,thing
artifact#instrumentality,track
artifact#instrumentality,way
artifact#line,sheet
artifact#opening,sheet
artifact#opening,structure
artifact#opening,surface
artifact#opening,way
artifact#paving material,surface
artifact#sheet,structure
artifact#sheet,surface
artifact#strip,surface
artifact#structure,surface
artifact#structure,way
artifact#surface,way
athlete#ballplayer,cricketer
athlete#basketball player,football player
atmospheric phenomenon#storm,weather
attitude#inclination,intolerance
attribute#property,shape
attribute#property,time
attribute#quality,shape
attribute#quality,state
attribute#quality,trait
attribute#shape,state
attribute#state,time
attribute#state,trait
attribute#state,uncheerfulness
auditory communication#music,speech
auditory communication#music,utterance
auditory communication#speech,utterance
bad person#destroyer,wrongdoer
bad person#libertine,wrongdoer
baked goods#bread,cake
bird#aquatic bird,passerine
bird#gallinaceous bird,passerine
body of water#inlet,lake
body part#external body part,organ
body part#external body part,structure
body part#feature,tissue
body part#organ,process
body part#organ,structure
body part#organ,tissue
body part#process,structure
body part#structure,tissue
building#hotel,house
building#house,place of worship
capitalist#businessperson,financier
celebration#festival,merrymaking
change of integrity#combination,joining
change of integrity#opening,separation
change of state#nullification,termination
change#change of direction,change of state
change#change of integrity,change of state
change#change of integrity,motion
change#change of magnitude,change of state
change#change of state,motion
change#change of state,movement
change#increase,transition
change#motion,motion
change#motion,movement
clothing#attire,garment
clothing#attire,woman's clothing
clothing#footwear,garment
clothing#garment,nightwear
clothing#garment,outerwear
clothing#garment,protective garment
clothing#garment,woman's clothing
cognition#ability,attitude
cognition#ability,cognitive factor
cognition#ability,information
cognition#ability,structure
cognition#attitude,content
cognition#attitude,process
cognition#cognitive factor,content
cognition#cognitive factor,information
cognition#content,information
cognition#content,practice
cognition#content,structure
cognition#information,process
cognition#information,structure
cognition#perception,process
cognition#process,process
combatant#boxer,wrestler
commodity#consumer goods,drygoods
common parent
communication#auditory communication,language
communication#auditory communication,signal
communication#document,indication
communication#document,message
communication#document,written communication
communication#expressive style,language
communication#indication,signal
communication#indication,visual communication
communication#language,message
communication#language,signal
communication#language,written communication
communication#message,visual communication
communication#signal,written communication
compound#base,organic compound
condition#difficulty,disorderliness
condition#difficulty,need
condition#difficulty,pathological state
condition#difficulty,psychological state
condition#disorder,financial condition
condition#disorder,pathological state
condition#impurity,sanitary condition
condition#pathological state,unsoundness
conifer#arborvitae,cedar
conifer#cedar,pine
conifer#hemlock,pine
conifer#pine,spruce
content#belief,goal
content#belief,idea
content#belief,knowledge domain
content#belief,representation
content#education,idea
content#goal,idea
content#idea,knowledge domain
content#idea,representation
content#knowledge domain,representation

contestant#athlete,player	court game#badminton,tennis	
covering#cloth covering,protective covering	covering#footwear,protective covering	covering#protective covering,wrapping
creation#art,representation	creation#product,representation	currency#cash,coinage
decoration#adornment,design	decoration#adornment,molding	deed#acquiring,recovery
deed#acquiring,touch	deed#causing,touch	deed#implementation,recovery
deed#propulsion,touch	definite quantity#number,unit of measurement	device#acoustic device,musical instrument
device#alarm,musical instrument	device#alarm,noisemaker	device#contraceptive,electrical device
device#electrical device,lighter	device#electrical device,mechanism	device#electronic device,instrument
device#flare,lighter	device#holding device,restraint	device#indicator,mechanism
device#instrument,mechanism	device#instrument,optical device	device#instrument,reflector
device#instrument,restraint	device#instrument,support	device#mechanism,memory device
device#mechanism,musical instrument	device#mechanism,restraint	device#mechanism,stabilizer
device#musical instrument,noisemaker	device#restraint,trap	discipline#humanistic discipline,science
disease#animal disease,communicable disease	diversion#gambling,sport	document#commercial document,legal document
english#middle english,old english	event#act,conference	event#act,group action
event#act,happening	event#act,miracle	event#act,session
event#act,social event	event#group action,happening	expressive style#device,turn of phrase
facility#course,recreational facility	feeling#desire,emotion	feeling#despair,emotion
feeling#emotion,pain	feeling#emotion,passion	feeling#emotion,sadness
feeling#emotion,shame	feeling#emotion,temper	feeling#enthusiasm,passion
feline#big cat,cat	fish#bony fish,cartilaginous fish	fish#bony fish,food fish
flower#bellwort,composite	food#baked goods,produce	food#beverage,foodstuff
food#foodstuff,nutriment	furniture#seat,table	game bird#grouse,phasianid
garment#trouser,undergarment	gathering#assembly,meeting	genus#arthropod genus,dicot genus
genus#bird genus,dicot genus	group action#assembly,social control	group action#conflict,military action
group action#conflict,social control	group action#cooperation,social control	group action#social control,transaction
group#arrangement,collection	group#arrangement,social group	group#biological group,people
group#multitude,social group	group#people,social group	group#social group,system
gum tree#eucalyptus,liquidambar	happening#change,discharge	happening#change,experience
happening#change,movement	happening#change,periodic event	happening#change,sound
happening#change,trouble	happening#contact,sound	happening#contact,trouble
happening#discharge,sound	happening#ending,movement	happening#ending,trouble
happening#juncture,periodic event	happening#movement,periodic event	happening#movement,sound
happening#movement,trouble	happening#periodic event,sound	happening#sound,trouble
herb#bedstraw,gramineous plant	herb#clover,oxalis	herb#mint,monarda
idea#concept,generalization	idea#concept,ideal	ill health#illness,infection
ill health#illness,pathology	implement#rod,sports implement	implement#tool,utensil
inhabitant#asian,european	instrumentality#connection,container	instrumentality#connection,device
instrumentality#connection,equipment	instrumentality#connection,system	instrumentality#container,conveyance
instrumentality#container,device	instrumentality#container,furnishing	instrumentality#container,implement
instrumentality#conveyance,device	instrumentality#device,equipment	instrumentality#device,implement
instrumentality#device,system	instrumentality#device,weaponry	instrumentality#equipment,implement
instrumentality#equipment,medium	instrumentality#medium,system	know-how#method,wisdomleader#aristocrat,politician
leader#head,presiding officer	leader#head,spiritual leader	

location#building,point	location#building,region	location#line,region
location#line,space	location#point,region	location#region,region
magnitude relation#rate,ratio	magnitude#amount,dimension	magnitude#dimension,size
mammal#metatherian,placental	mammal#metatherian,prototherian	mammal#placental,prototherian
material#adhesive material,discharge	material#animal material,paper	material#discharge,plant material
material#earth,mineral	material#earth,waste	measure#definite quantity,indefinite quantity
measure#definite quantity,linear measure	measure#definite quantity,point	measure#definite quantity,relative quantity
measure#definite quantity,time unit	measure#fundamental quantity,playing period	measure#fundamental quantity,point
measure#fundamental quantity,time unit	measure#indefinite quantity,point	measure#point,time interval
measure#point,time unit	mechanism#control,mechanical device	memory device#magnetic tape,recording
message#acknowledgment,approval	message#acknowledgment,statement	message#approval,statement
message#commitment,statement	message#direction,statement	message#disapproval,disrespect
message#disrespect,statement	message#information,statement	message#nonsense,statement
message#offer,statement	message#request,statement	military unit#air unit,army unit
military unit#air unit,naval unit	military unit#army unit,naval unit	motion#gesture,stroke
motion#locomotion,maneuver	motion#locomotion,travel	motion#maneuver,travel
movement#change of location,wave	music#music genre,musical composition	natural object#body,plant part
natural object#covering,plant part	natural science#earth science,life science	needlework#embroidery,sewing
null	number#constant,integer	nut tree#hickory,walnut
nutriment#course,dainty	nutriment#course,dish	nutriment#dainty,dish
nutriment#dish,meal	organism#animal,microorganism	organism#parasite,plant
organization#alliance,unit	organization#association,enterprise	organization#association,institution
organization#association,unit	organization#enterprise,unit	organization#force,unit
organization#institution,unit	organization#polity,unit	oscine#finch,thrush
oscine#finch,warbler	oscine#new world oriole,thrush	oscine#thrush,warbler
overgarment#cloak,coat	palm#fan palm,feather palm	passerine#oscine,tyrannid
passerine#oscine,wren	percoid fish#carangid fish,sciaenid fish	percoid fish#carangid fish,scombroid
percoid fish#grunt,wrasse	percoid fish#sciaenid fish,scombroid	person#adjudicator,expert
person#adjudicator,worker	person#adult,anomaly	person#adult,communicator
person#adult,creator	person#adult,enrollee	person#adult,female
person#adult,lover	person#adult,ruler	person#adult,unwelcome person
person#adventurer,communicator	person#adversary,contestant	person#advocate,drug user
person#advocate,good person	person#advocate,leader	person#advocate,lover
person#advocate,national	person#advocate,worker	person#anomaly,unwelcome person
person#bad person,capitalist	person#bad person,expert	person#bad person,inhabitant
person#bad person,juvenile	person#bad person,leader	person#bad person,quitter
person#bad person,religious person	person#bad person,traveler	person#bad person,unwelcome person
person#bad person,user	person#bad person,worker	person#capitalist,creator
person#capitalist,expert	person#combatant,contestant	person#commoner,inhabitant
person#commoner,national	person#commoner,worker	person#communicator,creator
person#communicator,entertainer	person#communicator,expert	person#communicator,literate
person#communicator,male	person#communicator,perceiver	person#communicator,unfortunate

person#communicator,unwelcome person
person#contestant,peer
person#contestant,traveler
person#creator,entertainer
person#creator,intellectual
person#creator,planner
person#disputant,warrior
person#domestic partner,leader
person#domestic partner,male
person#domestic partner,peer
person#enrollee,intellectual
person#enrollee,unskilled person
person#entertainer,juvenile
person#entertainer,occultist
person#entertainer,unwelcome person
person#entertainer,worker
person#expert,intellectual
person#expert,leader
person#expert,preserver
person#expert,scientist
person#fiduciary,preserver
person#friend,lover
person#friend,male
person#gambler,user
person#good person,worker
person#inhabitant,leader
person#inhabitant,religious person
person#inhabitant,worker
person#intellectual,literate
person#intellectual,perceiver
person#intellectual,religious person
person#intellectual,scientist
person#intellectual,unwelcome person
person#juvenile,male
person#juvenile,unwelcome person
person#leader,national
person#leader,peer
person#leader,preserver
person#leader,religious person
person#leader,ruler
person#leader,user
person#literate,scientist
person#male,relative
person#male,unwelcome person
person#nonreligious person,religious person
person#owner,unwelcome person
person#party,worker
person#peer,religious person
person#peer,worker
person#perceiver,preserver
person#religious person,unwelcome person
person#traveler,unwelcome person
person#traveler,worker
person#unfortunate,unwelcome person
person#unskilled person,worker
phenomenon#consequence,natural phenomenon
placental#carnivore,primate
plant#air plant,vascular plant
plant#houseplant,vascular plant
plant#poisonous plant,vascular plant
plant#vascular plant,wilding
position#angular position,placement
possession#assets,liabilities
possession#assets,transferred property
possession#liabilities,transferred property
possession#property,transferred property
process#decrease,natural process
process#development,organic process
process#human process,natural process
process#human process,organic process
process#increase,organic process
process#natural process,organic process
process#natural process,phenomenon
process#organic process,phenomenon
process#organic process,processing
product#book,work
property#age,temporal property
property#bodily property,magnitude
property#bodily property,spatial property
property#consistency,magnitude
property#degree,magnitude
property#degree,tactile property
property#magnitude,physical property
property#magnitude,sound property
property#magnitude,temporal property
property#magnitude,weakness
property#physical property,strength
property#physical property,weakness
quality#appearance,comprehensibility
quality#appearance,inelegance
quality#asset,power
quality#changeableness,difference
quality#changelessness,immobility
quality#characteristic,morality
quality#credibility,lawfulness
quality#elegance,morality
quality#good,worth
quality#immorality,inelegance
quality#immorality,unpleasantness
quality#inaccuracy,mobility
quality#incomprehensibility,opacity
quality#inelegance,unnaturalness
quality#morality,naivete
quality#regularity,sameness
quality#unnaturalness,worth
region#extremity,layer
region#geo-political entity,geographical area
relation#linguistic relation,part
relation#logical relation,opposition
relation#magnitude relation,position
relation#magnitude relation,possession
relation#opposition,part
relation#opposition,reciprocality
relation#ownership,possession
relation#part,possession
relation#position,possession
religious ceremony#rite,sacrament
seafood#freshwater fish,saltwater fish
shape#angular shape,line
shape#round shape,solid
shorebird#sandpiper,snipe
shrub#amorpha,subshrub
shrub#buckthorn,smoke tree
signal#indicator,symbol
skilled worker#aviator,sailor
skilled worker#aviator,serviceman
skilled worker#sailor,serviceman
snake#colubrid snake,viper
snake#elapid,viper
social dancing#ballroom dancing,folk dancing
social event#contest,show
social group#gathering,organization
social group#gathering,set

social group#movement,organization
social group#organization,organized crime
social group#organization,political system
social group#organization,set
social group#organized crime,set
sound#cry,noise
speech act#address,informing
speech act#challenge,disagreement
speech act#denial,rejection
speech act#disclosure,informing
speech act#informing,request
spiny-finned fish#percoid fish,plectognath
state#cognitive state,feeling
state#cognitive state,relationship
state#condition,condition
state#condition,disorder
state#condition,feeling
state#condition,imperfection
state#condition,order
state#condition,physiological state
state#condition,skillfulness
state#condition,status
state#death,inaction
state#death,physiological state
state#disorder,feeling
state#disorder,physiological state
state#feeling,imperfection
state#feeling,order
state#feeling,physiological state
state#feeling,relationship
state#feeling,separation
state#feeling,situation
state#feeling,status
state#illumination,status
state#inaction,physiological state
state#physiological state,temporary state
statement#declaration,pleading
structure#area,balcony
structure#area,establishment
structure#area,porch
structure#area,shelter
structure#cavity,passage
substance#body substance,chemical element
substance#body substance,fluid
substance#body substance,food
substance#body substance,material
substance#chemical element,compound
substance#chemical element,material
substance#chemical element,mixture
substance#compound,element
substance#compound,food
substance#compound,material
substance#compound,mixture
substance#compound,solid
substance#food,material
substance#food,mixture
substance#food,solid
substance#material,mixture
substance#material,solid
termination#destruction,killing
time period#calendar day,time off
time period#decade,time of life
time period#era,time of life
time period#time,work time
trait#character,drive
trait#demeanor,nature
trait#demeanor,pride
trait#indiscipline,stinginess
transgression#crime,evil
travel#air travel,journey
travel#journey,walk
tree#acacia,bottle-tree
tree#angiospermous tree,bottle-tree
tree#ash,gum tree
unit of measurement#explosive unit,mass unit
unit of measurement#mass unit,metric unit
unit of measurement#mass unit,weight unit
unit of measurement#volume unit,weight unit
unit#military unit,team
vascular plant#aquatic plant,herb
vascular plant#aquatic plant,woody plant
vascular plant#bulbous plant,herb
vascular plant#bulbous plant,woody plant
vascular plant#cormous plant,herb
vascular plant#desert plant,woody plant
vascular plant#herb,pteridophyte
vascular plant#herb,spermatophyte
vascular plant#herb,vine
vascular plant#herb,weed
vascular plant#herb,woody plant
vascular plant#pteridophyte,spermatophyte
vascular plant#pteridophyte,woody plant
vascular plant#spermatophyte,vine
vascular plant#spermatophyte,weed
vascular plant#spermatophyte,woody plant
vascular plant#vine,woody plant
vascular plant#weed,woody plant
vehicle#craft,military vehicle
vertebrate#bird,mammal
vessel#boat,sailing vessel
visual property#color,color property
volume unit#dry unit,liquid unit
way#passage,road
wheeled vehicle#car,horse-drawn vehicle
wheeled vehicle#self-propelled vehicle,wagon
wood#cedar,cypress
woody plant#arborescent plant,shrub
woody plant#shrub,tree
work#labor,undertaking
worker#assistant,skilled worker
worker#employee,skilled worker
writing#literary composition,matter
writing#matter,section
written communication#writing,writing
wrongdoing#falsification,transgression

basic cognitive process#discrimination,perception
bulbous plant#iridaceous plant,liliaceous plant
change#change of integrity,change of magnitude
commissioned officer#commissioned military officer,commissioned naval officer
communication#auditory communication,visual communication
communication#auditory communication,written communication
communication#indication,written communication
communication#visual communication,written communication
condition#pathological state,psychological state
creation#creating by mental acts,creating from raw materials
indefinite quantity#containerful,large indefinite quantity
indefinite quantity#containerful,small indefinite quantity
liquid unit#british capacity unit,united states liquid unit
measure#definite quantity,fundamental quantity
measure#definite quantity,system of measurement
measure#fundamental quantity,indefinite quantity
mechanism#mechanical device,rotating mechanism
monetary unit#czech monetary unit,slovakian monetary unit
monetary unit#moldovan monetary unit,romanian monetary unit
monetary unit#north korean monetary unit,south korean monetary unit
natural phenomenon#chemical phenomenon,organic phenomenon
natural phenomenon#geological phenomenon,physical phenomenon
process#basic cognitive process,higher cognitive process
relation#magnitude relation,mathematical relation
teleost fish#soft-finned fish,spiny-finned fish
unit of measurement#electromagnetic unit,temperature unit
wheeled vehicle#horse-drawn vehicle,self-propelled vehicle

Appendix B

Metaphoric Polysemy Patterns

act#activity,assumption

activity#diversion,use

arrangement#formation,ordering

artifact#building material,commodity

artifact#excavation,way

attribute#property,quality

attribute#property,state

attribute#property,trait

cognition#ability,content

cognition#ability,process

communication#auditory communication,expressive style

communication#auditory communication,message

communication#expressive style,message

communication#expressive style,visual communication

communication#expressive style,written communication

communication#indication,message

communication#signal,visual communication

device#conductor,support

device#holding device,mechanism

extremity#boundary,extreme point

geological formation#natural elevation,slope

group#arrangement,biological group

group#biological group,collection

group#collection,social group

happening#accident,change

information#evidence,stimulation

leader#employer,superior

measure#indefinite quantity,linear measure

measure#playing period,time interval

organism#animal,person

organism#mutant,person

organization#musical organization,unit

person#adult,bad person

person#adult,capitalist

person#adult,combatant

person#adult,domestic partner

person#adult,entertainer

person#adult,expert

person#adult,intellectual

person#adult,leader

person#adult,male

person#adult,occultist

person#adult,preserver

person#adult,relative

person#adult,worker

person#adventurer,unwelcome person

person#adventurer,worker

person#advocate,communicator

person#advocate,follower

person#advocate,religious person

person#advocate,user

person#bad person,combatant

person#bad person,peer

person#bad person,primitive

person#bad person,unfortunate

person#capitalist,communicator

person#capitalist,contestant

person#capitalist,entertainer

person#capitalist,leader

person#capitalist,money handler

person#capitalist,worker

person#combatant,commoner

person#combatant,large person

person#combatant,worker

person#communicator,leader

person#communicator,ruler

person#communicator,traveler

person#communicator,worker

person#contestant,engineer

person#contestant,entertainer

person#contestant,expert

person#contestant,gambler

person#contestant,leader

person#contestant,nonworker

person#contestant,unskilled person

person#contestant,unwelcome person

person#contestant,worker

person#creator,expert

person#creator,leader

person#creator,traveler

person#creator,worker

person#dissenter,inhabitant

person#domestic partner,worker

person#entertainer,simpleton

person#expert,worker

person#explorer,worker

person#follower,user

person#follower,worker

person#friend,leader

person#friend,peer

person#friend,relative
person#friend,worker
person#inhabitant,traveler
person#intellectual,leader
person#juvenile,relative
person#leader,male
person#leader,planner
person#leader,worker
person#lover,male
person#male,worker
person#perceiver,signer
person#planner,worker
person#preserver,worker
person#religious person,traveler
person#traveler,unskilled person
property#bodily property,physical property
property#magnitude,visual property
property#physical property,visual property
psychological feature#event,motivation
quality#clearness,comprehensibility
region#area,geographical area
region#extremity,top
social group#kin,organized crime
state#cognitive state,condition
state#condition,illumination
state#feeling,illumination
trait#character,nature
unit#administrative unit,military unit
whole#artifact,item
writing#editing,section
person#inhabitant,native
person#inhabitant,unwelcome person
person#intellectual,worker
person#juvenile,worker
person#leader,personification
person#leader,relative
person#linguist,literate
person#male,peer
person#peer,relative
person#perceiver,worker
person#preserver,unwelcome person
person#relative,unwelcome person
person#traveler,unfortunate
person#user,worker
property#bodily property,visual property
property#physical property,temporal property
psychological feature#cognition,motivation
quality#appearance,power
quality#morality,worth
region#area,public square
social group#kin,organization
speech act#command,request
state#cognitive state,temporary state
state#condition,situation
state#feeling,temporary state
unit of measurement#mass unit,monetary unit
unit#administrative unit,team
writing#document,matter

Appendix C

Homonymy Polysemy Patterns

act#action,inactivity
act#activity,hindrance
activity#acting,work
activity#concealment,work
activity#diversion,turn
animal#chordate,female
animal#chordate,invertebrate
aquatic bird#wading bird,waterfowl
artifact#article,covering
artifact#article,sheet
artifact#block,facility
artifact#block,instrumentality
artifact#block,structure
artifact#commodity,facility
artifact#commodity,line
artifact#commodity,plaything
artifact#commodity,surface
artifact#commodity,track
artifact#commodity,way
artifact#covering,enclosure
artifact#covering,line
artifact#covering,padding
artifact#covering,plaything
artifact#covering,surface
artifact#creation,plaything
artifact#creation,strip
artifact#creation,surface
artifact#decoration,surface
artifact#enclosure,surface
artifact#excavation,facility
artifact#excavation,structure
artifact#facility,surface
artifact#float,instrumentality
artifact#instrumentality,sheet
artifact#instrumentality,surface
artifact#instrumentality,weight
artifact#padding,sheet
artifact#padding,surface
artifact#strip,structure
attribute#shape,trait
bodily process#consumption,reaction
change#change of direction,motion
change#change of magnitude,motion
change#change of state,satisfaction
cognition#cognitive factor,process
communication#auditory communication,indication
communication#display,message
communication#expressive style,signal
communication#message,message
communication#message,sign
communication#sign,written communication
communicator#announcer,articulator
container#vessel,wheeled vehicle
covering#coating,protective covering
definite quantity#absolute value,number
device#dental appliance,support
device#electrical device,restraint
device#instrument,musical instrument
device#machine,memory device
device#machine,support
device#musical instrument,support
device#restraint,support
device#strengthener,support
event#group action,social event
event#happening,social event
extremity#boundary,end
facility#correctional institution,housing
food#beverage,nutriment
gathering#assembly,body
genus#fish genus,monocot genus
group#biological group,social group
group#collection,people
group#social group,subgroup
happening#beginning,discharge
happening#beginning,movement
happening#change,ending
happening#discharge,fire
happening#ending,failure
horizontal surface#paved surface,platform
implement#rod,stick
implement#sports implement,stick
instrumentality#ceramic,device
instrumentality#connection,implement
instrumentality#container,equipment
instrumentality#container,weaponry
instrumentality#conveyance,equipment
instrumentality#conveyance,implement
instrumentality#device,furnishing
instrumentality#device,medium
instrumentality#equipment,furnishing

instrumentality#equipment,system
instrumentality#furnishing,implement
measure#definite quantity,playing period
message#commitment,information
organism#animal,plant
organism#nonvascular organism,plant
person#acquirer,adult
person#acquirer,contestant
person#adult,contestant
person#adult,religious person
person#african,inhabitant
person#applicant,bad person
person#bad person,communicator
person#black,male
person#capitalist,good person
person#capitalist,traveler
person#combatant,leader
person#communicator,gambler
person#contestant,enrollee
person#contestant,preserver
person#disputant,worker
person#engineer,unskilled person
person#entertainer,peer
person#fiduciary,worker
person#gambler,leader
person#good person,user
person#intellectual,user
person#nonworker,traveler
person#relative,religious person
person#unfortunate,worker
placental#carnivore,ungulate
property#degree,physical property
property#sound property,visual property
quality#appearance,characteristic
relation#linguistic relation,logical relation
science#linguistics,natural science
side#rear,reverse
spiritual being#deity,spirit
state#condition,inaction
state#condition,relationship
substance#body substance,protoplasm
substance#element,material
trait#nature,stinginess
unit of measurement#force unit,monetary unit
vertebrate#aquatic vertebrate,bird
vertebrate#aquatic vertebrate,reptile
worker#assistant,employee
instrumentality#implement,toiletry
message#approval,information
message#offer,proposal
organism#individual,person
organism#person,plant
person#acquirer,communicator
person#adjudicator,contestant
person#adult,inhabitant
person#adult,user
person#amerindian,bad person
person#authority,capitalist
person#bad person,contestant
person#capitalist,enrollee
person#capitalist,preserver
person#capitalist,unfortunate
person#communicator,contestant
person#communicator,good person
person#contestant,party
person#creator,literate
person#drug user,traveler
person#enrollee,worker
person#fiduciary,leader
person#friend,religious person
person#good person,slave
person#homosexual,leader
person#leader,traveler
person#owner,worker
person#scientist,worker
person#unwelcome person,worker
plant#fungus,vascular plant
property#degree,sound property
property#spatial property,visual property
quality#asset,worth
relation#part,position
science#mathematics,natural science
social group#gathering,kin
state#cognitive state,illumination
state#condition,integrity
substance#body substance,compound
substance#chemical element,solid
time period#calendar day,work time
unit of measurement#computer memory unit,metric unit
unit of measurement#monetary unit,weight unit
vertebrate#aquatic vertebrate,mammal
vertebrate#mammal,reptile