An Automatically Generated Lexical Knowledge Base with Soft Definitions

by

Martin Scaiano

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

There is a need for methods that understand and represent the meaning of text for use in Artificial Intelligence (AI). This thesis demonstrates a method to automatically extract a lexical knowledge base from dictionaries for the purpose of improving machine reading. Machine reading refers to a process by which a computer processes natural language text into a representation that supports inference or inter-connection with existing knowledge (Clark and Harrison, 2010).[1]

There are a number of linguistic ideas associated with representing and applying the meaning of words which are unaddressed in current knowledge representations. This work draws heavily from the linguistic theory of *frame semantics* (Fillmore, 1976). A word is not a strictly defined construct; instead, it evokes our knowledge and experiences, and this information is adapted to a given context by human intelligence. This can often be seen in dictionaries, as a word may have many senses, but some are only subtle variations of the same theme or core idea. Further unaddressed issue is that sentences may have multiple reasonable and valid interpretations (or readings).

This thesis postulates that there must be algorithms that work with symbolic representations which can model how words evoke knowledge and then contextualize that knowledge. I attempt to answer this previously unaddressed question, "How can a symbolic representation support multiple interpretations, evoked knowledge, soft word senses, and adaptation of meaning?" Furthermore, I implement and evaluate the proposed solution.

This thesis proposes the use of a knowledge representation called Multiple Interpretation Graphs (MIGs), and a lexical knowledge structure called auto-frames to support contextualization. MIG is used to store a single auto-frame, the representation of a sentence, or an entire text. MIGs and auto-frames are produced from dependency parse trees using an algorithm I call connection search. MIG supports representing multiple different interpretations of a text, while auto-frames combine multiple word senses and information related to the word into one representation. Connection search contextualizes MIGs and auto-frames, and reduces the number of interpretations that are considered valid.

In this thesis, as proof of concept and evaluation, I extracted auto-frames from *Longman Dictionary of Contemporary English* (*LDOCE*). I take the point of view that a word's meaning depends on what it is connected to in its definition. I do not use a

[1]The term *machine reading* was coined by Etzioni et al. (2006).

predetermined set of semantic roles; instead, auto-frames focus on the connections or mappings between a word's context and its definitions.

Once I have extracted the auto-frames, I demonstrate how they may be contextualized. I then apply the lexical knowledge base to reading comprehension. The results show that this approach can produce good precision on this task, although more research and refinement is needed. The knowledge base and source code is made available to the community at `http://martin.scaiano.com/Auto-frames.html` or by contacting martin@scaiano.com.

Acknowledgements

I wish to thank my wife for supporting me. She has been patient, encouraging and a helper through this long process. She has participated in more discussions about my thesis topic and ideas than anyone should have to experience. I love you, Dale. I cannot express enough thanks.

Thanks to my parents and family for supporting me and encouraging me. My father's revisions and insight about writing papers has been invaluable. Thanks to my mother for caring about my well-being and looking out for me.

I wish to acknowledge Diana Inkpen, my supervisor, for guiding me. She asked the tough questions about evaluation and what people expect from a thesis. Diana was a friend when times were difficult.

Thanks to Stan Szpakowicz for both his time and frankness. His investment and feedback on my writing and research has challenged me and improved my work. A second very special thanks to Stan who put in a major effort editing the thesis: removing typos, making statements clear, correcting grammar, and just generally making it better for humanity. The effort was beyond the scope of his responsibilities and is thus even more appreciated, as he committed many long hours of his time to editing.

Thanks to my committee members Ash Asudeh, Caroline Barrière, Stan Szpakowicz and René Witte. They provided insight from many domains of research. They patiently guided me to write better dissertation and molded me into a better researcher.

Thanks to Alistair Kennedy, Anna Kazantseva, and Chris Fournier for their encouragement, discussions, and assistance during this work. They brought perspective to my work when I lacked it.

Finally, thanks to everyone who has told me to "do less and focus more".

Contents

1 Introduction **1**

1.1 Motivation 1
1.2 Goal 3
1.3 Intended Contributions 4
1.4 Outline of the Thesis 5

2 Literature Review **7**

2.1 Introduction 7
2.2 Machine Reading and Lexical Resources 8
2.3 Automatically Extracted Resources 9
2.4 Automatically Extracted Lexical Resources 11
2.5 Summary 13

3 Background: Words, Dictionaries, and Knowledge Representation **14**

3.1 Dictionaries 14
3.2 Definitional Style 15
3.3 The Problem with Word Senses 18
3.4 The Problem with Dictionaries 21
3.5 Knowledge Representation 23
3.5.1 Semantic Roles 24
3.6 Semantic Networks 27
3.7 Conceptual Graphs 27
3.8 Frames 28
3.9 Web Ontology Language (OWL) 30
3.10 Packed Graphical Representation (PGR) 30
3.11 Applications of Knowledge 31
3.11.1 Textual Entailment 31

3.11.2 Question Answering . . . 33
3.11.3 Reading Comprehension . . . 34
3.12 Summary . . . 35

4 Theory: Auto-frames 36
4.1 Why Dictionaries . . . 37
4.2 Auto-frames and Semantic Roles . . . 38
4.2.1 Auto-frames as Soft Definitions . . . 41
4.3 A Knowledge Representation . . . 42
4.4 Multiple Interpretation Graphs (MIG) . . . 43
4.4.1 Introduction . . . 43
4.4.2 A Closer Look at MIG . . . 44
4.4.3 Example . . . 53
4.4.4 Summary of MIG . . . 55
4.5 Auto-frames . . . 56
4.5.1 Introduction . . . 56
4.5.2 Auto-frames and Roles (Attachment Points) . . . 62
4.5.3 Attachment Example . . . 63
4.6 Contextualization . . . 65
4.6.1 Introduction . . . 65
4.6.2 Contextualization Examples . . . 67
4.7 Comparison with FrameNet . . . 70
4.8 Summary . . . 72

5 Methodology 73
5.1 Building a Lexical Knowledge Base . . . 73
5.1.1 Parsing . . . 74
5.1.2 Initial Transformations . . . 77
5.1.3 Marking Definition Type and Genus . . . 81
5.1.4 Assigning High-Confidence Sense Labels . . . 88
5.2 Connection Search . . . 90
5.2.1 Expansion . . . 90
5.2.2 Creating Attachments . . . 92
5.2.3 Weigh and Prune . . . 97
5.2.4 Constructing Frames . . . 106
5.3 Using Auto-frames: Entailment and Reading Comprehension . . . 111

5.3.1 Entailment . . . 111
5.3.2 Reading Comprehension . . . 113
5.4 Summary . . . 115

6 System Overview **116**
6.1 Overview . . . 116
6.2 MIG . . . 118
6.3 Parser . . . 118
6.4 Post-Parsing . . . 119
6.5 Knowledge Base . . . 120
6.6 Connection search . . . 122
6.7 Creating auto-frames . . . 122
6.8 Entailment . . . 122
6.9 Remedia . . . 123
6.10 Java Virtual Machine . . . 124
6.11 Summary . . . 124

7 Evaluation **126**
7.1 Intrinsic Evaluation . . . 126
7.1.1 Evaluation: Comparing against Logic Form (LF) . . . 127
7.1.2 Genus Detection Evaluation . . . 132
7.2 Inspection of Auto-frames . . . 136
7.2.1 Bicycle . . . 136
7.2.2 Piano . . . 152
7.2.3 Airplane . . . 156
7.2.4 Bribe . . . 158
7.2.5 Bank . . . 159
7.2.6 Comparison of Lexical Units . . . 159
7.3 Extrinsic Evaluation: Reading Comprehension . . . 162
7.3.1 Data . . . 162
7.3.2 Results . . . 164
7.3.3 Review of Remedia Answers . . . 169
7.4 Conclusions . . . 172

8 Conclusions and Future Work **174**
8.1 Summary . . . 174
8.2 Contributions . . . 176
8.3 Future Work . . . 177
8.3.1 Leveraging the definitions of prepositions . . . 178
8.3.2 Connecting adjective, adverbs, and other parts of speech . . . 178
8.3.3 Implicit arguments and discourse . . . 178
8.3.4 Corpus-based enhancements . . . 179
8.3.5 Other Dictionaries . . . 179
8.3.6 Merging Dictionaries . . . 179

Glossary **180**

Acronyms **182**

Bibliography **184**

A Building a Syntactic Logic Form **200**
A.1 Logic Form (LF) . . . 200
A.2 Transformation Rules . . . 203
A.2.1 Creating Predicates . . . 204
A.2.2 Processing Multi-Word Expressions . . . 205
A.2.3 Processing Conjunctions . . . 205
A.2.4 Assigning Fixed Identifiers . . . 206
A.2.5 Noun-Noun compounds . . . 207
A.2.6 Assigning Dependent Identifiers . . . 207
A.2.7 Assigning Parameters . . . 208
A.2.8 Processing Adverbial Clause Modifiers . . . 210
A.2.9 Assigning Prepositional Relations . . . 211
A.3 Logic Form Criticisms . . . 212

B Transformations **215**
B.1 Linear Graph Notation . . . 215
B.2 Transformations . . . 216

C Knowledge Base Implementation **226**

D Graphical Search User Interface: Search UI **230**

List of Tables

3.1 Definitions with the genus and differentia marked 16
3.2 Definitions of *dog* from *LDOCE* and *WordNet* 3.0 17
3.3 Circular definitions 22
3.4 Definitions which are hard to capture in binary relations 26

4.1 Examples of definitions which define their roles or frame elements 39
4.2 Examples of definitions which define properties of other frames 40
4.3 Example Sentences for Bicycle Frame 58
4.4 Polysemous Examples of Bank 59
4.5 Senses for Bank in *LDOCE* 61
4.6 Example Autoframe for Cook 63
4.7 Examples of Filter Auto-frames 67
4.8 Example modulations of bicycle 68

5.1 Examples of Difficult definitions to Parse 75
5.2 Rules for linking clauses 81
5.3 Example of Definitions Requiring Clausal Links 82
5.4 Sample Definitions Demonstrating Different Styles 84
5.5 Interpretation of Genus Term by Definition Features 85
5.6 Examples of Typical Genus Terms 87
5.7 Examples of Definitions with Context Patterns 87
5.8 Definitions with Bi-Directional Relationships 89
5.9 Senses Assigned to Named Entity Types 93
5.10 Example Adjective and Adverb Definitions 98
5.11 Definitions of Travel 107
5.12 Example Sentences for Bicycle Frame 109
5.13 Example definitions for *contribute, bribe* and *pay* 113

5.14 Sentences used to validate the entailment system 114

6.1 Search time of sample queries in *LDOCE* graph database 121

7.1 Comparison of syntactic representation against Senseval 3 Logic Form Identification task 129
7.2 Results on development data from Senseval 3 Logic Form Identification task 130
7.3 Explanation of Metrics for Genus Detection Evaluation 134
7.4 Evaluation of genus detection against Gold LFs in *Extended WordNet* . 134
7.5 Core Lexical Senses for bicycle Auto-frame 137
7.6 Evoked Senses for bicycle Auto-frame 138
7.7 Related Senses for bicycle Auto-frame 139
7.8 Attachment points for the *bicycle* Auto-frame 140
7.9 Non-Core FrameNet Frame Elements for the noun *bicycle* 140
7.10 Core FrameNet Frame Elements for the verb *bicycle* 141
7.11 Non-Core FrameNet Frame Elements for the verb *bicycle* 143
7.12 Example modulations of bicycle 144
7.13 Core Lexical Senses for piano Auto-frame 152
7.14 Core Evoked Senses for piano Auto-frame 152
7.15 Attachment points for the *piano* Auto-frame 153
7.16 Core Lexical Senses for airplane Auto-frame 156
7.17 Core Evoked Senses for airplane Auto-frame 157
7.18 Attachment points for the *airplane* Auto-frame 157
7.19 Core Lexical Senses for Bribe Auto-frame 158
7.20 Core Evoked Senses for bribe Auto-frame 158
7.21 Attachment points for the *bribe* Auto-frame 159
7.22 Senses for Bank in *LDOCE* 160
7.23 Comparison of Lexical Units 161
7.24 Baseline results on factoid questions from TREC 2004 Question-Answering (QA) task 164
7.25 Break down of question types and system performance on questions from grades 2 and 3 in the remedia data 167
7.26 Examples of sentences which match the queries, but do not contain the answer 168
7.27 Reading Comprehension Sample Results 172

List of Figures

3.1 Dependency parse trees of definitions with deep structures 25
3.2 Example of a simple conceptual graph (CG) 27

4.1 Example MIG: XOR Restriction 46
4.2 Example MIG: Bank expanding to two word senses 49
4.3 Example MIG: The teacher has a mustache 54
4.4 Examples of contextualized sentences 69

5.1 Example parse tree with prefix: "X is to run very fast" 76
5.2 Example parse tree with prefix: "X is a dog" 76
5.3 Dependency Parse of Conjunction 79
5.4 Post-processed Graph of Conjunction 79
5.5 Example of Multi-node Expressions 80
5.6 Example of Merged Multi-node Expressions 80
5.7 Parse Tree for *Anteater* 83
5.8 Parse Tree for *Bicycle* 83
5.9 Example of a conjunction as the genus for the definition of *close*, verb, *Extended WordNet*. 88
5.10 Example Parse Tree with Indirect Attachment 95
5.11 Example CG: The teacher has a mustache (repeated) 97
5.12 Example of Weighing and Prune MIG 101

6.1 The System's Modules 117
6.2 Parsing Model 120

7.1 Predicate-level comparison of mapping to LF 130
7.2 Argument-level comparison of mapping to LF 131

7.3 Percentage evaluation of genus detection against Gold LFs in *Extended WordNet* . . . 135
7.4 Contextualization Example 1 of "Boris oiled his bicycle" . . . 146
7.5 Contextualization Example 2 of "Boris cleaned his bicycle" . . . 147
7.6 Contextualization Example 3 of "Boris rode his bicycle" . . . 148
7.7 Contextualization Example 4 of "Boris put oil on his bicycle" . . . 149
7.8 Contextualization Example 5 of "Boris traveled someplace on his bicycle" 150
7.9 Contextualization Example 6 of "Boris bicycled to the store" . . . 151
7.10 Contextualization Example 7 of "Boris rode to the store" . . . 151
7.11 Contextualization Example of "Boris played piano" . . . 154
7.12 Contextualization Example of "Piano music is the best." . . . 155
7.13 Recall on Remedia reading comprehension and comparison of the different knowledge representations . . . 165
7.14 Impact of Multiple Interpretations on Remedia . . . 167

A.1 Dependency to LF Transformation Steps . . . 203
A.2 Visualization of the dependency parse tree from the Stanford parser . . . 204
A.3 Dependency parse tree of conjunction . . . 206
A.4 Post-processed graph of conjunction . . . 206
A.5 Example Stanford parse tree of an adjective with a subject . . . 209
A.6 Example Stanford parse tree with an adverbial modifier clause . . . 211
A.7 Collapsed Dependency Parse Tree . . . 213
A.8 Uncollapsed Dependency Parse Tree . . . 213

B.1 Example transformation . . . 217

C.1 Structure of the graphs in the database . . . 227

D.1 Search UI . . . 231

List of Algorithms

4.1 Expand MIG Operation 53
5.1 Creating Attachments 96
5.2 Weigh and Prune Concepts 105

Chapter 1

Introduction

1.1 Motivation

Many computer scientists, including myself, are attracted to the idea of classical Artificial Intelligence (AI): an AI that can reason, communicate, and answer questions, especially those that require understanding, insight, and composition of knowledge. While there are many forms of AI today and many interesting tasks that can be accomplished, classical AI remains a long-term dream.

Over the years, researchers have shown that computers can be programmed to solve complex problems that involve reasoning, composition of knowledge, planning, and other aspects of cognition. For decades, expert systems have been very successful at inferring solutions given certain facts in small domains (Compton and Jansen, 1990). A recent AI system was shown to be able to solve a first-year chemistry exam with grades comparable to those of good students (Barker et al., 2004).

Reasoning and logic systems have existed for many years. They perform well, better and faster with every passing decade. Two common difficulties with these systems are acquisition of background knowledge and transformation of text into a semantic representation.

The problem of knowledge acquisition could be solved if a computer program could read and understand text. Humans can learn by reading targeted informational materials, such as textbooks, encyclopedia, or even dictionaries. Each sentence can be full of valuable information; understanding the information as a whole is critical in modeling and extracting detailed information.

The fields of Information Extraction (IE), Information Retrieval (IR), and Question-

Answering (QA) have focused on finding information and answers from large (often redundant) corpora (Lin and Pantel, 2001; Richardson et al., 1998; Akbik and Löser, 2012; Mitchell, 2010; Chambers and Jurafsky, 2008; Wang et al., 2012; Schubert, 2002). Yet, Barker et al. (2004) required manual entry of a chemistry textbook and manual encoding of questions. It is interesting that only one text-book was required for a computer to pass the exam, not hundreds or thousand of documents, as are normally used in IE, IR, and QA.

If, as proposed, machine reading and text understanding may be a solution to the knowledge acquisition problem, then it would also be a key step toward classical AI. "*Machine reading* itself is a loosely-defined notion, ranging from extracting selective facts to constructing complex, inference-supporting representations of text" (Clark and Harrison, 2010). The majority of the field is focused on extracting selective facts, even those who focus on inference-supporting representations still tend to rarely extract knowledge. A few systems and researchers have tried to process and represent sentences and texts (Rus, 2004, 2005; Delmonte, 2004, 2013).

Machine reading is not an end task, but an intermediate task that enables Artificial Intelligence (AI) and more practical tasks. With respect to AI, reading comprehension and QA are a means of evaluating the slow progress towards smarter AI; until AI and machine reading reach levels comparable to humans, they are unlikely to be practical in these tasks. However, without research into machine reading, how can such advanced systems be developed?

A significant advancement related to machine reading was the development and adoption of Semantic Role Labeling (SRL) systems (Gildea and Jurafsky, 2002). These systems are components in a machine reading system, and a step towards deeper semantic representations needed for AI. SRL has shown improvements in IR, QA, entailment, and summarization (Pozo et al., 2004; Schlaefer et al., 2007; Wu and Fung, 2009; Palmer et al., 2010). Research into SRL has been made possible by the development of FrameNet (Baker et al., 1998) and Propbank (Babko-Malaya et al., 2004).

FrameNet and Propbank effectively enumerate senses (frames and predicates, respectively), and note which semantic roles are associated with each sense. These resources and their annotated corpora are manually built. This requires extensive time and effort to build and update, furthermore, they provide only limited coverage of the language. Currently, *FrameNet* has about 10000 lexical units (words) and only about 1000 frames (senses).

An automatically built lexical resource could provide more coverage and could be

more easily updated. The purpose of this thesis is to automatically extract a lexical knowledge base for machine reading. This work is broad and exploratory; the literature review covered many topics before a theoretical solution was chosen. Implementing a proof of concept required many iterations and extensive skill and effort to complete. The result is a proof of concept, which attempts to implement previously unaddressed linguistic ideas about the meaning of words and how they are understood.

1.2 Goal

While starting from the goal of knowledge extraction, this thesis focuses on problems associated with machine reading. The first difficulties associated with machine reading are related to lexical knowledge bases, word senses, and their representations. To improve the state of the art, I attempt to address a number of issues identified by other researchers (see chapters 2 and 3), specifically related to words evoking knowledge and then the contextualization of that knowledge. Chapter 3 introduces these problems in detail and Chapter 4 proposes a theoretical solution.

To advance the state of the art in machine reading, I draw mainly on ideas proposed by (Fillmore, 1976) in his theory of frame semantics, although Fillmore's work is based on prior work (Minsky, 1975b; Schank and Abelson, 1975). *FrameNet* (Baker et al., 1998) is an attempt to realize this theory; however, it has limitations that this work will attempt to address.

FrameNet lacks coverage of the language; the project has been active since 1997 and currently only contains 1000 frames and 10000 lexical units.

Frame semantics suggests that evoking a frame brings to mind ideas, knowledge, and expectations about the frame; however, this information is only partially captured in *FrameNet* through the use of mappings and a short textual description.

FrameNet provides an uniform representation of related concepts, which can help in comparing meanings, but it does not seem to support contextualization of frames. This is to say, if a word or lexical unit evokes a frame, then the surrounding text (clause, sentence, paragraph), which I shall refer to as context, may alter or emphasize certain expectations about the evoked frame.

Lastly, *FrameNet* does not define a representation, but is an inventory or lexicon of frames and frame elements. My research defines a representation intended to support multiple interpretations (different reading with possibly different frame/word sense assignments) and to support contextualization of frames.

The main question that I am addressing can be stated as: "Is there an implementable and tractable symbolic representation and methodology that can support words evoking knowledge and the contextualization of that knowledge?"

As many researchers have noted, the acquisition of knowledge requires knowledge (Ide and Véronis, 1995; Barrière, 1997). The first kind of knowledge needed is likely lexical knowledge, which is knowledge about the meaning of words. Ideally, to develop and test machine reading, we must start with language that is simple and central to understanding language and knowledge, that is to say, words and meanings that a child or someone new to a language would learn.

Symbols and their use in automated reasoning are usually treated as having strict meaning satisfying a precise definition. This work considers that a word evokes a frame,[1] which contains definitions and information about a word-sense, though not all the information is necessarily true in any given context. A graph-based representation called Multiple Interpretation Graphs (MIGs) is used to represent this knowledge, a lexical knowledge structure called *auto-frames* is used to organize this knowledge, and an algorithm called connection search is used to apply and contextualize this knowledge. Auto-frames include information from multiple related definitions, default values, expectations about participants, and information about relevant contexts.

To assess the value and effectiveness of auto-frames and the connection search algorithm, the knowledge was applied to reading comprehension. Reading comprehension is the task of reading a document and answering questions about it; the questions focus on testing a reader's understanding of the document. Reading comprehension provides an evaluation of how well, compared to a human level, the system understands a text.

1.3 Intended Contributions

Most of the concepts mentioned in this section are introduced and explained in Chapters 2 and 3. The main contributions of this thesis are as follows:

1. A method of detecting the genus terms of dictionary definitions.
2. A method of classifying the type of a definition.
3. A proof of concept of a knowledge representation (auto-frames) which satisfies linguistic ideas previously unaddressed in a symbolic representation, such as modeling

[1] as in *frame semantics* introduced in Chapter 3

what a word evokes and contextualizing this information.

4. A representation that supports multiple interpretations allowing processing of a partially disambiguated knowledge representation.

5. An approach to building a lexical knowledge base which does not require predetermined semantic roles.

6. A demonstration of the application of these techniques to reading comprehension.

1.4 Outline of the Thesis

Chapter 2 is a broad literature review which situates this research in the field of Natural Language Processing (NLP).

Chapter 3 presents a literature review and background information required to understand the theory and implementation of this research. A major topic addressed in this chapter is that word senses are impractical to enumerate and definitions generally too strict or narrow. Furthermore, this section discusses knowledge representations and task-based application of knowledge bases.

Chapter 4 presents the theory and concepts on which this thesis is based. How will word meaning be modeled? How will word senses be inter-connected? How will this representation include information beyond a strict definition? How will contextualization be supported?

Chapter 5 describes the implementation of the theory. This includes the knowledge extraction, the knowledge representation, and the application of the knowledge to the task of reading comprehension.

Chapter 6 describes the system design and a number of the challenges related to creating such a system. The main challenge was having algorithms with high computational complexity run in reasonable times.

Chapter 7 provides a variety of evaluations of this work. Two empirical evaluations of the knowledge extraction process are presented: comparison against Logic Form (LF) (Rus, 2004) and an evaluation of genus detection. A manual review of auto-frames and comparison against *FrameNet* is included. Subsequently, a task-based demonstration is given, where auto-frames are applied to reading comprehension. The task-based results show the benefit of using multiple interpretations and demonstrate that the application of auto-frames can lead to good precision in the task of reading comprehension.

Chapter 8 summarizes the thesis, provides conclusions, and describes future work.

Chapter 2

Literature Review

2.1 Introduction

Natural Language Processing (NLP) has benefitted from the availability of knowledge resources and the tools to create semantic annotations or interpretations of text. In the series of shared evaluations started in 2004, the PASCAL Recognizing Textual Entailment (RTE) Challenges, many of the systems use symbolic representations, knowledge resources, and semantic representations: RTE1 (Dagan et al., 2005), RTE2 (Haim et al., 2006), RTE3 (Giampiccolo et al., 2007), ..., RTE7 (Bentivogli et al., 2011). More recent challenges have included contradiction detection, which seems to be a harder task than entailment detection, showing significant movement towards text understanding (Giampiccolo et al., 2009).

In the field of Question-Answering (QA), we see that the availability of Semantic Role Labeling (SRL) systems has allowed the use of predicate-argument structures (Schlaefer et al., 2007; Sun et al., 2005).

A recent letter to the editor of *Bioinformatics* suggests that QA technologies have reached the point that they are ready for use in the medical domain (Wren, 2011), a domain where there is a lot of data to sift through, and where manual searches can be overwhelming. The author cites Watson (Ferrucci, 2010; Ferrucci et al., 2010), the system which beat some of the greatest players in the Jeopardy game show, as demonstrating the current state of the art and capabilities of QA technology.

Most of these systems depend on good quality knowledge bases. The textbook used by Barker et al. (2004) took 16 person-month to encode and well educated and trained transcribers. *WordNet* (Fellbaum, 1998), *FrameNet* (Baker et al., 1998) have been in

development by linguists for decades. Most good quality knowledge bases are manually built, require extensive effort, and are costly to produce. Could these barriers to knowledge acquisition be overcome by a program that could read and understand texts? Instead of having linguists transcode knowledge, this program could read the resources experts would have referenced and then build or update its own knowledge base.

2.2 Machine Reading and Lexical Resources

Machine reading and text understanding may be a solution to the knowledge acquisition problem. The number of research projects focused on completely representing the meaning of sentences and texts is small (Rus, 2004, 2005; Delmonte, 2004, 2013).

To this end there are a number of natural language resources and tools that assist in machine reading. Resources such as *WordNet* (Fellbaum, 1998), *FrameNet* (Baker et al., 1998), *PropBank* (Babko-Malaya et al., 2004) and *VerbNet* (Schuler, 2005) can be used to advance machine reading. The tasks involved in applying these resources, Word Sense Disambiguation (WSD) (Mihalcea, 2007; Navigli, 2009) and SRL (Gildea and Jurafsky, 2002; Litkowski, 2007; Johansson and Nugues, 2007; Bejan and Hathaway, 2007), and the tools which complete these tasks are key advancements.

WordNet is well known for its ontology and semantic network, which contains word senses and glosses. *WordNet* contains many relations, such as hypernymy, homonymy, meronymy, troponymy, entailment, and mappings between different parts of speech (POS). The hypernymy relation, often called the "is-a" relation, is used extensively in research. The presence of these relations has consistently made *WordNet* a valuable resource for NLP research.

WordNet is commonly used as a thesaurus, that is, an inventory of senses for WSD, which is the task of assigning the intended sense to each word. Lesk (1986) searched neighboring words (in the context) for overlap of terms in their definitions. Lesk's algorithm has been updated (Banerjee and Pedersen, 2002) and is still used today, though sometimes as an intelligent baseline; *WordNet* glosses are treated as definitions for the purpose of applying the algorithm. Agirre and Soroa (2009) and others have developed algorithms that use the relations between the senses in *WordNet* for WSD.

The supervised WSD algorithms are consistently the best performing in shared tasks (Agirre and Soroa, 2009). Supervised systems still barely outperform the most frequent sense baseline; furthermore these algorithms depend on large amounts of manually annotated data. When these supervised WSD systems are run on new domains, their per-

formance suffers significantly, particularly if there are no domain-specific training data. Lexical knowledge-based systems seem less affected by domain shift and may perform better than supervised systems in situations with limited or no training data (Agirre et al., 2009).

Three common resources, which go beyond sense inventories and ontologies to include roles, are *FrameNet* (Baker et al., 1998), *PropBank* (Babko-Malaya et al., 2004) and *VerbNet* (Schuler, 2005). *PropBank* provides an inventory of senses (predicates) and set of roles for these predicates. *PropBank* is manually built using a corpus of annotated examples of the predicates and roles. Another similar and related resource, *VerbNet* (Schuler, 2005) only includes verbs as predicates, and has a small set of general-purpose roles. Shi and Mihalcea (2005) write: "VerbNet roles are general and valid across different classes, in PropBank they are strictly tied to a specific role set. As a consequence, VerbNet has only 20 thematic roles, while PropBank has more than 1400 roles."

FrameNet has about 10000 lexical units (words) and only about 1000 frames (senses). Each of the frame elements (roles) are tied to specific frames. *FrameNet* is a high quality, manually produced resource built from evidence in manually annotated corpora. This requires extensive time and effort to build and update, furthermore it only provides limited coverage of the language.

These resources have enabled development and application of SRL systems (Gildea and Jurafsky, 2002; Litkowski, 2007; Johansson and Nugues, 2007; Bejan and Hathaway, 2007). These systems are components in machine reading systems. SRL has shown to improve Information Retrieval (IR), QA, entailment, and summarization (Pozo et al., 2004; Schlaefer et al., 2007; Wu and Fung, 2009; Palmer et al., 2010). Mohammad et al. (2014b) annotated tweets with SRL so that information such source, target, and emotion could be identified. SRL has shown benefits in detecting plagiarism (Osman et al., 2012).

These resources and tools are dependent on manual effort, which is a slow and costly process. Most of these linguistic resources have taken decades of effort. However, this effort might be significantly reduced with automatic methods.

2.3 Automatically Extracted Resources

The Cyc Project (Lenat, 1995; Foxvog, 2010), or its public version, OpenCyc,[1] is a large knowledge base, which tries to codify everyday common-sense knowledge. The project

[1] `http://www.opencyc.org` accessed on Jan 3, 2016

started in 1984, and Cyc one of the most extensive resources available. It is continuously being expanded in a semi-supervised way using both machine learning methods and human intervention. This resource focuses on common sense information, and it has already been formalized into a predicate-like knowledge. Over time Cyc has migrated to include more and more automated extraction (Sharma and Forbus, 2013).

Many resources and methods have been published over many years about knowledge extraction (Lin and Pantel, 2001; Richardson et al., 1998). Reasoning systems require accurate (high-precision) representations and many resources are not yet accurate enough. Some extraction methods require large amounts of data to extract only a few facts (Akbik and Löser, 2012; Mitchell, 2010). Some extract probabilistic relationships, ordered events, and co-occurrence probabilities (Chambers and Jurafsky, 2008; Wang et al., 2012).

An interesting resource is ConceptNet (Havasi et al., 2007), which is a semantic network of common-sense information. ConceptNet was derived from the information in the Open Mind Common Sense (OMCS) database (Singh et al., 2002) by using shallow parsing and pattern matching. OMCS is a crowd-sourced collection of common-sense statements.

There is a growing body of work that uses Wikipedia as the corpus for knowledge extraction (Aliprandi et al., 2011; Milne et al., 2007; Liu et al., 2008) or WSD (Mihalcea, 2007). Wikipedia is often used because it is a comprehensive resource, constantly being updated, and has some structure in the form of links, sections, and summary tables.

Researchers are often interested in extracting relations between concepts or named entities; such techniques are good for building semantic networks or filling slots in a structure for a named entity. Knowledge extraction usually has a specific goal; a particular type of information it is trying to extract. Some methods continuously read information, trying to learn new relationships and concepts (Betteridge et al., 2009; Mitchell, 2010).

However, most of these methods do not construct complete or high quality linguistic knowledge bases, instead they extract factoids or relations. These low-density[2] extraction techniques are often trying to learn implied relations (not explicitly stated) (Schubert, 2002); many are so far from "reading" that they simply rely on textual patterns to identify relationships.

The problem of knowledge acquisition can be viewed as a computer's lack of ability to read and learn. Humans can learn by reading targeted informational materials, such as textbooks, encyclopedias, or even dictionaries. Each sentence can be full of valuable

[2]By low density, I am referring to fact that they infrequently extract information over a large corpus or body of information. That is to say, $extraction\_density = \frac{num\_extracted\_facts}{corpus\_size}$.

information; understanding the information as a whole is critical in modeling and extracting detailed information. Recall that Barker et al. (2004) only required encoding *one* chemistry textbook for an automated system to pass a university chemistry exam, as opposed to hundreds or thousands of documents that are normally used in information extraction and QA. A few researchers have attempted to construct high-quality, complete, and automatic linguistic resources.

2.4 Automatically Extracted Lexical Resources

Vanderwende's (1995) work on classifying noun sequences required information on the relations between several words. Using string patterns to extract relations between words in *Longman Dictionary of Contemporary English* (*LDOCE*) (Guan-yu et al., 2010), this research produced a private resource, which evolved into MindNet (Richardson et al., 1998; Vanderwende et al., 2005). MindNet provides semantic relations between words in the *LDOCE* dictionary.

Barrière (1997) extracted knowledge from children's dictionaries (also see Barrière and Popowich, 1999). Barrière placed the syntactic representations into Conceptual Graphs (CGs). By applying a series of transformations, words were disambiguated, and syntactic relations were transformed into semantic relations. Because the same representation held the syntactic and semantic information, unresolved word senses or semantic roles remained ambiguous (in syntactic form) until another transformation or heuristic could resolve them. Barrière (1997) included consideration for a number of deeper linguistic concepts such as situational memory and convert categories. In today's research arena, Formal Concept Analysis (FCA) (Bendaoud et al., 2008; Guan-yu et al., 2010) is used to achieve similar goals, to organize concepts by unseen categories based on common features.

Rus (2002) converted WordNet 1.7 glosses into a representation called *Logic Form (LF)* (described in detail in Appendix A.1).[3] Rus notes: "The logic format is preferred when it comes to reasoning and other logic manipulations in knowledge bases." His work focused on syntactic logic forms, which have significant limitations, unlike Barrière's and Vanderwende's more semantic representations.

As an example of an LF consider the sentence "*Jane gave the gorilla a kiss.*" In LF, It would be represented as

[3]The logic form used by Rus was in fact titled *Logic Form*.

```
Jane:NN(x1) give:VB(e5, x1, x3, x2) gorilla:NN(x2) kiss:NN(x3)
```

Each noun, verb, adjective, and adverb is lemmatized and turned into a predicate, which includes the part of speech. Arguments are assigned to each predicate, forming connections between predicates. The representation is described in detail in Appendix A.

One goal of the representation was to be "as close as possible to English", thus the representation is directly derived from the output of a highly accurate syntactic parser. Rus devotes a large part of his work to improving tagging and parsing. Using supervised machine learning, a number of rules for converting syntactic representation into logic forms were learned.

According to Rus, Logic Form avoids the hard problems of WSD and SRL by using a syntactic representation. Word senses are effectively assigned in the prover: whatever sense was required to derive the proof must have been the intended sense. Logic Form ignores plurals, verb tenses, auxiliary verbs, quantifiers, modal operators, comparatives, and negation. Since the representation uses syntactic relations instead of semantic relations, then semantically equivalent sentences with different syntactic realization will fail to match.

Rus's work sets itself apart by actually applying the knowledge base to a real task and making the resource publicly available in *Extended WordNet*. He applies the knowledge base to improve a QA system. The system answered 52 out of 351 questions, 35 of them correctly (Moldovan and Rus, 2001).

Extended WordNet (Mihalcea, 2001) has been updated (Rus, 2005) to include Logic Forms; each synset, which can be treated as a sense, has a Logic Form representation of the gloss and indicates the quality (*i.e.*, was it manually generated, automatically generated, or partially reviewed).

With the exception of Barrière's work, most of these automatically extracted resources used shallow representations which lack support for certain linguistic concepts. Furthermore, these resources are based on handmade patterns, albeit these patterns cover most of the resource well.

FrameNet was inspired by and built based on a rich linguistic theory but falls short in implementing too few of the concepts. In particular, the theory of *semantic frames* (see 3.8) includes ideas about semantic memory, adaption of meaning, and the process of transforming words into a meaningful interpretation. As we will see in chapter 3, the meaning of a word may be modulated; it may vary from context to context and not all details hold true under all contexts. Furthermore, these words have an array of implications that are only true in some contexts.

Rais-Ghasem and Corriveau (1998) proposed and demonstrated a solution to the problem of modulation that required an advanced lexicon and annotated contextual examples. However, such a resource would require decades of development. This still leaves the problem of producing a large automatic lexicon with an implementation of modulation to be resolved.

2.5 Summary

Many advances in NLP and Artificial Intelligence (AI) have been enabled by the availability of knowledge resources. The development of good and complete knowledge resources is challenging and usually requires manual effort. The development of machine reading may one day be a solution to this problem; however, machine reading is a difficult and unsolved problem. Most of the advances in machine reading have come from the very slow manual creation of linguistic resources.

Many of these lexicons have poor lexical coverage or lack deeper representations. Even resources such as *FrameNet* with a rich linguistic theory do not include all the ideas that its theory proposes about the meaning of a word. The next chapter will explore the relevant literature and point out the shortcomings of lexical resources and knowledge representations in more detail.

Chapter 3

Background: Words, Dictionaries, and Knowledge Representation

This chapter reviews research about dictionaries, word meaning, and knowledge representations. The information provided in this chapter is a deeper review of important background information needed to understand this thesis and the specifics of the challenge.

A number of researchers show that the meaning of a word is a soft concept that is modulated (or adapted to its context). Furthermore, words evoke ideas, which may or may not be contextually relevant and the listener/reader must filter them. Most representations and knowledge resources today do little or nothing to account for this softness. This may be more challenging for symbolic systems, which usually assume a single sense and single interpretation. This chapter starts with an examination of how dictionaries encode the meaning of words, then reviews knowledge representations, and finally explores tasks that benefit from lexical knowledge.

3.1 Dictionaries

Dictionaries are meant to help people understand the meaning and use of a word or expression. They are usually well structured, using specific denotations, often including cross-references, but the main definition is expressed in natural language,[1] which is not an ideal representation for computers. Furthermore, there are many different styles of definitions; some depend on the dictionary, while others depend on the word being

[1]Dictionaries meant for machines may use a logic-based representation.

defined.

Dictionaries provide *senses* for each word or expression. A *sense* is a distinct[2] meaning; sometimes the distinctions between senses are subtle; sometimes they are completely different. Section 3.3 contains a detailed discussion on word senses. Each *sense* is given a *definition* or gloss, which describes or summarizes the *meaning*. It should be noted that a *definition* attempts to explain the *meaning* of a word in a concise way, but it does not include an exhaustive list of properties, features, uses, etc. A word denotes something or some idea, but a definition usually does not completely describe the thing or idea.

3.2 Definitional Style

In this section, we examine a number of ways in which definitions may differ in style. Each style may require unique detection, processing, and representation. Table 3.1 has definitions from *Longman Dictionary of Contemporary English* (*LDOCE*) and *WordNet*, with different styles, which we discuss in the following paragraphs.

Some definitions use a style where certain words are left out (see examples 3 and 4) and when applied to a context, these words should be added. In Table 3.1, the words with "<>" are placeholders for words intentionally left out of the definition. When understanding a definition in context, or reasoning about it, the placeholders should be substituted with the appropriate word. This style of definition will be referred to as *substitutional*. Substitutional definitions use a simple superficial modification applied to some other definitional style. They are typically used when defining a verb, although on occasion adverbs or adjectives may not include the word they modify.

The most common structure for a definition involves the use of a genus and a differentia. The genus is a general class (or superclass) of the word, usually these genus terms are from the same part of speech as the word being defined. The *differentia* describes how this subtype is different from the general class, or other subtypes of the general class, with respect to its properties, behavior, uses, or appearance. The examples in Table 3.1 have been coloured to indicate the genus in red underline and the differentia in *blue italics*.

Some dictionaries present multiple definitions for similar meanings, which have only slight differences in minor aspects. Some dictionaries provide different definitions for

[2]Section 3.3 discusses the differences in the meanings of words, in particular as they relate to homographs, polysemy, polylexy, etc. Ultimately, a dictionary's creators determine how to test for distinctness, that is when words require separate senses and definitions.

Example	Source	Definition
1	Dog *LDOCE*, noun_1	a common animal *with four legs, fur, and a tail*. Dogs are kept as pets or trained to guard places, find drugs
2	Bank *LDOCE*, noun_1	a business *that keeps and lends money and provides other financial services*
3	Bank *WordNet*, verb_6	<subject> put <object> *in a bank account*
4	Bank *WordNet*, verb_3	<subject> do business *with a bank* or keep *an account at a bank*

Table 3.1: Definitions with the genus and differentia marked

transitive (mono- or di-) and intransitive verbs, which differ in the addition of an argument, a direct object or an indirect object. *Over-specified* definitions describe subtle differences in meaning, use, or context of the word. *Over-specification* can make it harder to determine the intended definition of a word in context, because there are so many very closely related meanings. Over-specification tries to provide definitions for many specific contexts. Kilgarriff (1992), using *LDOCE*, claims that frequently a word in context does not have an appropriate sense listed in the dictionary; there may be no appropriate sense, or more than one.

Another style of definition is *under-specification*, in which fewer definitions are used, but each definition gives either a general description (a truth that would be present in each of the over-specified senses) or a more detailed description where the reader is expected to reason about its use in specific contexts. Often, this style of definition comes in one or more complete sentences.

An example of under-specification found in *LDOCE* is the definition of *basketball* shown below.

> *A game played indoors between two teams of five players, in which each team tries to win points by throwing a ball through a net, or the ball used in this game.*
> ($\text{basketball}_{\text{noun}}$, *LDOCE*)

The target audience motivates the content and style of the definitions. *LDOCE*, often used by students learning English as a second language, provides simple definitions usually in complete sentences. Some dictionaries intended for native English speaking

Example	Definition
1	a common animal with four legs, fur, and a tail. Dogs are kept as pets or trained to guard places, find drugs (*LDOCE*)
2	a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds (*WordNet*)

Table 3.2: Defintion of *dog* from *LDOCE* and *WordNet* 3.0, respectively

adults, tend to provide definitions that might be unreadable by a child or someone learning English. *WordNet* (Fellbaum, 1998), which is not a dictionary but a thesaurus and a semantic network, is often used in Natural Language Processing (NLP) because it provides a good inventory of word senses (implemented in synsets), relationships between senses, and glosses for senses. Glosses are similar to definitions but written for someone who understands the language. Consider Table 3.2, which contains the definition of "dog" from *LDOCE*, example 1, and the definition of "dog" from WordNet, example 2.[3] *LDOCE* describes the appearance of the animal and some familiar uses of a dog, while *WordNet* describes the animal's biological classification, which gives a better sense of related animals and some historic information, but no visual description or function is given.

Most commonly, definitions are *intensional*, that is to say they define the common properties and qualities of the things the word denotes, such as defining a dog as "a common animal with four legs, fur, and a tail". *Extensional* definitions list the things that the word refers to. Consider the definition of *color* in *LDOCE*, "red, blue, yellow, green, brown, purple etc"; it simply lists a small set of colors. Even though intensional definitions provide a set of properties for a word, they usually are not enough to uniquely classify only the things that it represents (something in its extension). For example, a cat could satisfy our excerpt from *LDOCE* for the definition of dog, because a cat is an animal with four legs, a tail and fur, and is often kept as pet.

Dictionaries use a mix of definitional styles; some definitions are incomplete sentences: a noun phrase is used to define a noun; clauses without head nouns are used to define verbs (*i.e.*, they lack a subject). Others are complete sentences defining the word, while sometimes additional sentences or clauses are used to define the appropriate context

[3]The word *genus* in example 2 refers to the animal's biological classification in a taxonomy of living things.

of the word. Consider a definition of *run* from *LDOCE*, which provides context, "if a machine or engine runs, it operates"; the subject of the verb must be a machine or engine for this sense.

The different styles of definitions in dictionaries make the conversion into a formal knowledge representation non-trivial. Special detection and processing will be needed for each style, along with a way to represent it.

3.3 The Problem with Word Senses

Kilgarriff (1997) explains that there is no universal set of word senses that work in all circumstances. Starting with different corpora or different goals will lead to different word senses. The information in any dictionary is prepared for a particular audience and purpose. Dictionary content may not be appropriate for any NLP task.

Cruse (1995) makes a distinction between polysemy and polylexy. Polysemy refers to the idea that a word has distinct senses, such as the difference between *run* as in move and *run* as in operate. Polylexy refers to senses requiring distinct lexical entries. Some words may be polysemous but not polylexic, when the meanings can be generated or inferred from a single definition. Consider a *bank* as a financial institution; some dictionaries also include *bank* as the building in which a financial institution is housed. Perhaps one of the two senses of bank can be derived from the other? Cruse proposes a different structure from traditional dictionaries; senses would indicate how they are related to other senses; furthermore, they should indicate where there is shared meaning and where the meanings differ. Instead of defining word senses independently of one another, closely related senses should be connected; in fact, perhaps, they should not even be considered distinct senses, but a *modulation* of the meaning.

Modulation is the subtle adaptation of or emphasis on a sense to produce a similar meaning that better fits a context. The meaning of the words, phrases, and sentences surrounding a frame – or, more specifically, our understanding of those elements – are its context.

Consider the sentences *Frank went to a restaurant* and *Frank went near a restaurant.* Both describe Frank traveling, both involve a restaurant, but *Frank went to a restaurant* suggests he went to eat, although this is not explicitly stated. Each sentence is a slightly different context and this helps a reader understand the differences between the two scenarios.

Additionally, when understanding a sentence, we consider more than the just the def-

initions of the words, we consider our knowledge and experience. *Going to a restaurant* suggests an activity not explicitly stated, that is ordering food and eating a meal. However, this suggestion is only appropriate in some contexts. We will refer to the meaning of a word, which includes experience and knowledge, as a frame.[4] Frames will be discussed in more detail in Section 3.8.

The frame for *restaurant* helps us understand what may be expected when working or eating at a restaurant. Context also includes our understanding of the concepts involved, such as named entities. For example, knowing that *Frank* worked at a restaurant might change what *going to a restaurant* suggests. Or consider the phrase *Captain Jack*; knowledge about the individual or surrounding text could indicate whether *captain* refers to a military rank, or that he is pilot or ship captain. Ideally, modulated senses can be inferred or generated from a core sense given a slightly different context. For example, *bank* as a financial institutions could be modulated to mean *bank*, the location or building from which a financial institution operates.

Consider the following examples (Kilgarriff, 1997). *Bank* has two distinct senses with no overlap in meaning, while the *bike* examples are modulations of a single concept, a bicycle or motorcycle; the modulations refer to the bike's mechanical parts, its surfaces, and its motion. Modulation is a kind of operation on a larger meaning (the frame), which produces a specialized meaning for the context.

Have you put the money in the bank?
The rabbit climbed up the bank.
He doesn't often oil his bike.
I dried off the bike.
Boris's bike goes like the wind.

For another example, consider the company *Google*, whose original product was a search engine. Over time *google* became a verb, which meant to search using Google - a form of modulation. The term is still occasionally modulated to just mean search the Internet regardless of which search engine.

A dictionary that considers modulation may not need to completely enumerate all polysemous senses. The context of a word may indicate a coarse sense, but that sense may

[4]When using the term *frame*, I am specifically referring to semantic frames. A semantic frame is "a script-like structure of inferences that characterize a type of situation, object, or event" (Johnson et al., 2002). The usage of the *frame* in this thesis may differ slightly from the readers' expectations by including the idea of experiences.

need to be modulated. A coarse-grain sense is like an under-specified sense with minimal or no overlap with other senses. Coarse-grain senses have been shown to improve results on many NLP tasks (Palmer et al., 2006). Modulation adjusts for subtle differences in the meaning, similar to selecting or generating an over-specified sense.

Rais-Ghasem and Corriveau (1998) thoroughly describes the issue of modulation in language and the cognitive research relating to it. A word activates many ideas in the listener's mind, but context emphasizes particular details of that concept. Rais-Ghasem and Corriveau describe two primary forms of modulation, which they model with a multi-tier representation. A word in context is assigned a sense, that sense is modulated to a *sense-concept* and then further modulated to a *sense-view*.

The *sense-concept* represents the sense at functional level for comprehension and classification. The sense-concepts for the examples of *bike* (above) are *machine, object, vehicle*,[5] respectively.

The *sense-view* represents the sense with properties and relations. At this level, modulation may assign, access, or emphasize properties of the concept, or refer to particular relations of that concept. This is easily observed in the sentence:

> *The teacher had a mustache.*

The concept *teacher* does not specify a gender, though *mustache* implies a man, as the definition of *mustache* restricts the person to a male. In this context, *teacher* is modulated as a male. This modulation of properties, the *sense-view*, will be further discussed in later sections.

This example presents an interesting problem, as *mustache* is sometimes applied to women. Certainly, arguments can be made that the definition is wrong or the use of the word is wrong. Instead, let us accept that words are applied in contexts that change them, and NLP systems must support and adapt to their changing meanings.

Rais-Ghasem and Corriveau (1998) believed that enumeration of senses is not practical or feasible and machine-readable lexicons should support modulation. The solution they proposed requires both an advanced lexicon and annotated contextual examples; they proposed using a lexicon which contained both *sense-concepts* and *sense-views* and an annotated corpus of examples so that the correct assignment of concepts and views could be learned.

[5] These are not definitive sense-concepts, but how I have chosen to illustrate the distinctions.

3.4 The Problem with Dictionaries

During the 1990s, much research was done on converting Machine Readable Dictionaries (MRDs) to knowledge bases for computers. Ide and Véronis (1995)[6] noted that "MRDs failed to live up to early expectations that they would provide a source of ready-made, comprehensive lexical knowledge." While they do not believe that dictionaries "are conclusively unsuitable as a source for automatically building knowledge bases", they do suggest that the effort is greater than expected and a number of barriers face researchers. They point out a number of difficulties with dictionaries and their conversion to knowledge bases.

Dictionaries define words in terms of words, thus there is a somewhat circular requirement: we must understand words to learn about words. In practice this is mitigated by the fact that dictionaries often have an expectation that the reader/user will already have some minimal understanding of language; thus dictionaries may attempt to define words in terms of simpler words, or, more specifically words which the reader is expected to know. Ide and Véronis (1995) refers to this as the bootstrapping problem, while Barrière (1997) calls this the *knowledge spiral*, which is the term used here. The knowledge spiral refers to the problem that an understanding of language is a requirement for expanding an understanding of language (*i.e.*, learning new words and information). A system must first know a few hundred or thousand central terms, before it can iteratively learn more terms and knowledge. Barrière (1997) attempts to start near the beginning of the knowledge spiral by processing a children's dictionary, while others have used *LDOCE* (Vanderwende, 1995), because of its controlled vocabulary.[7] While the knowledge spiral poses a real problem, Barrière (1997) and Rus (2002) have shown that it is possible to learn a large vocabulary by using iterative techniques and heuristics.

As Ide and Véronis point out, a dictionary is not a complete source of knowledge from which to build a knowledge base. The information in a dictionary is incomplete due to space, readability, and stylistic restrictions. In particular, the definitions are not exhaustive lists of features and they do not provide contrast to all similar or related definitions. Dictionaries require, at a minimum, a vocabulary central in the knowledge spiral; *LDOCE*'s controlled vocabulary might be considered near the beginning of the knowledge spiral, as most other words can be defined using them. Once a system has a good vocabulary, it may attempt to read either more dictionaries for further insight

[6] While this reference is quite dated, it provides some significant criticism that is still relevant today.

[7] *LDOCE* is said to have a controlled vocabulary because it defines words using about 2000 common words.

Word	Definition
whole (2)	an assemblage of **parts** that is regarded as a single entity
part(2)	something less than the **whole** of a human artifact
part (1) component (2)	something determined in relation to something that **includes** it
include (1)	have as a **part**, be made up out of
include (2)	consider as **part** of something
whole (2)	all of something **including** all its **component** elements or **parts**

Table 3.3: Circular definitions relating to *whole*, *part*, and *include* from *WordNet* 3.0

into concepts, or even tackle an encyclopedia. Encyclopedias are above and beyond the scope of this work.

Most dictionaries were not designed as ontologies or taxonomies of concepts. However, *WordNet* is well known for its ontology and semantic network, while not a dictionary, it does contain synsets, which are similar to senses, and glosses, which are similar to definitions. *WordNet* has been beneficial in numerous tasks, algorithms, and NLP research; it has also seen many criticisms (Borin et al., 2013; Richens, 2008; Clark et al., 2006; Lenat et al., 1995) and refinements over the years.

Some definitions in dictionaries may be circular. The definition of term X is circular, if it uses term Y in its definition and term Y uses X in its definition. Thus, to understand the first definition requires understanding the second, and vice versa. While this is not always problematic, it becomes more difficult when the genus terms refer to one another. Ide and Véronis give an example of *tool* being defined as *implement* and *implement* as a *tool*. What would be an appropriate genus concept for both? Where would they fit in a taxonomy?

The circular definitions do not need to form immediate loops, there could be a series of definitions using terms in similarly circular fashion. In Table 3.3, I present a number of definitions from *WordNet* related to *part*, *include*, *component*, and *whole*. These are circular definitions, because they each require an understanding of the other terms.

Amsler (1981) indicates how some circular definitions are indicative of conceptual or language primitives. Primitive words may not be broken down into simpler words, but

must be described in terms of similar words. Amsler points to how difficult it can be to define a *group*, *set*, *collection*, *unit*, or *member* without one of these similar words. In *WordNet*, these words are defined in terms of *parts* and *whole*, which, as shown in Table 3.3, have circular definitions.

In the circumstances where one dictionary has circular definitions, which are not indicative of primitives, another dictionary may not have circular definitions; the taxonomical issues presented by Ide and Véronis may be resolved by merging several dictionaries.

Normally, words have a genus, but occasionally a definition or word does not have an appropriate genus. Amsler (1981) points out that these words are better defined by what they are related to. Barrière refers to these as empty heads. Amsler gives an example of *leaf* being defined as an *"... outgrowth of a plant..."* where outgrowth would normally have been the genus, but is not really any sort of class, but a property; leaf is better described in its relation to the plant. In several dictionaries and in *WordNet*, it is common to find definitions where the genus would be *part* or *member*, but these words do not define a hypernym relationship, instead they define some other relationship.

3.5 Knowledge Representation

For centuries, philosophers such as Plato, Aristotle, Leibniz and Boole have considered how to represent knowledge and how to derive new knowledge by inference or deduction. In NLP there are two general paradigms of knowledge representation: statistical and symbolic.

Statistical representations model words as loose associations with other words. The most common forms of statistical representation are co-occurrence vectors or matrices. These representations allow for a soft modeling of meaning, different vectors and contexts can be compared using measures of similarity or relatedness. For most of these representations automated reasoning or formal logics have not been defined, or may not be possible to define. In recent years researchers have theorized about and tested compositional and logical models for tensors (Welty et al., 2010; Peñas and Hovy, 2010; Grefenstette, 2013; Cohen et al., 2013).

Symbolic representations use symbols instead of vectors and measurements to represent the meaning of word. Symbols are usually treated as discrete, unambiguous, monotonic concepts. This treatment of symbols is generally important for automated reasoning but these require the meaning of symbols to be rigid. Consider the example of *"the teacher with the mustache"* (see section 3.3); a system should support the under-

standing that only men have mustaches. An unambiguous and monotonic representation might not then support a circumstance where the teacher is known to be female, which a statistical representations could. Furthermore, this contradiction[8] would lead most reasoning systems to a failure.

hair that grows on a man's upper lip (LDOCE, $mustache_{noun}$)

Let us say a *soft definition* is one that allows the meaning to be slightly altered by context. The process of altering a definition for context will be called *modulation* or *contextualization*. Statistical representations tend to be soft; similarity measures can be used to measure if a context agrees with a meaning. Symbolic representations are not usually soft; the semantics of most representations tends to require discrete, unambiguous, monotonic concepts.

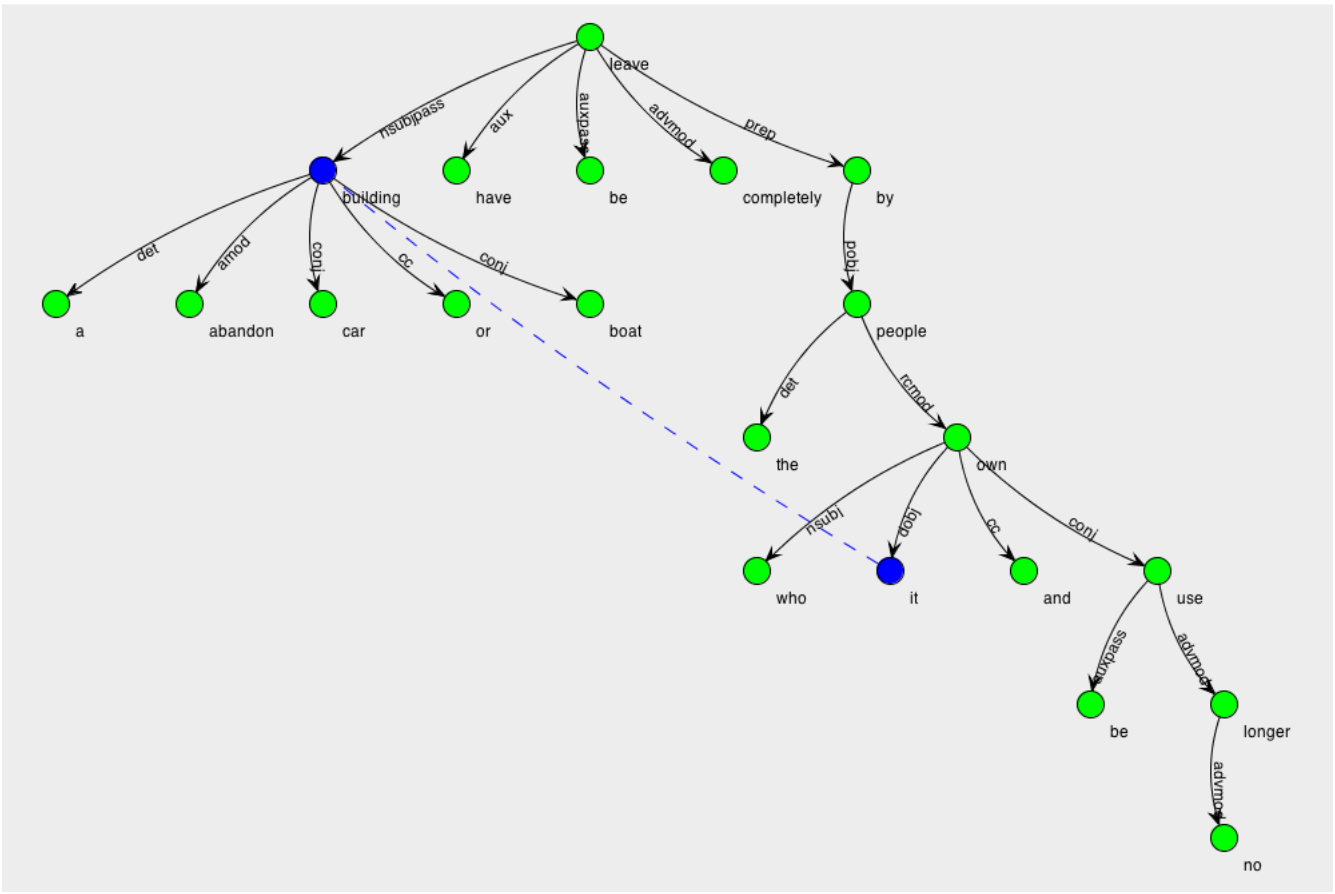
In the following sections we examine a few symbolic representations: semantic networks, Conceptual Graphs (CGs), and Web Ontology Language (OWL). These representations tend to have strict semantics, which would not support the type of modulation this thesis is attempting. Since all these representations use semantic roles, let us review this topic first.

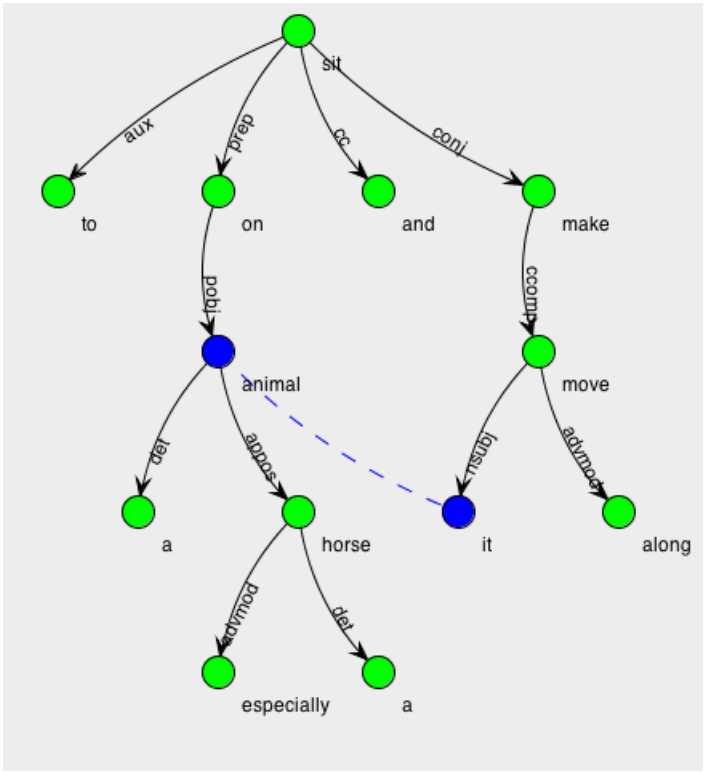
3.5.1 Semantic Roles

Much research, many technologies, and numerous resources in the past and present have focused on semantic roles and semantic networks. While these resources and technologies have been beneficial in many tasks and challenges, they may not be enough to represent the meaning of a word. Each relation can only capture a simple association between two concepts, while definitions are composed of complex predicates and syntactic structures, which cannot always be simplified into binary relations.

For example consider the following definitions for *mustache*, *ride*, or *abandoned* in Table 3.4; their dependency parse trees are show in Figure 3.1. Note that the blue dashed lines indicate coreference, as determined by the Stanford dependency parser (de Marneffe et al., 2006; de Marneffe and Manning, 2008). Each definition uses at least one verb and its argument to define the word; some of those arguments are modified prepositional phrases. Furthermore, *ride* and *abandoned* use subclauses. There are numerous syntactic structures in these definitions that are hard to capture with single binary relations.

[8]The definition of *mustache* assumes that the person is male, which is in contradiction with the gender of the teacher – female – a fact assumed to be previously established.





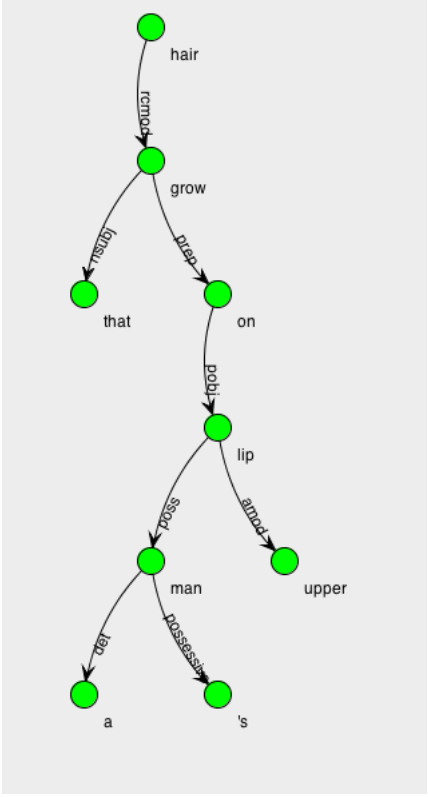


Figure 3.1: Dependency parse trees of definitions with deep structures

Example	Source	Definition
1	mustache *LDOCE*, noun	hair that grows on a man's upper lip
2	ride *LDOCE*, verb	to sit on an animal, especially a horse, and make it move along
3	abandoned *LDOCE*, adjective	an abandoned building, car, boat etc has been left completely by the people who owned it and is no longer used

Table 3.4: Definitions which are hard to capture in binary relations

As noted previously, Barrière's knowledge base, or *Extended WordNet* (Mihalcea, 2001) Logic Form (LF) attempt to model the meaning of a word in expressive representations. Instead of defining a word or frame in an expressive representation, FrameNet uses mappings between frames, which indicate when a frame uses (is composed of or defined using) another frame.

FrameNet's mappings indicate how frame elements are mapped between frames but not if a particular value is assigned to a frame element. These mapping still provide some type of relational comparison between different frames. As mentioned earlier, Rus used syntactic relations between concepts, which have significant limitations when attempting match semantically equivalent roles with different syntactic realizations. Barrière used CGs (see Section 3.7) to represent a definition and a predetermined set of semantic relations to connect the concepts in the definition.

Semantic roles from various resources will differ by the number of roles and by what types of concepts they connect. Furthermore, some roles are very context-specific, while others are so general that they are hard to interpret in any particular concept. The Component Library (CLIB) (Barker et al., 2001), for example, has about 200 roles (and they are far from exhaustive) organized by what they connect (Noun-to-Noun, Verb-to-Noun, Verb-to-Verb, etc). Many of CLIB's roles are only appropriate for a small set of predicates. Other systems may use a small set of one or two dozen roles, which are applied to everything. Usually these roles focus on relations between a verb and its arguments or adjuncts.

3.6 Semantic Networks

Semantic networks are a knowledge representation where concepts are represented as nodes in a graph; labelled edges indicate relations between concepts. These networks are similar to Charles S. Peirce's "existential graphs" (Peirce and Sowa, 2010; Roberts, 1973). Semantic networks became popular in the 1970s and are still in use today. *WordNet* is an example of a modern semantic network; the senses (or synsets) are equivalent to concepts, and there are labelled relations between senses/concepts. A semantic network encodes knowledge in the relations between concepts; to understand a concept we should understand how it relates to other concepts. Ontologies are often represented as semantic networks.

3.7 Conceptual Graphs

Sowa (1984) derived CGs, which have well-defined semantics and a set of operations for reasoning, from the idea of semantic networks. CGs are directed bipartite graphs composed of concepts and relations. Concepts are graphically represented as rectangles and relations as ovals. Concepts can only be connected to relations, and relations only to concepts.

Figure 3.2 shows an example of a simple CG for the sentence "The cat sat on the mat". The *mat*, *cat*, and *sat* are concepts while *on* and *agent* are the relationships connecting these concepts.[9]

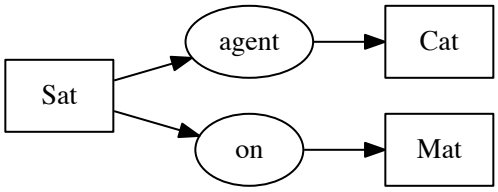


Figure 3.2: Example of a simple conceptual graph (CG)

CGs support many features common in predicate logic, but less common in graph-

[9]More examples and variation of this CG are available at `http://www.jfsowa.com/cg/cgexampw.htm`.

based knowledge representations, such as quantification, negation, and propositions. Each concept can have quantifiers, such as *for all*, *there exists*, *multiplicity values*, and even *sets of individuals*. Individuals may be named, assigned a variable, or coreferent to other concepts. Propositions are represented as rectangles containing graphs; propositions are treated as conjunctive negations (logical NAND). Using conjunctive negations, all other common propositions can be composed: conjunction (logical AND), disjunction (logical OR), negation (logical NOT), and implication. Many semantic networks are a single network or graph relating all concepts to one another, while a CG usually defines a single statement or proposition. A CG knowledge base is composed of many statements represented as many graphs.

As a general-purpose knowledge representation CG does not restrict itself to a fixed set of relations or concepts. A significant benefit of CGs is that it provides a syntax for defining new concepts and relations in terms of other concepts and relations. Thus, the language of conceptual graphs can be extended by their own statements.

CGs are an active area of research and an ISO standard knowledge representation under the Common Logic Effort (ISO/IEC IS 24707:2007). CGs can be used directly for reasoning or converted to other logic forms.

3.8 Frames

There are a number of uses for the word *frame* in linguistics and knowledge representation: syntactic frames, case frames, subcategorization frames, temporal frames, and frame semantics. In this work, *frames* will refer, unless otherwise qualified, to frame semantics. Frame semantics does not define a formal knowledge representation, but an idea about memory and concept knowledge; thus, frames must be used in the context of another knowledge representation. Frames can easily be represented and used in CGs (Sowa, 2000).

Frame semantics (Fillmore, 1976) share much in common with ideas expressed by Minsky (1975a) and Schank and Abelson (1977). Frame semantics suggests that, when a concept is evoked in our memory, it includes all our past experiences, memories, expectations, and prototypes of the concept. In Fillmore's words:

> ... the contexts within which we have experienced the objects, properties or feelings that provide the perceptual or experiential base of our knowledge of

> the meaning of a word (or phrase, or grammatical category) may be inseparable parts of those experiences.

Initially, frames seem similar to other conceptual representations. Frames have slots, or frame elements, which are like relations in CGs, whose values specify details about the event, object, or idea represented; for example, frame elements may identify actors, props, conditions, provide descriptions, or modifiers. Furthermore, these slots may have default values, likely values, and value restrictions. Yet a frame encompasses more than just a simple concept, definition, or a lexical entry.

Frames set themselves apart from other conceptual representations because they include information about prototypical examples, expected sub-events, likely locations, or causes that come from association or past experience. Some of these ideas appear in *scripts* proposed by Schank and Abelson (1977).

Let us consider Fillmore's example of a frame we might call the *commercial transaction*. This frame may contain roles for buyer, seller, goods, and price. We would expect sub-events: buyer gives money and receives goods; seller gives goods and receives money. We also have an understanding that ownership of the goods is transferred in this frame. This frame might be evoked by words like "buy", "sell", "pay", "cost", "spend", "charge" (Fillmore, 1976). While these words evoke similar images in our minds, the definitions of each word are not identical - each word may suggest slightly different perspectives, the actors, or results; the use of a single frame to represent a commercial transaction will provide a uniform representation for words that are not equivalent, but where the evoked idea is the comparable.

Some words will evoke similar frames, but provide different expectations about circumstances, histories, and characters. Consider the following words, which involve an exchange of money but might evoke very different images, settings, characters, motivations, and histories: "alimony", "bribe", "tip", "rebate", "change".

> One of the most characteristic features of intelligent activity is knowing more than you are told (Fillmore, 1976).

Frames as a conceptual structure go much deeper than lexical definitions and include many more details, which will be important in reasoning, structuring, and indexing knowledge. Furthermore, they allow a certain form of common-sense reasoning, assuming more than we are told.

Frames are a large knowledge structure incorporating more information than a traditional dictionary definition. This idea seems compatible with, even related to, ideas

presented by Cruse (1986) about connecting and organizing dictionary definitions. The frames that this work proposes should encompass many definitions, indicating how they are related; a single frame should be usable in many different situations.

3.9 Web Ontology Language (OWL)

OWL is a set of three expressive description logics used to construct Knowledge Bases (KBs) and ontologies. Each of the languages increases in expressive power, thus also increasing potential computation required to reason about a KB. The languages are expressed in XML and are World Wide Web Consortium (W3C) standards. One goal has been to create a semantic web of information, thus these standards enable ontologies and KBs to be shared and reused. There are a number of existing and reusable ontologies such as OWL-Time (Hobbs and Pan, 2006).

OWL is an appealing knowledge representation because of the existing tools (Sirin et al., 2007; Klinov and Parsia, 2010) and resources that already support it, but it is not the right representation for this research, because OWL is a highly formalized knowledge representation with strict semantics that are required to facilitate reasoning. The idea of soft definitions and modulation would violate these strict semantics. A more flexible representation is needed to support modulation.

3.10 Packed Graphical Representation (PGR)

As Kilgarriff (1992) noted, there are circumstances where more than one sense is appropriate. This suggests a need for a representation and approach that considers multiple interpretations. This differs from most approaches used by NLP systems today, particularly those systems that are used for text understanding and annotating predicates, which tend to annotate a single label.

Usually, parsing, Word Sense Disambiguation (WSD), and Semantic Role Labeling (SRL) are applied in a pipeline, each assuming the previous step correctly assigns a single correct label, thus carrying no ambiguity forward and feeding back no information. Yeh et al. (2006) showed that combining WSD and SRL can improve both; this is expected since a role may restrict the valid senses for the predicate and argument, and likewise the sense restricts which roles are appropriate. One way to join SRL and WSD is by carrying multiple interpretations between the processes.

Some pipeline systems will carry multiple possible interpretations through the pipeline, to allow later steps to determine which is the best interpretation. Kim et al. (2010) developed the Packed Graphical Representation (PGR), which extends a single dependency tree to represent multiple interpretations, by storing only the differences. The Packed Graphical Representation supports many types of ambiguity in interpretation:

1. Concept type ambiguity - the type or sense of concept may vary.
2. Relational ambiguity - the assigned relation between two concepts may vary.
3. Structural ambiguity - the structure of the representation, *i.e.*, which concepts are connected.
4. Co-reference ambiguity - which terms or concepts refer to the same object may differ.

Kim et al. showed that it is more computationally practical to carry multiple interpretations forward as differences (as PGR does) instead of complete trees. Furthermore, delaying ambiguity resolution can lead to better text understanding. Section 4.4 will introduce Multiple Interpretation Graphs (MIGs), which is built on Packed Graphical Representation, but add support for expanding concepts to auto-frames, ambiguity of sense assignment, and ambiguity of role/attachment point assignment.

3.11 Applications of Knowledge

To evaluate and apply the lexical knowledge base from this thesis, let us consider applications that clearly benefit from access to lexical knowledge base and machine reading systems. Tasks such as entailment, question answering, and reading comprehension can perform much better with knowledge and understanding of a text.

3.11.1 Textual Entailment

Textual entailment is the task of deciding, given two text fragments, whether the meaning of one text is entailed by (known to be true from) the other text (Dagan et al., 2009). Entailment is a directional evaluation, because it must account for implied knowledge and understanding. Consider A: *John was murdered* and B: *John is dead*. A entails B, because if John was murdered, a consequence (inference) is that he is dead, but the

entailment does not hold in the other direction. If John is dead, he was not necessarily murdered. Formally, if A entails B then B should always be true when A is true.

Many techniques exist for testing entailment. The methods range from simple bag-of-word techniques, syntactic transformation, machine learning, paraphrasing, predicate-argument structure comparisons, to logic and automated reasoning approaches. The approach used in this research is comparable to predicate-argument structure comparisons or graph-based comparisons: Does the query predicate (graph) match some predicate (sub-graph) in the knowledge base? The entailment testing done in this thesis is limited to testing that an answer entails a question for reading comprehension.

Frequently, entailment is considered true when one statement is probably true given that the other is true. This idea seems consistent with Fillmore's claim that "One of the most characteristic features of intelligent activity is knowing more than you are told" (Fillmore, 1976). Voorhees (2008) explains this well, with regards to entailment in the PASCAL RTE Challenges:

> RTE uses an "ordinary understanding" principle for deciding entailment. The hypothesis is considered entailed by the text if a human reading the text would most likely conclude that the hypothesis were true, even if there could exist unusual circumstances that would invalidate the hypothesis. It is explicitly acknowledged that ordinary understanding depends on a common human understanding of language as well as common background knowledge.

Entailment is an "abstract generic task that captures major semantic inference needs across applications" (Dagan et al., 2009). Thus, entailment systems may be used in Question-Answering (QA), Information Retrieval (IR), reading comprehension, and summarization tasks. In QA, one could test if an answer (or the supporting evidence) entails the question; if so, then the answer is plausible. In summarization, entailment can be used to detect redundant information. While a generic entailment system can provide benefits to each of these tasks, the requirements from an entailment system for each task may differ.

Entailment is not the ideal task for evaluating this work, because the context is limited to a single sentence. Furthermore, the research area is saturated with successful systems, which cannot read but can do well at entailment, making for difficult comparisons. Finally, entailment is complicated by information not represented in this thesis, such as quantification and negations.

3.11.2 Question Answering

QA is the task of returning answers to questions rather than referencing documents, as is done in information retrieval (IR). The task has immediate value in real-world applications; QA systems can provide answers to users' questions on the Internet or from a large document collection. Furthermore, it can be used as a support system for many other practical applications, such as summarization. QA systems tend to follow the same general methodology: analyze the question, retrieve documents or passages, perform detailed matching on the retrieved text, rank, sort, and return answers.

In its simplest form, question analysis consists of determining the expected answer type (Loni, 2011; Toba et al., 2011). The availability of SRL systems has allowed QA systems to build predicate-argument structures (Schlaefer et al., 2007; Sun et al., 2005). The question is represented as a predicate-argument structure; then text retrieval and matching can use the structure for search or comparison.

Document retrieval tends to use standard IR techniques as they are fast, well studied, and available in many public systems. Most systems will, at minimum, use word frequency and query expansion.

Detailed matching is where systems differ the most; some systems use logic forms and reasoning, some apply syntactic transformations, some use graph matching, others machine learning methods, and some predicate argument structure. This step of the QA pipeline is where entailment systems are applied. Recently, the Ephyra system used the ASSERT (Pradhan et al., 2004) SRL system to produce a predicate-argument structure, which increased the accuracy of the QA results (Schlaefer et al., 2007).

For years, the Text REtrieval Conference (TREC) has hosted QA tracks (Voorhees, 2004; Dang et al., 2007) which have been used to evaluate QA systems. This has been the standard evaluation for many QA systems. The corpus and types of questions have changed over time as the challenge's goals evolved. The data is publicly available and can still be used in the development and testing of new QA systems.

Large corpus QA is not the ideal evaluation of this thesis, since reading the entire corpus may not be practical or possible. Reading a small number of documents, filtered by some other method is possible, and can lead to good results. However, the majority of the documents and questions involved in QA are of a reading and vocabulary level above what this thesis aims to achieve.

3.11.3 Reading Comprehension

Some researchers have considered reading comprehension tests as an evaluation of intelligent systems (Hirschman et al., 1999; Charniak et al., 2000; Riloff, 2000; Wellner et al., 2006). These tests provide comparison to human performance. The tests are produced for different grade levels and ultimately assess a reader's understanding of text relative to an educational grade level (*e.g.*, the readers comprehension is that of an average grade 3 student). As systems are refined, they can track their improvement as they progress through primary school reading to graduation (*i.e.*, ready for real-world tasks).

A reading comprehension test consists of a story (depending on the reading level, possibly a news article); the reader is questioned about the story. As the reading level increases, the level of abstraction and inference needed for understanding increases. An important difference between reading comprehension and QA evaluations is that reading comprehension has a single short document and the questions focus on understanding it. Many QA systems require large amounts of redundant data; these tests would struggle with this task. Furthermore, this task encourages text understanding more than QA does.

Cross-Language Evaluation Forum (CLEF) has recently included a reading comprehension evaluation for QA systems (Rodrigo et al., 2010; Peñas et al., 2011, 2013). The data set from this challenge provides an evaluation data set for reading comprehension research. Yet, I argue that this reading comprehension evaluation is inappropriate for my goals, as it is in the medical domain, which is far too complex for most systems to compare with human performance. Instead, I have opted for use of the REMEDIA Story Comprehension Corpus[10] also used by Hirschman et al. (1999), Riloff and Thelen (2000), and Hirsch Jr (2003).

The REMEDIA corpus contains plaintext stories, as well as a number of pre-annotated support files, such as annotated Named Entities (NEs), SRL annotations, coreference labels and more. These files are used to test a system under a variety of conditions and compare different components within a system, for example, a comparison of a system's performance with its own NE recognizer and with a perfect NE recognition (contained within the annotated files). Hirschman et al. (1999) describe a number of evaluation metrics used with reading comprehension.

[10] `http://cogcomp.cs.illinois.edu/page/resource_view/11`

3.12 Summary

Chapter 3 examined the challenges related to extracting lexical knowledge from a dictionary. While dictionaries are semi-structured, they are limited in content and presume a certain minimum knowledge from their target audience. The problem of representing the meaning of a word is further complicated by the understanding mechanism used by humans; humans understand words based on experience and assumptions, and can adapt meaning based on context. A number of knowledge representations are introduced, each with its own set of benefits. Finally, a number of tasks are described that commonly apply such knowledge bases. The reading comprehension task acts as a good evaluation for a machine reading system. The following chapter explains my theory for automatic extraction and application of lexical knowledge based on these ideas.

Chapter 4

Theory: Auto-frames

This chapter introduces my theory, which is based on and refers to previous work in the field. It addresses the previously unaddressed issues about the representation of word senses. Fundamental concepts, such as an auto-frame[1] will be introduced and theoretically defined. I will introduce the knowledge representation Multiple Interpretation Graph (MIG) to be used in this research, and justify its selection. I will explain the theory behind making auto-frames soft[2] and how this relates to and addresses issues from the literature review and the background chapter. Lastly, I will contrast FrameNet with auto-frames, since both are realizations of the same theory: frame semantics.

This chapter also answers a number of questions about the theory this thesis proposes to achieve its goals. Some key questions answered in this chapter are:

1. Why start from a dictionary?
2. What is automatic extraction of a lexical knowledge base?
3. How to generate semantic roles instead of applying a predetermined set of roles?
4. What is a soft lexical knowledge representation?
5. How to support multiple interpretations within the representation?

[1]This name, auto-frame, provides enough distinction as not to be confused with temporal frames, frames from description logics, FrameNet frames, and plethora of other concepts known as frames. "Auto-frame" was chosen as the name to emphasize that these frames are meant to be automatically constructed.

[2]Section 3.5 defines the term *soft definitions*. An auto-frame will be soft because it will not require a single strict meaning. Instead meanings will be contextualized, which will allow the meaning to change slightly from context to context.

6. How to support sense modulation?

4.1 Why Dictionaries

I have chosen to build auto-frames starting from dictionaries because I consider lexical knowledge to be near the start of the knowledge spiral. As a system progressively learns along the knowledge spiral, eventually encyclopedias, textbooks, and general corpora will all be important sources of knowledge.

A major goal of this research is to express lexical knowledge in a soft representation, which will support contextualization. Since the problem focuses on word meaning, this knowledge extraction task starts with dictionaries, a type of resource focused on word meaning.

A dictionary is a resource relatively easy to process automatically, as the text is usually well structured, uses consistent writing styles, and is grammatically correct. The definitions are usually concise, yet effective at conveying meaning. Concise definitions are good for knowledge extraction, because there should be almost no extraneous details, though some pertinent details may be absent.

There are several resources which could be considered for this work: *Longman Dictionary of Contemporary English* (*LDOCE*) (Procter, 1978), *WordNet* (Fellbaum, 1998), *Wiktionary*, Cobuild (Cobuild, 2006), or the American Heritage First Dictionary used by Barrière and Popowich (1999). *Wiktionary* provides a number of benefits over other resources: it is large, open-access, and continuously being updated. However, *Wiktionary* is not as strictly edited or controlled as other dictionaries: it does not have a controlled vocabulary, consistent definition styles, or clear phrasing. Since this work starts near the beginning of the knowledge spiral, *Wiktionary*, lacking the previously mentioned features, is not an appropriate starting point.

LDOCE has a controlled vocabulary, consistent definition styles, and clear phrasing, thus making it a good starting point.

WordNet and *Extended WordNet* (Mihalcea, 2001) are popular in Natural Language Processing (NLP) because the glosses can be treated as definitions and synsets as senses. However, these resources are not dictionaries. *WordNet* and *Extended WordNet* glosses are not as clear and simple as the definitions in *LDOCE*, although leveraging the semantic network and public availability of the resource would be beneficial. The relations in *WordNet* may provide some measure of which senses should be related. Or the Logic Forms (LFs) in *Extended WordNet* (Rus, 2002) may provide a knowledge extraction com-

parison (see Section 3). Companion resources, like coarse-grain sense clusters (Mihalcea and Moldovan, 2001; Navigli, 2006; Navigli et al., 2007), may also assist in comparison and evaluation.

WordNet 3.0 has a number of standoff files, which provide useful extensions and additional information for *WordNet* synsets and glosses. Of particular use to this work are the standoff files which sense-tag words in the *WordNet* glosses. While these sense tags are incomplete, they can still help to correctly assign and interpret senses, or they can be used as an evaluation resource.

Even with all the benefits associated with *WordNet* and *Extended WordNet*, the main focus of this research will be knowledge extraction from *LDOCE*. With the focus on building soft definitions, using the clear concise definitions and simple language of *LDOCE* is more important than the additional resources in *WordNet*. For the purposes of evaluation, *WordNet* and *Extended WordNet* will be used to evaluate part of knowledge representation and extraction processes.

4.2 Auto-frames and Semantic Roles

How should someone choose a set of semantic roles? Any set probably has specific motivations and goals, though how do they perform on those goals? Any attempt at an empirical evaluation using a set of semantic roles implicitly acts as evaluation of the roles, annotated or training data, and any methodology and components associated with a particular system. Semantic roles cannot be evaluated in isolation; any indirect evaluation will be highly influenced by training data, methodology, and software. Which semantic role set should be used for this lexical knowledge base?

In this research, I avoid selecting a set of semantic roles and instead focus on mappings between related auto-frames. Instead of a predetermined set of roles, the dictionary (specific definitions and parts of definitions) will reveal roles for each sense. Each auto-frame will have a set of roles associated with it, referred to as attachment points; section 4.5.1 provides more details regarding auto-frames.

I do not commit to any specific set of semantic relations. Instead, I focus on interpreting nouns and verbs – connected by semantic roles – with respect to their definitions and auto-frames. For example, consider the statement, *John travelled from Toronto to Ottawa.* We could assume a word like *travel* has a semantic role *origin or source* and another *destination, goal, or direction*, which are satisfied by "from Toronto" and "to Ottawa" respectively. Instead, if we examine definitions of travel found in Table 4.1, we

see that both *LDOCE* and *WordNet* specify that they involve going from one place to another.

Example	Source	Definition
1	travel *WordNet*, verb	change location; move, travel, or proceed, also metaphorically
2	travel *WordNet*, noun	the act of going from one place to another
3	travel *LDOCE*, verb	to go from one place to another, or to several places, especially to distant places
4	ride *LDOCE*, verb	to travel on a bicycle or motorbike

Table 4.1: Examples of definitions which define their roles or frame elements

This example was chosen because the example roles were clearly present, but most roles are not this clear. Sometimes, the roles of the genus term must be considered to find the appropriate roles for a context. For example, consider the sentence *John rode from Toronto to Ottawa.* The definition of ride is provided in Table 4.1 and does not convey any roles, which could match *from Toronto to Ottawa*. However, the genus term of *ride*, *travel*, as we have previously seen, can satisfy these roles.

The problem of finding and labeling roles is even more difficult when considering properties and attributes of nouns and verbs. Consider *weight, price, speed,* and *colour* as roles, as many resources do, and examine their definitions (see Table 4.2). These words – attributes – are not found in the definitions of *object, move, or purchase*, instead the definition of weight, price and colour indicate what they modify or describe: *objects, motions, or purchases*.

To extract roles (properties) of this nature requires connecting definitions with what they modify. With the exception of *colour* from *LDOCE*, these roles are defined as answering a question. *LDOCE* frequently uses words like *how*, *measure*, or *amount* to indicate these types of properties. *Extended WordNet* (Mihalcea, 2001) tends to use *attribute* or *property*, though *measure* and *amount* are also used.

Example	Source	Definition
1	price *LDOCE*, noun	the amount of money for which something is sold, bought, or offered
2	price *WordNet*, noun	the high value or worth of something
3	price *WordNet*, noun	the property of having material worth
4	weight *LDOCE*, noun	how heavy something is when measured by a particular system
5	weight *WordNet*, noun	the vertical force exerted by a mass as a result of gravity
7	colour *LDOCE*, noun	red, blue, yellow, green, brown, purple etc
8	colour *WordNet*, noun	a visual attribute of things that results from the light they emit or transmit or reflect
9	speed *LDOCE*, noun	how fast something moves or travels
10	speed *WordNet*, noun	distance travelled per unit time
11	object *LDOCE*, noun	a solid thing, especially something that you can hold or touch
12	move *LDOCE*, noun	to change your place or position, or to make something do this
13	purchase *LDOCE*, noun	to gain something but only by losing something else

Table 4.2: Examples of definitions which define properties of other frames

4.2.1 Auto-frames as Soft Definitions

I propose an approach inspired and motivated by the theory of semantic frames (Fillmore, 1976). Auto-frames will evoke[3] more information than a definition. A definition is a concise way to communicate a meaning to a particular audience,[4] where the reader is expected to fill-in intelligently or know related information. In contrast, an auto-frame includes related information, such as common scenarios or contexts, expected sub-events or activities, purposes or goals. That is to say, the auto-frame requires less intelligence because no "filling-in" is needed. For example, *bribe* usually refers to purchasing favor in a criminal way, but this is not necessarily true, as parent's are often said to *bribe* children into good behavior. The *bribe* auto-frame should include the fact that the context is usually criminal, but it should be flexible enough for use outside of criminal scenarios. Another example: *oiling* could have a definition of "applying oil to a machine" but an auto-frame should include the purpose "to make it operate smoothly or efficiently". The auto-frame for the concept of restaurant should include information about sub-events and actors related to this: food, ordering, waiters, tips, etc.

Evoked knowledge may allow an auto-frame to fit more contexts and provide common-sense implications, such as understanding that *going to a restaurant* suggests *eating there*, though it is not strictly true or known. This evoked knowledge may actually lead to inconsistency or contradictions in a knowledge base, from the formal logic standpoint. I will address this problem of consistency of evoked knowledge in the Contextualization section (4.6.1).

Each auto-frame will contain at least one sense (possibly more if they are closely related) and information about the context of the sense. One might consider an auto-frame as a large coarse-grained definition, which should include expectations, defaults, and implications. This idea is consistent with Fillmore's description of semantic frames, and similar to Concept Clustering Knowledge Graphs (Barrière, 1997). Section 5.2.4 describes the specifics of how auto-frames are constructed and which senses are included. The process of contextualization reduces all this coarse information to an appropriate fine-grained definition. Thus the auto-frame assignment is coarse, but the information that remains after contextualization (expansion, attachment, and pruning) is a context-specific definition.

[3]To evoke a memory or idea is to bring it into conscious mind from memory. To evoke an auto-frame will be to fetch it from storage and include it in an active process and knowledge representation.

[4]The audience could be, for example, someone who knows another language, young child learning, or a native speaker.

Each auto-frame will have roles, which we will refer to as attachment points, which will help in understanding the relation between two auto-frames in context. Metaphorically speaking, one might view an auto-frame as a puzzle piece, with the attachment points as the interlocking parts; creating an interpretation, that is, assigning the senses and relations/attachment points, is the process of trying to fit the pieces together.

There are three ideas that I have adopted from the literature review and background to make auto-frames support soft definitions. First, an understanding that a word in context often requires more than just a strict sense; consider what a word evokes, an understanding of related senses, and how they are related. This is the primary function of the auto-frame, to provide additional information which is not necessarily relevant to all contexts. Second, an auto-frame may need to be modulated to fit the context; that is, the added information may need to be pruned (removing unrelated details or relaxing a requirement or implication). Lastly, the knowledge representation should allow for more than one interpretation of an expression.

4.3 A Knowledge Representation

There are a number of possible representations that can be used to represent frames and support formal logic. Two common knowledge representations are Conceptual Graph (CG) and Web Ontology Language (OWL); both have clearly defined automated reasoning processes, have implemented and publicly available reasoning systems,[5] are well-defined logics,[6] and have extensive community support.

While this research originally aimed at using a knowledge representation that directly supported logic and reasoning systems, it also requires a flexible representation. The representation needs to support storing both syntactic and semantic information, as CG does, but also to support multiple interpretations, inconsistent information, and possibly non-monotonic reasoning. Without significant extension to CG, it cannot support holding multiple interpretations, that is, multiple distinct senses for a single concept and multiple different types for a relation, as well as variable structure (that a relation might exist in some interpretations and might not in others). Furthermore, there seems to be no way to include what is evoked by a concept without replacing the original concept

[5]Reasoners for OWL: Pellet and Pronto (Klinov and Parsia, 2010); reasoners for CG Anime (Kabbaj, 2006), CGPro (Petermann et al., 1997).

[6]CG can fully express the Common Logic standard (ISO standard 24707), while RDF and OWL express only a subset of the standard; thus CG is more expressive than OWL.

with its definition.

I expect that contradictions may arise from evoked information and contextual information (*e.g.*, the context may declare someone as female; however, a statement in an auto-frame may suggest that it is a male (imply because of default knowledge). A representations, such as CG or OWL, with strict semantics intended to support automated reasoning, would allow us to quickly determine such contradictions. These contradictions should never exist in such a representation, as they lead to reasoning problems. Both OWL and CG require ontologies, which form either a tree or lattice (*i.e.*, they do not support cycles); however, there may be cycles in the knowledge base I intend to extract.

The goals of having a representation and process that models evoked information, common sense, and contextualization (modulation and filtering) does not seem compatible with strict representations such as OWL and CG. It should be possible to select one interpretation from an MIG and convert it into a strict formal knowledge representation, such as OWL or CG.

Throughout this research, it has become apparent that the process of interpretation of text should not be constrained to be sound or complete, as reasoning systems require. Instead, automated reasoning may be a post-process of an interpreted text; thus I have put aside the restrictions imposed by many logics. The process of *interpreting text* is the determination of which senses to assign to each concept and how to attach concepts.

In the following section I describe the MIG representation, which evolved through the course of this research. I will not describe intermediate ideas or representations, but instead present MIG as it has been presently realized in this work.

4.4 Multiple Interpretation Graphs (MIG)

4.4.1 Introduction

In this work, I use a representation that I call Multiple Interpretation Graph (MIG) (described in section 4.4.2), which fully supports Packed Graphical Representation and more. Let an *interpretation* refer to a representation with assigned senses and roles. Let an interpretation process be one that assigns senses and roles to a representation and applies any contextualization if required. In short, this is the process of selecting what this sentence is attempting to communicate.

MIG can function as an intermediate representation between knowledge extraction processes and labelling processes, such as Word Sense Disambiguation (WSD) and Se-

mantic Role Labeling (SRL), allowing access and annotation of multi-interpretations. When processing multiple sentences, for example a story or news article, nearby sentences (or sentences not yet processed) may influence the final interpretation. Thus storing multiple interpretations can allow future sentences, discourse analysis, or some other high-level process to contribute to the interpretations process.

Support for multiple interpretations and multiple senses will help us build a soft representation of meaning. Fillmore's semantic frames are an idea about memory, but they are less strict than traditional dictionary definitions. To implement something like semantic frames, auto-frames will connect closely related senses, effectively making a single auto-frame for nearly inseparable definitions. Section 5.2.4 describes the method for clustering senses into a auto-frame. Briefly stated, derived senses are included with the root sense; senses for the same term which share a common genus term (such as transitive and intransitive verb definitions.) are merged into a single auto-frame. As we will later see, this is intended also to satisfy the conclusions of other researchers who think less distinct senses and definitions are needed yet more structure within definitions. Thus, auto-frames, based on Fillmore's idea, evoke much more than a single dictionary definition; but an auto-frame may need to be *contextualized*.

Consider the clause "*Bill is going to the bank*". Let us consider the word *bank* and what may be the intended sense. The context is that *Bill is going to a* location; it is likely and expected that *Bill* is going both to the bank as a *building* and as a *financial institution*. The representation should not be restricted to selecting only one valid sense, but should both senses of bank appear in the same auto-frame? The sense *bank as a building* refers to the building in which a financial institution is located. This sense is clearly derived[7] with the first sense, bank as *financial institution*, and in many contexts both are probably evoked. By merging both senses into the same auto-frame, we can assign an auto-frame to the context and accept both senses as valid interpretations in this context.

4.4.2 A Closer Look at MIG

The MIG representation draws on ideas from Packed Graphical Representation, dependency graphs, the Stanford Dependency tree (de Marneffe et al., 2006; de Marneffe and Manning, 2008) representation, and semantic role labeling. It may be considered as a

[7]Some derived definitions use the word being defined, although it should be noted that a derived definition in fact refers back to different sense/definition.

graph of graphs. Let us define the MIG representation starting from the simplest units.

MIG is used to contain dependency graphs, and in this research we have used graphs from the Stanford parser. Each node in the graph can contain multiple annotations, such as lemma, part of speech (POS), original text, position, whether it is the root of the tree, whether it is a clause head, etc. The use of annotations has made the representation flexible and extensible. Any processing can add annotations as needed; annotations may be for internal use (like a cache) or for later processes.

Each dependency graph also contains a list of edges. Each edge also supports annotations.

Two additions are needed to make a dependency parse tree function as a Packed Graphical Representation. First, all nodes and edges should be grouped into interpretation sets, or macros as Kim et al. (2010) refer to them. An interpretation set collects nodes and edges that must be present together;[8] that is, they are always in an interpretation together or not at all. The second thing needed is interpretation restrictions.[9] Interpretation restrictions bind interpretation sets together. Kim et al. define two restrictions: simple and mutual exclusion. Let X and Y represent interpretation sets. A simple restriction means set X must be present for set Y to be present in the interpretation; or Y depends on X. A mutual exclusion (XOR) restriction means X and Y cannot both be present in the same interpretation.

In this work, as will be shown later, I have found one more restriction practical: inclusion (OR); if A is in the interpretation, then at least one of X, Y, Z, ... must be included. The restriction is used to indicate at least one of these senses, implications, or attachments must be true.

Consider the example found in Figure 4.1 for the sentence "I saw a man with a telescope". There are two dependency trees which are likely and would form valid interpretations; either the *seeing* was done using the telescope, or the *man* had a telescope. These two interpretations are mutually exclusive. The parse tree is broken into interpretation sets (visually represented as grey rectangles and marked as interpretation sets). Notice how one common interpretation set contains "I saw a man", because the structure occurs in both parse trees. However, there are two other interpretations sets depending

[8]These concepts and relations always appear together in all represented interpretations that include them. Consider that Packed Graphical Representation was developed as way to express concisely multiple parse trees; in the set of parse trees, these concepts always exist together in the same subtree, branch, or structure.

[9]Or dependency relations as Kim et al. (2010) refer to them, though this terminology can be confused with dependency relations from the dependency parser.

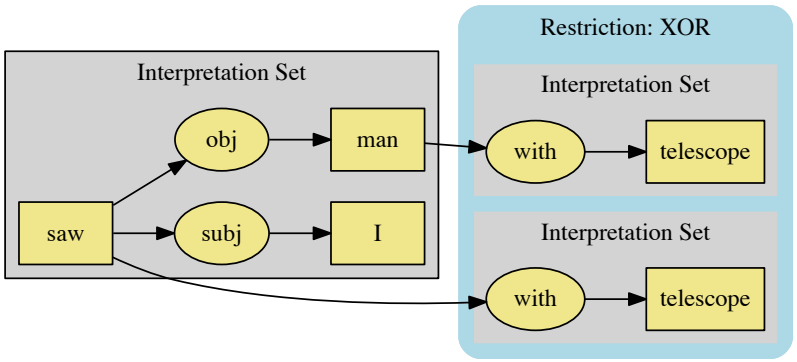


Figure 4.1: Example MIG restrictions and two interpretations: *I saw a man with a telescope*

on where "with a telescope" is connected the common interpretation. A XOR restriction is placed on the two "with telescope" interpretation sets. Figure 4.1 does not depict it but we would define the XOR restriction as dependent on "I saw a man" interpretations set (referred to as the head of the restriction).

In the following pages, I will use grey boxes, like the one below, to provide a formal definition of MIG.

Example of a box where the definition of MIG will be stated formally.

Each box relates to the neighbouring textual description but formalizes the concepts using sets and tuples. A box defines a particular aspect of MIG: nodes and relations, interpretation sets and restrictions, links, MIG as whole, common MIG operations, expanding MIGs. The following is a formal definition of annotations, nodes, relations. Next – a formal definition of interpretation sets, interpretation restrictions, and graphs.

Let a be an annotation, defined as a tuple (k, v), where k is the key and v is a value. A is a set of annotations $\{a_1, a_2, ..., a_i\}$

Let n be a node defined by a tuple (t, A_n), where t is the type of node (word, word sense, or other value indicating what the node denotes), and A_n is set of annotations specific to this node. N is a set of nodes $\{n_1, n_2, ..., n_i\}$

Let r be a relation defined by a tuple (rt, n_1, n_2, A_r), where rt is the type of the relation, n_1 is the starting node of the relation, n_2 is the ending node of the relation, and A_r is an set of annotations specific to this relation. R is a set of relations $\{r_1, r_2, ..., r_i\}$

Let is be an interpretation set defined by the tuple (N_{is}, R_{is}, L_{is}), where L_{is} is a set of links (to be defined later). N_{is} and R_{is} are sets of nodes and relations, respectively. IS is a set of interpretation sets $\{is_1, is_2, ..., is_n\}$
Let ir be an interpretation restriction defined by the tuple $(is_1, IS_2, restriction)$, where $restriction$ is the type, which must be a value from the set $\{Simple, XOR, OR\}$, is_1 is the head interpretation set on which all of the interpretation sets in IS_2 depend.
All restrictions require that is_1 must be in the interpretation for any part of IS_2 to be in the interpretation. A simple restriction only has the condition above (that is_1 must be in the interpretation). A XOR restriction means that only one interpretation set from IS_2 can be in the interpretation and is_1 must be in the interpretation. An OR restriction means that at least one interpretation set from IS_2 must be in the interpretation if is_1 is in the interpretation.
Let IR be a set of interpretation restrictions $\{ir_1, ir_2, ..., ir_n\}$
Let g be a graph defined by the tuple $(N_g, R_g, IS_g, IR_g, A_g, entry)$,
where $entry$ is an interpretation set from IS_g; the $entry$ interpretation set (which may be empty) must be true for all interpretation of this graphs. The entry interpretation set is the starting point for constructing an interpretation for this graph. IR_g is the set of interpretation restrictions for graph g.

At this stage, any dependency graph obtained from the parser is referred to as a *surface graph*. The surface graph is the dependency graph obtained from parsing a

particular text that is to be interpreted. The surface graph is *expanded* upon by the interpretation process. *Expansion* is the addition of knowledge (auto-frames) from our knowledge base to the MIG for the purpose of interpretation.

Expansion is part of the connection search process (Section 5.2), which is used both when building frames and interpreting sentences. When building auto-frames, expansion is the process of including and connecting information from another sense. When interpreting a sentence, expansion is the inclusion of auto-frames as possible senses of a concept. In NLP, a concept is normally assigned one sense, instead this thesis expands the representation to include the auto-frames for all possible senses of the concept. This *expansion* of a concept to its auto-frame does not replace the concept but adds an additional graph connected by a *link*.

The *links* are not like dependency relations,[10] instead they may connect two graphs, connect a node from one graph to another graph, or may connect two nodes in the same or different graphs. Furthermore, links and graphs may be associated with interpretation sets.

Consider the sentence *John went to the bank*; the MIG is illustrated in Figure 4.2. We might expand *bank* to multiple senses, such as a riverbank and a building containing a financial institution. We would then restrict *bank* with an mutual exclusion (XOR) restriction to the two senses. Figure 4.2 shows grey nodes that represent the word senses; the XOR restriction is used to indicate that only one of these senses is expected to be valid (included) in an interpretation.

In Figure 4.2, the links (dotted lines) connect the concept being expanded (*bank*) to the graphs (referred to as *sense graphs*) of the definitions or auto-frames for those word senses. The example provided has been simplified for the purpose of visualization; the grey rectangle would in fact contain the sense graph. It should also be noted that in this example, each complete sense graph is a single interpretation set; however, the sense graph could be composed of multiple interpretation sets and restrictions as shown in Figure 4.1.

A sequence of links (the end of one connecting the start of the next) is referred to as a *link path*. Link paths can be used to indicate the relation to a concept or graph and the surface graph. Most links are directed; they have a source and destination with distinct interpretations.

This research uses a number of different types of links.

[10]Dependency relations are limited to occurring within a graph.

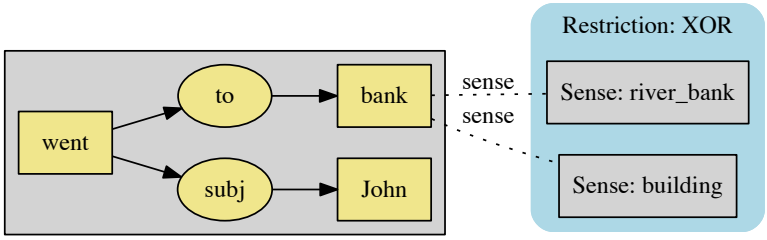


Figure 4.2: Example MIG: two word senses and interpretations: *John went to the bank*

Self This link type is used internally in link paths that only reference one concept (the start of a link path).

Coreference This link indicates a concept that is coreferent with another concept; this can be within the same graph or across graphs. These links are usually produced through anaphora resolution by the Stanford parser, but they can be created by any rule or process.

Sense This link indicates that the target graph is a possible sense for the source concept.

Genus This link indicates that the target concept is the genus term in an auto-frame, which should, in effect, make it coreferent with the source concept (that has been expanded and assigned to this sense).

Clausal This links a concept in a dependent clause (target) with a concept in the head clause (source). These links are only produced for some dependent clauses based on the output of the Stanford dependency parser.[11]

Example This links a concept to a list of examples of that concept, within the same graph. These are produced based on the analysis of the dependency parse tree.

TransferredConjunctionArgument These links are not presently used, but are intended to link arguments and adjuncts between concepts associated through a conjunction. For example, *Loud cats and dogs are annoying*, where loud can be applied to both cats and dogs; or *the friends drank and ate all night long*, where all night applies to both *drank* and *ate*.

[11]The parser connects relative noun clauses (rcmod), adverbial clauses (advcl), and some adjective clauses to a concept in the head clause.

Attachment This link connects an attachment point to something that may satisfy it; this is similar to filling a slot or assigning a semantic role. This is demonstrated in the following section on auto-frames.

Attachment Point This link connects an expanded concept to almost all concepts in the expansion, which are not the genus term. These links are actually just markers.[12] This link effectively indicates which concepts in a definition represent arguments in predicate-argument structure, or slots in a traditional frame representation.

Links, linkpaths, and MIGs are formally defined in the following box. The subsequent boxes defines some common operations on MIGs.

[12]This link type is theoretically not required. Even if this link type did not exist, these concepts would be attachment points, but this provides an easy mechanism to index, reference and find attachment points using linkpaths.

Let l be a link defined by a tuple $(lt, s, e, sense)$, where:
lt is the link type, which is one value from $\{Self, Coreference, Sense, Genus, Clausal, Example, Attachment, AttachmentPoint\}$,
s is the start of the link, which can be either a node (n) or a graph (g),
e is the end of the link, which can be either a node (n) or a graph (g),
$sense$ defines the sense of s, which is being selected. $sense$ is only used when $lt \in \{Sense, Genus, AttachmentPoint\}$, otherwise it is null.

Let a LinkPath $(linkpath)$ be defined as sequence of links $\{l_1, l_2, ..., l_i\}$, where $\forall_{j>1}$ $e_{l_{j-1}} = s_{l_j}$

An MIG is defined as a tuple (SG, EX, L, IS, IR) where,
SG is the set of surface graphs (graphs returned by the parser),
EX is a set of expansion graphs defined as a set of tuples $\{(linkpath_1, g_1), (linkpath_2, g_2), ..., (linkpath_n, g_n)\}$
EX can be expressed as a function $f(linkpath_i) = g_i$ defined by the set of EX
L is set of links, defined by the set of tuples $\{(linkpath_1, l_1), (linkpath_2, l_2), ..., (linkpath_n, l_n)\}$
Each tuple in L contains the $linkpath_i$ to the starting node and a link (l_i)
L can be expressed as a multi-map or function $subpaths(lp) = X$ where:
X is $\{(lp, l_1), (lp, l_2), ..., (lp, l_n)\}$,
IS is a set of interpretation sets,
IR is a set of interpretation restrictions.

Let us define some operations applicable to a **LinkPath**.

$last(linkpath) = l_n$ where $linkpath = \{l_1, l_2, ..., l_n\}$

$lastsense(linkpath) = sense$ where $last(linkpath) = (t, s, e, sense)$

$trunc(linkpath) = \{l_1, l_2, ..., l_{n-1}\}$ where $linkpath = \{l_1, l_2, ..., l_n\}$

$getSenseOR(linkpath)$ is a complex operation which finds the set $Expansions = (\{(linkpath, g_1), (linkpath, g_2), ..., (linkpath, g_n)\} \subset subpaths(linkpath) \subset EX)$ and then finds an Interpretation Restriction which joins them all together, as is created in the *expand* operation defined below.

We define two main operations on a MIG: $expand(mig, lp, g)$ and $validate(mig, IS)$.
$expand(mig, lp, g)$ expands an existing concept to a graph of the definition, sense, or auto-frame (lp is the linkpath to graph g, and $trunc(lp)$ should already exist in the mig).
$validate(mig, IS)$ returns a boolean value indicating whether the interpretation set given forms a valid interpretation.

Let $getIS(linkpath)$ take a linkpath and return the interpretation set which contains the end point. This function answers the question,"What interpretation set contains (the end point of) *linkpath*?"

$expand(mig, lp, g)$

Algorithm 4.1 Expand MIG Operation

```
# Add the expansion graph
EX_mig = EX_mig ∪ (lp, g)
L_mig = L_mig ∪ (trunc(lp), last(lp))
IS_mig = IS_mig ∪ IS_g # Add the interpretation sets
if lastsense(lp) ≠ null then
  # This should not happen. We would expand but not with a sense.
else
  # Add restrictions to the MIG
  IR_mig = IR_mig ∪ (IR_g \ entry_g)
  # Get an existing OR restriction for the concept being expanded
  # A restriction is a tuple of this structure: (is_1, IS_2, restriction)
  existingSenseOR = getSenseOR(trunc(lp))
  if existingSenseOR ≠ null then
    # Add the entry interpretation set to the OR restriction
    # Access the middle element of the tuple (indexing from 1)
    existingSenseOR[2] = existingSenseOR[2] ∪ entry_g
    # In a more object-oriented style:
    # existingSenseOR.IS = existingSenseOR.IS ∪ entry_g
  else
    # This is the first expansion of the concept.
    # Create a new OR restriction so that more senses can be added later
    newRestriction = (getIS(trunc(lp)), IS_entry_g, OR)
    IR_mig = IR_mig ∪ newRestriction
  end if
end if
```

4.4.3 Example

Let us consider the sentence "The teacher has a mustache". In *LDOCE*, there is only one sense for *teacher* and one for *mustache*. Figure 4.3 shows a representation for this sentence with both *teacher* and *mustache* expanded by their definitions (as they would be in MIG). The definitions represented in this figure are simplified for space and clarity. In this example, the definition of *teacher* and *mustache*, respectively, are "a human who teaches as a job"[13] and "hair that grows above the lip of a man".

[13] or "a human whose job it is to teach"

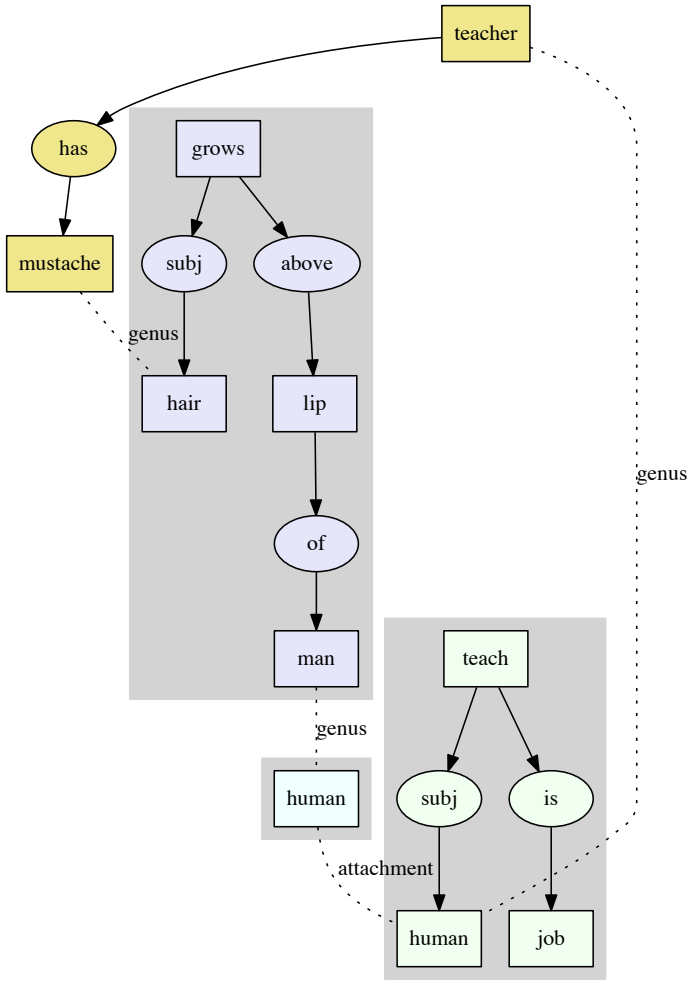


Figure 4.3: Example MIG with Expansion and Attachment: *The teacher has a mustache*

The dotted lines represent links which indicate that the concepts denote the same instance; in this example, they connect concepts from different statements. *Teacher* is connected with *human* by a genus link through the expansion of teacher to its auto-frame; both refer to the same instance, *i.e.*, the teacher is the human. *Mustache* is connected by a genus link with *hair*, through the expansion of mustache to its auto-frame. These links connect concepts from the surface graph to the concepts in the auto-frames. The attachment points of mustache are concepts in an expansions graph (*e.g.*, grows, lip, man human), which are not the genus concept, and concepts representing adjectives, adverbs, verbs, and nouns (*i.e.*, they are not conjunctions, determiners, articles, prepositions, etc).

Consider the surface relation *has*. It connects *teacher* to *mustache*, but our interpretation connects *hair* to *grows, lip, man* and finally, *human*. The surface-level relations (syntactic or dependency) only superficially capture how two things are connected. In auto-frames, all surface relations are mapped to some path which should include an attachment and an attachment point, and should hypothetically model our understanding of the relations. Furthermore, I believe that contextualization occurs below this surface level as we attempt to fit the interpretation to the information provided.

4.4.4 Summary of MIG

Section 4.4 defined MIG formally, descriptively, and with some examples. This section provides a quick descriptive summary.

Concepts and edges both store annotations. An edge connects two concepts with a dependency relation. A graph is composed of concepts, edges, interpretation restrictions, and interpretation sets. An interpretation restriction defines restrictions between interpretations sets. An interpretation set contains concepts, edges, graphs, and links. A link has a direction, a type, and connects concepts or graphs.

An MIG is a complex structure of graphs, links, interpretation sets, and interpretation restrictions. Any graph derived from the text being interpreted in an MIG is defined as the *surface graph*, while graphs added during expansion are defined as *expansion graphs*. Each graph may have one or more link paths associated with it, which indicate the path from the surface graph to this graph.

4.5 Auto-frames

4.5.1 Introduction

Each auto-frame is evoked by at least one lexical unit.[14] Auto-frames are stored in MIGs, thus auto-frames contain dependency graphs (derived from definitions or statements) and are inter-connected via links. Although these statements and definitions are in fact stored as graphs within an MIG, I will refer to them as statements, as this is what they are derived from.

Each sentence describes part of what is evoked by the auto-frame. I have chosen to use sentences instead of a global representation such as a semantic network, because sentences (individual propositions) can contain context-specific information. Furthermore, if a single statement is in contradiction with a given context, it is easy to remove that statement, instead of pruning a larger graph or network.

Some concepts in the auto-frame are marked as genus concepts,[15] these are the appropriate genus terms from the definitions in the auto-frame. As with definitions, these genus concepts form a taxonomy. An auto-frame may have more than one genus term, though in a given context at least one of these genus terms is expected to be connected and appropriate. This is important because both noun and verb definitions for a concept will be included in the same auto-frame, while their genus terms and concepts differ.

Each of the sentences in an auto-frame should be connected with other sentences by links.[16] For example, consider the auto-frame evoked by the word *bicycle*. Table 4.3 contains some of the definitions from *LDOCE*, which might be evoked by this frame. Some of the words or expressions are coloured to indicate where they share references with other senses. Neither the coloured references nor the list of sentences are exhaustive.

[14] word or expression

[15] Using the annotation systems that already exists for concepts in MIG.

[16] These are the same links already provided by MIG. Sentences within an auto-frame are expected to be connected since they are about the same idea.

Let $autoframe_i$ be an auto-frame defined by a tuple (LEX_i, mig_i, GC_i) where:
LEX_i is a set of lexical units which evoke this auto-frame,
mig_i is the MIG (knowledge representation) of this auto-frame containing all statements about this auto-frame,
GC_i is the set of genus concepts for this auto-frame.

Recall that mig_i contains a set of surface graphs, which we will refer to as SG_i.
SG_i is set of surface graphs for $autoframe_i$.
For all statements in the auto-frame,
there exists a graph g in SG_i which represent the statement.

$GC_i \subset$ concepts in $SG_i \subset$ concepts in mig

I have chosen to include these definitions in the example *bicycle* auto-frame for two reasons: they refer to bicycles or bicycle's definition refers to them. Most of the definitions contain the word *bicycle*, and thus relate directly to a bicycle. Other definitions, such as those for *vehicle*, *go*, *travel*, define words that are important to understanding a bicycle; that is to say, these words appear in the definition of bicycle or in definitions that use the word *bicycle* (for example see the definitions of *ride, pedal, vehicle*, and *go/travel*).

Let us informally consider how some of these definitions may be related. The first sense of *bicycle* refers to the physical object, while the second sense refers to the primary use or activity associated with that object. *Riding* generally refers to the activity done on a bicycle, thus is also related to the second sense of *bicycle*. *Pedaling* is an activity which enables or powers the riding; this relationship is actually the least clear from these definitions. *Crossbar* defines a part of a bicycle, and the definition also states that a bicycle has a seat and handlebars.

Table 4.3 is only intended to be an example to demonstrate the idea of an auto-frame within this thesis. Note that the red, violet, blue and green terms are the genus concepts in this auto-frame. If we expand *travel* by adding its definition, we find travel's genus term is *go* which would connect the violet and green links into a single chain. In addition, *journey* is defined using *travel*, indicating that all these terms have some commonality.

In the example sentences above, links are not explicitly defined, but the coloured words suggest where links exist. As an example, in the first definition (*bicycle*), *vehicle* would have links to the definitions of *machine* and *vehicle*. This means that *vehicle* from

Word	Definition
Bicycle	a two-wheeled vehicle that you ride by pushing its pedals with your feet
Bicycle	to go somewhere by bicycle
Crossbar	the metal bar between the seat and the handlebars on a man's bicycle
Cyclist	someone who rides a bicycle
Pedal	to ride a bicycle
Pedal	to turn or push the pedals on a bicycle or other machine
Ride	to travel on a bicycle or motorbike
Ride	a journey on a horse or bicycle, or in a vehicle
Vehicle	a thing such as a car, bus etc that is used for carrying people or things from one place to another
Bicycle	bike
Bike	a bicycle or motorcycle
Machine	a vehicle

Table 4.3: Examples of Sentences Expected to be in the Bicycle Auto-frame from *LDOCE*

sentence 1 will have a genus link with *machine* and *thing*. *Ride* from the first definition is expanded to the definition of *ride*, and thus there are genus links to *travel* and *journey*. *Cyclist* would have coreference links with the first definition: *someone* would link with *you*, *rides* with *ride*, and *bicycle* with *vehicle*. *Ride* from the definition of *Cyclist* could be connected via a coreference link to the definition of *ride* or other instances, but it is not necessary since connecting to the first definition (*bicycle*) connects with an entire chain of links that connect to these other instances of *ride*.

Let us consider how auto-frames will relate to homographs, polysemy, and polylexy. Homographs are words with the same spelling, but completely different or unrelated meanings. For example consider the word *bank*; the meanings, the side of a river and a financial institution, are completely distinct. These two meanings for the word *bank* are

homographs. Homographs will require distinct frames.

Polysemy refers to a word having more than one sense, though these senses should be related. They may be derived from or related to the same root (etymology), or have some strong parallel or metaphor. Again consider the word *bank*, the polysemous meanings found in Table 4.4. Examples 2-4 all refer back to example 1, thus they have a clear relationship to the first sense. Example 5 might be considered to have a weak relation to example 1, as the concepts have parallels; in both senses, the word means a place to store and withdraw money, but the contexts are different. Polysemous words should be contained in the same auto-frame.

Example	Definition
1	a business that keeps and lends money and provides other financial services
2	a local office of a bank
3	to put or keep money in a bank
4	to keep your money in a particular bank
5	a supply of money used to gamble, that people can win

Table 4.4: Polysemous Examples of *Bank* from *LDOCE*

Polylexy refers to a polysemous word requiring a distinct lexical entry. The examples above have distinct lexical entries, although the question remains: do they require distinct entries? Some polysemous senses may not be polylexic, as the alternative sense can be derived from another sense. While examples 2-3 have been given distinct lexical entries in *LDOCE*, these entries may not be needed. In particular, it would seem likely that when *bank* is being used as a location, example 2 could automatically have been inferred from example 1. Likewise, example 3 and 4 are similar enough that one of them may be unnecessary.

Auto-frames, Context, and Entailment

Auto-frames, Context, and Entailment

Consider what is evoked by a word or what are the expected conditions.[17] Consider the words *bribe, tip, alimony,* and *paid*. All the words evoke a transfer of money from one

[17]These expected conditions are likely found in the word's definition, which is the only source of information that this research considers. One advantage of using *LDOCE* over other dictionaries is that some of this information is included in the definitions (*e.g.*, *bribe* suggests dishonest activity - "... by doing something dishonest").

person to another, but each evokes different conditions. *Bribe* suggests a criminal/illegal context in exchange for a favor. *Tip* suggests that good service was already received. *Paid* suggests that the money was used in an equivalent exchange for a good or a service. *Alimony* suggests divorce between two people, and the money is to support one of them.

All these words are linked because they involve an exchange of money, but the evoked context is different. Yet the evoked context is not always accurate, as writers or speakers will try to evoke a context more suited to their goal; whether for juxtaposition, humor, hyperbole, or to manipulate/temper a reader's reaction. For example, in the following sentences, all suggest the same circumstance, even though the words are different, but some might seem more or less criminal.

The student bribed the teacher for a good mark.
The student paid the teacher for a good mark.
The student tipped the teacher for his good mark.
The student gave money to the teacher for a good mark.

Each of the examples above will evoke its own distinct auto-frame (or auto-frames). The commonalities and differences between these auto-frames will be apparent in their shared sentences (senses) and the senses which they uniquely contain. Entailment, or graph comparison, can find the commonalities, while the differences in the auto-frames would help us understand the differences between the contexts. *Bribed, paid,* and *tipped* all suggest giving of money (this should be common between the auto-frames); however, *bribed* suggests it is criminal and for unearned favor, *paid* suggests a legal exchange of money for a good mark, and *tipped* suggests it was given in appreciation of the good mark. The circumstance where a speaker or writer uses an inaccurate word to change or manipulate the evoked context is beyond the scope this thesis, though it does relate to contextualization.

FrameNet does not provide frames for most of these words, but often the frames are very general. An example is the *cook* frame. It is also used to represent *fry, grill, steam, boil, steam, toast* and more, even though all of these forms of cooking (or heating foods) have their own unique distinctions.

As another example of the generality of frames consider the word *bank* as a financial institution. This evokes the frame *Businesses*, which is also evoked by words such as *chain*, *mill*, *shop*, *firm* and more. The definition of the *FrameNet* frame is very general "A Proprietor owns or runs a Business which provides a Product (which may be goods or services." *LDOCE*, on the other hand, has definitions specific to *banks*; see Table 4.5.

Another common sense of *bank* is river bank (definition 12). *FrameNet* has a lexical

Number	Definition
1	a local office of a bank
2	to put or keep money in a bank
3	to keep your money in a particular bank
4	to arrange something into a pile or into rows
5	a business that keeps and lends money and provides other financial services
6	the money in a gambling game that people can win
7	a slope made at a bend in a road or racetrack to make it safer for cars to go around
8	to make a plane, motorcycle , or car slope to one side when turning
9	to cover a fire with wood, coal etc. to keep it going for a long time
10	to depend on something happening or someone doing something
11	a large pile of earth, sand, snow etc.
12	land along the side of a river or lake

Table 4.5: Senses for Bank in *LDOCE*

unit with a related frame, *Relational_natural_features*, which is also evoked by *summit*, *shore*, *footholl*, and more. The definition of this frame is "The Focal_feature is defined in relation to a Landmark_feature, either as a particular part, or as an immediately bordering entity."

The fact that many of *FrameNet* frames are general is both a strength and a weakness. A number of tasks around message and opinion understanding (Gangemi et al., 2014; Mohammad et al., 2014a; Kim and Hovy, 2006) can benefit from a simple general representation. Such tasks often find and label key elements and then further process those key elements. In addition, a simple general representation is easier to compare than a complex representation which requires proofs and automated reasoning.

However, tasks which aim to make inference or understand implicit relationships may require more specific knowledge about each word. *FrameNet*'s representation allows for easy uniform comparison of concepts. With auto-frames– since each term will have a distinct unique auto-frame– such comparison will be achieved through a loose form of entailment. For example, terms such as *fry, grill* and *steam* all evoke (include in their auto-frames) the definition of *cook*.

Auto-frames and Personal Experiences

Semantic frames (Section 3.8) are an idea about how words and human memories are associated. A word evokes a particular frame; that frame contains information about the meaning of the word, its associations, implications, default values, and past circumstances. Some of what is evoked is personal; it could be slightly different for each person. Semantic frames are somewhat related to the idea of semantic priming. As a frame is evoked, it is also priming (preparing) someone for a number of associated ideas. There is an expectation that concepts normally related to the frame should also be present. Ideas that have been primed are easier or faster to access (in someone's mind) than other ideas.

Auto-frames, within the scope computer software, lack personal memories and experiences. Future research might attempt to simulate these experiences through expanding frames with information from corpora (particularly stories). Auto-frames will only contain the information available in a dictionary; this should include the word's meaning, some associations, defaults, and possibly some implications. The fast access or expectation of primed concepts is paralleled by an auto-frame including expected and related information, which, if not relevant, must be pruned away afterwards.

4.5.2 Auto-frames and Roles (Attachment Points)

An auto-frame is a set of interconnected statements. These statements have links between them, connecting concepts which refer to the same instance. Some concepts are marked as genus concepts. All the concepts in the auto-frame, except those functioning as genus concepts, function as attachment points.

Attachment points are similar to slots, roles, or arguments, as they define what can be and is expected to be connected with an auto-frame. Furthermore, each attachment point acts as a selectional restriction. When an attachment point has a list of examples within a definition, these examples can be treated as default or likely values. Consider part of the definition of *ride*: *"to sit on an animal, especially a horse"*. *Horse* is a default or likely value for the attachment point *animal*. Assigning an *attachment link* between an attachment point and an adjacent auto-frame (when interpreting a sentence) is the same as filling a slot or assigning a role.

Defining auto-frames in this manner depends on the hypothesis that the roles (attachment points) appear in the definition of each concept, which, as seen in section 3.5.1, is not unreasonable, although perhaps not all roles are present in every dictionary. Furthermore, a number of roles were shown not to be defined in a definition but in related

definitions. An earlier example was how the *price* role for the word *buy* is defined in the definition of *price*, not *buy*.

4.5.3 Attachment Example

Previously, the example "The teacher has a mustache" (Section 4.4.3 and Figure 4.3) demonstrated attachments. However, let us consider the attachment points for a different auto-frame, *cook*, which is shown in Table 4.6.

Again, I will select a subset of definitions that may be included in the *cook* auto-frame. The sentences are presented in Table 4.6 using a similar notation: coloured words indicate linked concepts. These links connect different sentences. Again, not all links will be shown because they are too numerous.

Word	Definition
cook	to prepare food for eating by using heat
cook	to make a meal quickly, often using food that has been left from a previous meal
cook	someone who prepares and cooks food as their job
food	things that people and animals eat, such as vegetables or meat
meal	the food that you eat on a particular occasion
meal	an occasion when you eat food, for example breakfast or lunch
heat	to heat food thoroughly
heat	to make something become warm or hot
oven	a thing inside which food is cooked, shaped like a box with a door on the front
cooker	a large piece of equipment for cooking food on or in
eat	to have a meal
eat	to put food in your mouth and swallow it
kitchen	the room where you prepare and cook food
temperature	a measure of how hot or cold a place or thing is
boil	to cook something in boiling water

Table 4.6: Example of the Cook Auto-frame derived from *LDOCE*

The genus concepts for this auto-frame are *prepare* in the first sentence, *make* in the second sentence, and *someone* in the third sentence. The attachment points are the

non-genus concepts. In fact, when a genus concept does not act as the genus concept of the auto-frame in a given context, it is an attachment point. Recall that an auto-frame is made up of several definitions, and each definition has a genus concept. Some of these genus concepts may be genus concepts for the auto-frame. For example, the bike auto-frame would have two genus concepts, *ride* and *vehicle*. In a given context, the usage of the word *bike* will only directly denote one of the two genus concepts. Consider "John biked to the store." Here, *biked* would refer to ride, while vehicle is now functioning as an attachment point.

Again consider the sentence "The cook prepared an excellent meal." Here *cook*, referring to a person, would attach to *someone* with a genus link; *prepare* in the *cook* auto-frame can (in this context) be an attachment point, and will be attached to *prepare* in the original sentence, but in other contexts *prepare* could be genus term and *someone* the attachment point.

Linked words in red, violet, blue, brown, orange and green are attachment points, as are any unlinked words, such as *oven*, *breakfast*, *occasion*, *swallow*, *boil*, etc. The colours are assigned so that they match with the colour scheme in Table 4.6, that is to say, the coloured words in the sentence would be attached into the auto-frame described above at the attachment points with matching colours. For example, dinner (brown) would be expanded to and linked to meal, cooking (red) to the definition of cook, and cake (blue) to food.

It may seem strange that unlinked words can also be attachment points. Unlinked words are simply not connected with other concepts in the auto-frame; however, they can still connect with other adjacent auto-frames.

There are many more attachments than traditional semantic roles. Attachment points are not intended to only connect syntactic roles or dependency relations; instead they provide a mechanism to connect distant (not immediately connected in a dependency tree) concepts. Given this example consider the following sentences and how a number of the words are connected to the cooking auto-frame.

> When everyone arrived for dinner, the kitchen was warm from cooking all day. Everyone enjoyed dinner but the cake was the highlight.

Consider the coloured words, which are all related to one another. As readers, we can already see how the ideas are connected and assume that people ate the cake, which is not mentioned, and that the cake was at least one of the things cooked. All of these concepts are related but are distant (in different clauses and sentences). While this

example demonstrates an interesting goal and use of auto-frames, this is a very difficult task that has barely been touched on in the implementation.

One may note that there are almost no attachment points for adjuncts phrases; this is a significant limitation in this work, but in theory they may be supported by processing the definitions of prepositions. In *LDOCE* there are definitions for a number of prepositions and these define the semantic relations of a prepositional phrase to head word. Future work should use these definitions to make meaningful attachments between adjunct phrases and auto-frames.

4.6 Contextualization

4.6.1 Introduction

Contextualization is the process of adapting an auto-frame to a context; this process will contain the following steps: modulation, loosening and filtering.

Modulation is the process of selecting or adapting senses appropriate for the context. *Loosening* is attaching a concept to an attachment point, which is more general than the attachment point point expects (*i.e.*, it does not satisfy the selection restriction, but it belongs to a superclass). *Filtering* is a non-monotonic process of removing information from an auto-frame that is incompatible with or irrelevant to the context. Filtering will specifically remove defaults, implications, and assumptions that are not connected with the context.

Modulating an auto-frame for a context can be achieved in two ways, depending on whether the auto-frame includes multiple senses for the same word or not. If the required sense already exists in the auto-frame, then modulating a frame is simply selecting the appropriate sense for the context from the auto-frame. This is a form of fine-grained WSD. But unlike traditional WSD, it would be adequate to return all fine-grained senses that could be correct modulations, instead of selecting only one sense. While there may be many measures of appropriateness of sense in context, this work focuses on how attached[18] a sense or auto-frame is to other concepts.

The second form of modulation is changing genus terms. This is intended to allow a sense to change its part of speech, or just change its emphasis, similar to the *sense-views* in (Rais-Ghasem and Corriveau, 1998). Consider our example of a *bicycle*, which is a

[18]By *attached*, I am referring to whether or not the auto-frame (for a given sense) is attached, by links in the MIG, to the context.

machine, *vehicle*, and a *physical object*. A *sense-view* emphasizes particular properties of a concept, such as those related to a *machine*, a *vehicle*, and a *physical object*. "Oiling a bicycle" emphasizes *bicycle* as a *machine*. A context may be emphasizing a particular *sense-view*.

With auto-frames and MIG, this emphasis is determined by the link path attaching one surface concept to another (passing through certain genus concepts and attachment points). This will be illustrated in section 4.6.2 and described in the following paragraphs.

Because an auto-frame contains multiple sentences, it is expected that only some of them will be connected in the context. If there are no attachments, as can occur at times, then all senses must be included.

At times creating an attachment requires connecting through more general or more specific concepts. This is where loosening occurs, when a concept is more general than the attachment point to which it is attached. For example, *cook* can be attached to an attachment point of *boil*, a more general concept.

Likewise, an attachment point may require something like a *vehicle* or a *machine* and be attached to a *bicycle*; *bicycle* is a subclass of *vehicle* and *machine*. When a *bicycle* is treated as *vehicle* (*i.e.*, a mode of transportation), or as a *machine*, it provides emphasis for certain aspects of the concept: how the concept is being considered in this context. This is similar to the *sense-view* in (Rais-Ghasem and Corriveau, 1998). The implementation of this is defined in the following chapter.

If a sense is not polylexic, then an appropriate sense may need to be generated or inferred. Generating modulated senses of a word should be easy when the context requires a common semantic relation, such as *location*, *activity*, *instrument*, and *manner*. Consider example 1 from the definitions of *bank* in Table 4.4. An activity associated with banks is given *"keeps and lends money"*. Consider that example 2 might be inferred from example 1 by knowing that a bank is *a business* and knowing the definition of *office*: *the building that belongs to a company or organization, with a lot of rooms where people work (office, LDOCE).* It is likely that the business (bank) is operating out of an office. Likewise, example 3 could easily be inferred given example 1, even though they differ (verb and noun respectively). In this thesis, we will treat this type of modulation as selecting a different genus term from a definition.

Most of the examples provided in this section already had multiple senses and definitions, thus this second method of modulation is less relevant. This second technique might also be of value in creating senses for morphological variations of a word. Frequently, a word undergoes a predictable transformation in meaning when it is used as

Example	Definition
1	We went to Toronto in the car.
2	I repaired the car.

Table 4.7: Examples of Filtering Auto-frames *LDOCE*

a different part of speech or with a different suffix. Consider how the proper noun "Google", the company that operates a search engine, has morphed into a verb, *google*,[19] meaning to search the Internet using Google. This transformation might be predicted and automatically generated without the need to update a lexical knowledge base. This thesis does not intend to focus on this issue.

When interpreting a sentence represented in MIG, we will expand surface concepts with auto-frames and make attachments between them. Once the interpretation process is over (all auto-frames added, all attachments made and all sentences processed) a filter process can occur. This will remove any unattached auto-frames and any unattached sentences from an auto-frame that is partially attached. The hypothesis is that unattached sentences contain implications, assumptions, or facts that are not relevant or true for this context, thus the goal of this step is to prune such information.

Consider the examples found in Table 4.7. The car auto-frame would include the definitions for *driving* and *traveling*, as well as a definition for *machine*. The first sentence should attach *went* to *driving* and *traveling* leading to the understanding that the car was driven to Toronto. The definition of *machine* could be filtered from the auto-frame in the first example. The second sentence would not connect with the definitions for *drive* and *travel*, but *repair* might connect with *machine*. The definitions of *drive* and *travel* could be filtered out of the auto-frame in the second example.

4.6.2 Contextualization Examples

The example previously shown in Figure 4.3 contains a form of modulation as defined by Rais-Ghasem and Corriveau (1998). The concept *teacher* is modulated to a male sense, because mustache has a selectional restriction of male.

If the *teacher* was known to be female and the knowledge system considered male and female incompatible types[20] then the interpretation above would be inconsistent for most knowledge systems. Let us assume that an interpretation system finds a superclass

[19]The verb "google" is used in English both with and without the first G capitalized.

[20]This restriction or incompatibility is not present or extractable from most dictionary definitions, though some dictionaries contain additional annotations or information of this sort.

of *male* and *female* compatible, say *human*. To contextualize this example and make it consistent, we must remove the requirement that the mustache belongs a man and instead accept its superclass *human*, at least for this specific context. Because we have expanded the definition into this sentence, it is possible to edit or alter the interpretation of *mustache* just in this context. This is another form of contextualization: the original example changed or restricted the understanding of the *teacher*. This contextualization loosens our understanding of the definition of mustache, or at least it loosens the selectional restriction.

Both of these forms of contextualization can easily be identified by the path required to make an attachment. In Figure 4.3 (p. 54), the surface concepts *teacher* and *mustache* are directly connected by surface dependency relation (*has*); however, to create an interpretation an attachment must be made between senses, which should represent the semantic relation. We can see that the attachment of *teacher* to *mustache* takes a path through *man* which functions as selection restriction. The distinction between these two forms of contextualization is whether the original types are compatible or not. This work does not consider subtypes of the same supertype as incompatible (mutually exclusive, such male and female), because there is no simple way to infer compatibility/incompatibility from the definitions in *LDOCE*.

Another form of contextualization has to do with emphasis of properties of the concept, as was seen in section 4.6.1 and in Rais-Ghasem and Corriveau's (1998) work. Consider the following sentences about bicycles, similar to those from (Kilgarriff, 1997).

Example	Sentence
1	Boris oiled his bicycle.
2	Boris cleaned his bicycle.
3	Boris rode his bicycle.

Table 4.8: Example modulations of *bicycle*

Each sentence refers to the same sense of *bicycle* but the verb modulates how each *bicycle* is considered. Example 1 focuses on the bicycle as a machine. Example 2 focuses on the bicycle as an object, particularly one that was dirty or can be dirty. Example 3 focuses on the bicycle as a vehicle, a mode of transportation. The emphasis of the concept is easily identifiable from the link path attaching the verb and bicycle.

MIGs for each sentence are shown in Figure 4.4. For clarity, I added only one in-

terpretation and most of the definitions have been simplified. Example 3 simplifies the expansion of bicycle and vehicle to just the genus terms. The yellow nodes represent the surface-level graph.

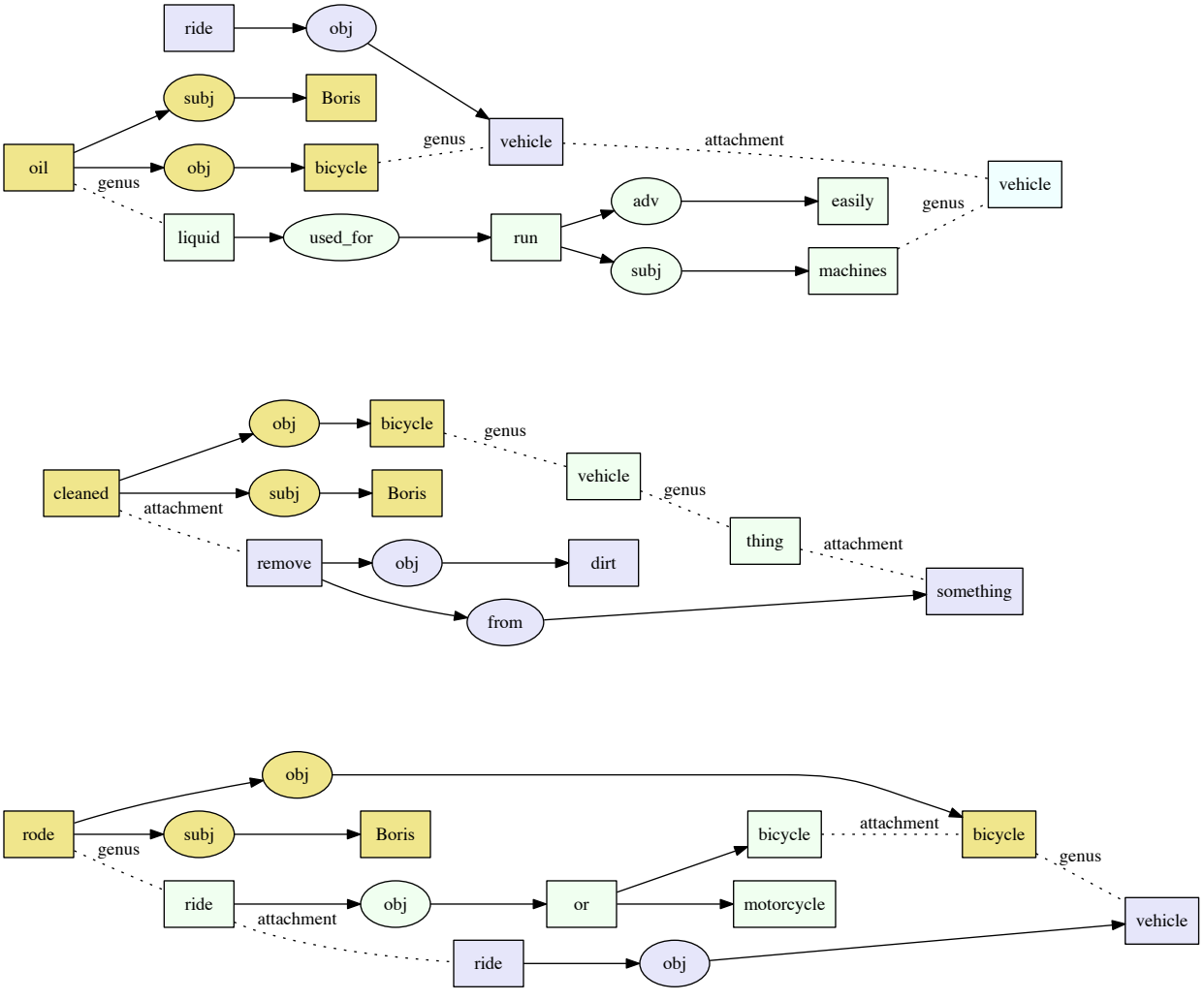


Figure 4.4: Examples of contextualized sentences

The attachment between the verb and *bicycle* connects to different attachment points in each sentence. Recall that the path that includes the attachment should capture the semantics of the surface relation, that is to say, why the two things are related.[21] The path and attachment also indicate the emphasis.

[21] An example is provided in Section 4.4.3.

In the first example, the word *oil* from the surface graph has been expanded by the definition "a liquid used for making machines run easily". A genus link has been created between *oil* and *liquid*. The word *machines* in this expansion has been further expanded to *vehicle*.[22] The word *bicycle* from the surface graph has been expanded by the definition "a vehicle that you ride". A genus link has been created between *bicycle* and *vehicle*. An attachment is made between these two adjacent expansions. The attachment path runs through *machine* and attaches *vehicle*, thus the emphasis is on a *bicycle* as a *vehicle* and *machine*.

Recall that the process of interpretation would have brought the whole of the auto-frames for the senses of *bicycle* and *oil* into the MIG. The last step in the interpretation process filters unattached sentences. The remaining graphs (after filtering) would only include this small selection of attached sentences. These graphs are the result of the contextualized auto-frame.

In the second example, the word *cleaned* from the surface graph has been expanded by the definition "to remove dirt from something". A genus link has been created between *clean* and *remove*. The word *bicycle* from the surface graph has been expanded by the definition "a vehicle" and that has been further expanded to *thing*.[23] A genus link has been created between *bicycle* and *vehicle*, and another between *vehicle* and *thing*. An attachment is made between these two adjacent expansions. The attachment path runs through *vehicle*, then *thing* attaching *something*, thus the emphasis is on a *bicycle* as a *vehicle* and *thing*.

In the third example, the word *rode* from the surface graph has been expanded by the definition "to ride a bicycle or motorcycle". A genus link has been created between *ride* and *ride*. *Bicycle* has been expanded by the definition "a vehicle that you ride". A genus link has been created between *bicycle* and *vehicle*. Two attachments exist, one connecting *bicycle* to *bicycle* and another connecting *ride* to *ride*.

4.7 Comparison with FrameNet

FrameNet (Baker et al., 1998) is an inventory of frames based on frame semantics (as is this work), and is manually produced from corpus evidence. Each frame is given a brief textual description and a list of frame elements, occasionally with semantic type restrictions or default values. Each frame may be evoked by a variety of words, thus

[22] One of the definitions of machine in *LDOCE* in simply the word *vehicle*.

[23] I have limited these definitions to single word, the genus concept, to keep the graphs compact.

creating a uniform representation for many closely related or similar words. Furthermore, there exists a taxonomy of frames, associating frames and providing mappings between their respective frame elements. One of *FrameNet*'s greatest benefits is that it contains an annotated corpus of evidence for each frame. Systems which automatically annotate frames in a corpus can be trained on the annotated data. These systems do not learn new frames, but annotate existing frames in context. The SemEval 2007 Task 19: Frame Semantic Structure Extraction (Baker et al., 2007) evaluated many of these systems.

While this research is founded on the same idea as *FrameNet* (*frame semantics*), there are a number of differences. This work aims to generate auto-frames automatically from resources, while *FrameNet* is manually generated. There have been recent attempts to automatically extend FrameNet. The most relevant was the induction of new frames from text (Cheung et al., 2013). Many others have tried to learn and add new lexical units to existing frames (Pennacchiotti et al., 2008).

This work treats an auto-frame as a collection of inter-connected statements, while *FrameNet* defines a frame with a textual description and provides mappings between frames. The *FrameNet* mappings between frames (senses) are not sufficient to define a frame in terms of other frames, but they provide some of the details. The mappings (when present) provide inheritance (superclass) relationships, sub-events, and role mappings, but they do not define when frame elements are partially or completely filled by a definition. For example, the frame for *cook* (cooking_creation) inherits from the Intentionally_create frame, but the mapping between *cook* and Intentionally_create do not place restrictions on the components or created_entity frame elements, nor do the frame elements ingredients or produced_food in the *cook* frame have any selectional restriction. It should be noted that the human-readable frame element names do make some implications about ingredients and food, but these names encoded for readers not offer structured semantics for a computer system. It should be noted that the cooking_creation frame also uses the apply_heat frame. A definition derived from these mapping indicates that cooking is nothing more than producing something by applying heat.

This work aims to include common-sense information (to the extent found in dictionaries), while *FrameNet* is limited to some semantic restrictions and default values. *FrameNet* frame elements can have semantic type restrictions, though they are not present for very many frame elements. Furthermore, since *FrameNet* does not provide structured definitions (it provides a textual description of the frame), there is no mechanism to encode common sense information. Frame elements are designed around roles realized within a sentence or clause containing the frame based on corpus evidence

(Johnson et al., 2002), while auto-frames aim to support more distant connections.

FrameNet provides a uniform representation; a single frame is evoked by a variety of words (*e.g.*, the *Apply_heat* frame is evoked by *toast, stew, steep, simmer, grill, deep fry, cook*, etc.). This research will produce different frames for each of these words, though they may contain some of the same statements. *FrameNet*'s uniform representation makes comparison of similar statements easier than auto-frames. This research aims to support similar comparisons, through comparison of what is evoked by an auto-frame. *FrameNet* provides a clear list of frame elements for each frame, while this research does not intend to identify or focus on frame elements, though they effectively exist as attachment points, as was described in Section 4.4.2. *FrameNet* has frame elements for adjunct phrases, while auto-frames do not at this time include adjunct phrases, though it possible to add them.

4.8 Summary

This chapter introduces my ideas about how to build and implement soft definitions. We saw how MIGs are composed of graphs, links, interpretation sets and restrictions. An MIG can hold multiple interpretations because it can contain many different sets of assigned senses and attachments. The validity of an interpretation depends on the satisfaction interpretation restrictions. Auto-frames are definitions inter-connected by links and represented in MIG. Auto-frames contain many definitions, but not all are true or valid in all circumstances. Contextualization is the process of selecting and attaching definitions, then pruning disconnected or unrelated interpretations (senses and attachments) from the MIG. The following chapter describes the implementation of this theory.

Chapter 5

Methodology

This chapter describes the data structures and algorithms relating to Multiple Interpretation Graphs (MIGs) and to constructing auto-frames from *Longman Dictionary of Contemporary English* (*LDOCE*).

The chapter is broken into three major sections: building a lexical knowledge base, connection search, and using auto-frames. *Building the lexical knowledge* focuses on parsing and pre-processing the dictionary, as well classifying and annotating the definitions. *Connection search* is the name of the main algorithm used in the construction and application of auto-frames. The end of the connection search section describes how senses are grouped into auto-frames and what additional information is included in auto-frames. *Using auto-frames* focuses on a methodology for applying auto-frames to the task of reading comprehension.

5.1 Building a Lexical Knowledge Base

The following list outlines the high-level steps taken to construct auto-frames. Each step is described in detail in a later section.

1. Parse definition (Section 5.1.1)
2. Transform graphs (Section 5.1.2)
3. Mark type and genus term of each definition (Section 5.1.3)
4. Assign simple high-confidence sense labels (Section 5.1.4)
5. Connection search (for disambiguation of dictionary senses) (Section 5.2)
6. Construct auto-frames (Section 5.2.4)
   (a) Group senses into auto-frames

(b) Apply connection search to expand auto-frames

5.1.1 Parsing

The Stanford Pipeline (CoreNLP) (de Marneffe et al., 2006; de Marneffe and Manning, 2008) was chosen for parsing because of its availability and its acceptance in the Natural Language Processing (NLP) community. It has many useful features built in, such as part of speech (POS) tagging, sentence splitting, dependency parsing, Named Entity Recognition (NER), and anaphora (or co-reference) resolution. In addition, the pipeline is well designed and can easily be extended.

The output of the parser is a dependency tree, which is an effective choice converting text to a knowledge representation (Bayer et al., 2004; Ahn et al., 2004; Anthony and Patrick, 2004; Baker et al., 2007; Das et al., 2014). Dependency parse trees are a convenient starting representation, because they contain a one-to-one mapping of terminals (tokens) to nodes. A constituency tree usually contains many non-terminal nodes, which would require significant restructuring to obtain a Conceptual Graph (CG) or even predicate-argument structures.

CoreNLP had noticeably more accurate dependency trees than the older version of the Stanford parser used by Kim et al. (2010) with Packed Graphical Representation (PGR). The CoreNLP parser does well with properly structured sentences, but as sentences becomes more ambiguous, incorrect, or include additional information, it produces less accurate dependency trees and has longer runtimes.

There were a few challenges related to parsing the definitions: getting accurate parse trees, working around problems with the parser, and managing its runtime. The problems with getting accurate parse trees were usually related to the format of the definitions; the solutions to these problems are described in the following sections. Problems related the runtime and errors with the parser required an engineering solution, which is described in Section 6.3.

Difficulties Parsing Definitions

Table 5.1 contains a sample of definitions which the parser has trouble parsing correctly. This section describes how these issues were mitigated. Note that these solutions are only applied in the context of parsing definitions.

Not all definitions are proper sentences and this can lead to the parser failing to find the correct dependency tree. Consider example 1 in Table 5.1 where the definition starts

Example	Word	Definition
1	$\text{bicycle}_{\text{noun}}$	a two-wheeled vehicle that you ride by pushing its pedals (pedal) with your feet
2	swim	to move yourself through water using your arms, legs etc
3	basketball	a game played indoors between two teams of five players, in which each team tries to win points by throwing a ball through a net, or the ball used in this game

Table 5.1: Examples of Difficult Definitions to Parse from *LDOCE*

with a noun phrase, which has a subordinate clause; the parser does not know how to attach this sub-clause to the noun phrase. However, if we simply prefix the definition with "X is " the definition becomes a sentence the parser can handle. This prefix is not correct for all definitions, as some are already valid sentences.

For every definition, at least two texts are sent to the parser: one with a prefix and one without. The prefixes used are, "X is a","X is to", or "X is", depending on the first word in the definition, and whether the definition is for a noun or verb. Note that X represents the word being defined, and will be referred to as the dummy concept. Once the parser has returned the parse trees, the tree with the fewest signs of parsing problems or ambiguous dependencies (such as dep and appos) is selected. I assume that having fewer of those dependencies implies more accurate parse trees. Subsection *Selecting Parse Trees* defines how the selection of parse trees is done.

If the text required a prefix to obtain a good parse tree, then the concepts from the prefix are removed. If the dummy concept was the subject of a concept (usually a verb), then that concept is marked as having had the dummy concept before removing the dummy concept. If the "is" concept from the prefix was considered the root concept, then the root annotation is moved to its direct object.

Figures 5.1 and 5.2 illustrate two sample parse trees with prefixes for the sentences "X is a dog" and "X is to run very fast". In both circumstances, the "X" and "be" nodes will be removed. In Figure 5.1, "run" becomes the new root node, while in Figure 5.2, the root node remains "dog".

The second type of error is caused by parenthetical material or other notations that are not part of the main sentence, as in example 1. In most of these cases, simply removing the parenthesis allows the parser to correctly process the sentence, though in

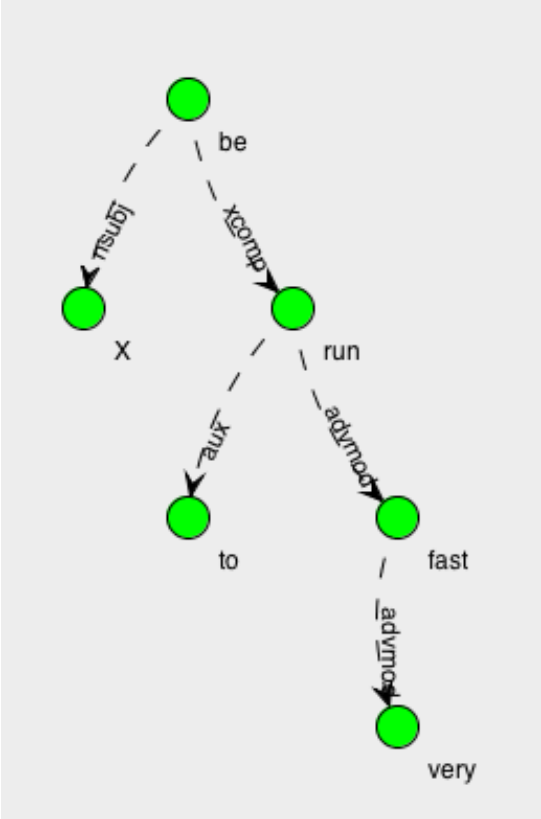


Figure 5.1: Example parse tree with prefix: "X is to run very fast"

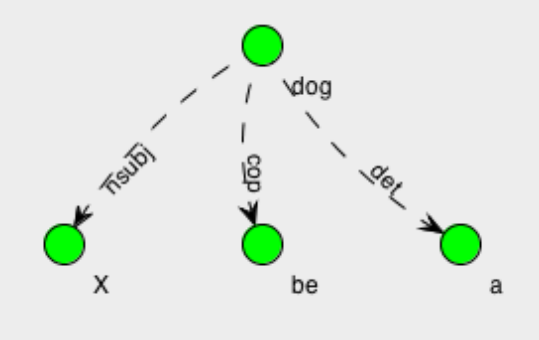


Figure 5.2: Example parse tree with prefix: "X is a dog"

a few circumstances the text in the parenthesis is required. If a definition contains parentheses then the number of texts sent to the parser is at least doubled: as all combinations of text with and without parentheses and with and without prefixes are processed.

The third error is caused by the presence of "etc" in a sentence, which is often used to indicate an incomplete list. Most of these lists are of examples given within a definition. The presence of the "etc" confuses the Stanford parser and the resulting parse trees are incorrect. If the "etc" is removed, then the parser returns the expected results. Slightly better results may be obtained by inserting a conjunction ("or") between the last and second last examples in the list. I created an annotator, which is added into the Stanford Pipeline that removes the "etc" tokens but leaves an annotation indicating that they existed. Using this annotator, the parser returns the correct parse trees and all the information is kept (as an annotation marking the removed "etc" token).

The fourth type of error comes from extensive use of commas in the *LDOCE* defini-

tions, which occasionally leads to ambiguity about where phrases and clauses attach. In example 3 the dependency incorrectly connects "or the ball" with "two teams".

Selecting Parse Trees

After parsing a definition and getting multiple parse trees, the trees are ordered by:

- which has the fewest "dep" relations (an ambiguous dependency relation used to connect concepts when the structure is not clear).
- which trees have the fewest "appos" relations (indicating appositional modifier). The "appos" relation is sometimes indicative of a word or phrase that could not be correctly connected.
- which had parentheses removed and which did not require prefixes.

The first dependency tree from this ordering is selected as the parse for this definition.

5.1.2 Initial Transformations

The purpose of the initial transformations is to restructure and annotate content to assist later processes.

The transformations done during this step are as follows:

1. Relabel Participle Adjectives
2. Mark Clause Heads
3. Restructure Conjunctions
4. Combine Multi-node/word Expressions

Relabel Participle Adjectives

Participle adjectives are a special form of some verbs, which bear inflection but function as adjectives. The POS tagger in CoreNLP will label these tokens as `VBN` for the past participle (-ed suffix), and `VBG` for the present participle (-ing suffix). For example, *abandoned* or *amazing* in the follow sentences function as adjectives:

The abandoned warehouse was a blemish on the skyline.
The amazing magician was delightfully entertaining.

While the POS tagger generically labels these as inflected verbs, their dependency relation indicates their function. Dictionaries will contain specific definitions for the adjective senses of these words. To correctly identify the sense of the word in a dictionary, the POS tag and the term are not enough. Any `VBN` or `VBG` nodes with an "`amod`" relation are automatically relabeled as adjectives. This allows for later processes to correctly find the sense associated with these words, instead of assigning these a verb sense when the adjective or adverb sense is intended.

Mark Clause Heads

All clause heads and inter-clause relations are marked. The root node of the dependency tree is the head of a clause. Other clause heads can easily be identified by their dependency relation.

The Stanford dependency parser separates the word indicating the type of relation from the inter-clause dependency relation itself. This processing step also removes the node that introduces the new clause, in particular a complementizer, and adds it as annotations to the inter-clause relation.

This step serves several purposes:

- Remove nodes that would not exist in CG and other representations.
- Mark nodes that are clauses
- Associate the term that introduces the clausal relation (such as a complementizer) with the clausal relation itself.

Consider the sentence "I went to the store because I wanted coffee." The parser will return a node for the word *because* which is connected by a "`mark`" dependency to *want*. In this step, the *because* node is removed and stored as an annotation in the relation which connects the two clauses.

Restructure Conjunctions

The Stanford dependency parser represents conjunctions with a "`conj`" and a "`cc`" dependency. The "`conj`" connects the conjoined nodes. The "`cc`" dependency is used to connect the coordinating conjunction to the node indicating the type of relation between the conjoined nodes. The representation is slightly impractical; instead the coordinating relation is made the root of the conjoined nodes. Annotations are added to the conjoined nodes indicating their order.

For example, consider the phrase "the cat and the dog". Figure 5.3 contains a graphical representation of the dependency parse from the Stanford parser. Figure 5.4 contains the transformed version of the graph.

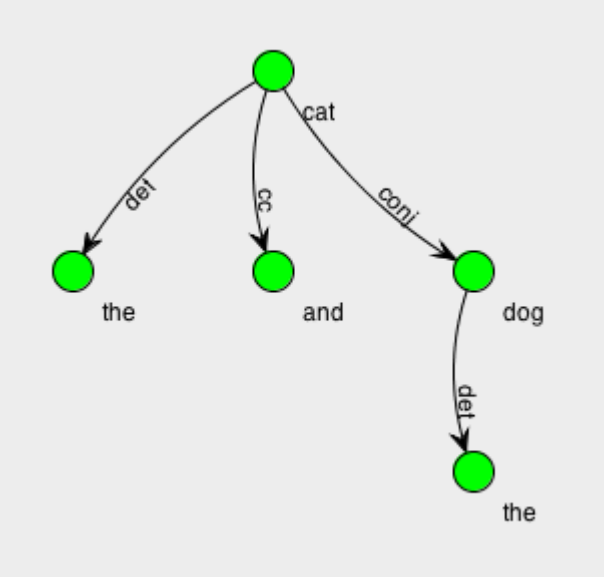


Figure 5.3: Dependency Parse of Conjunction

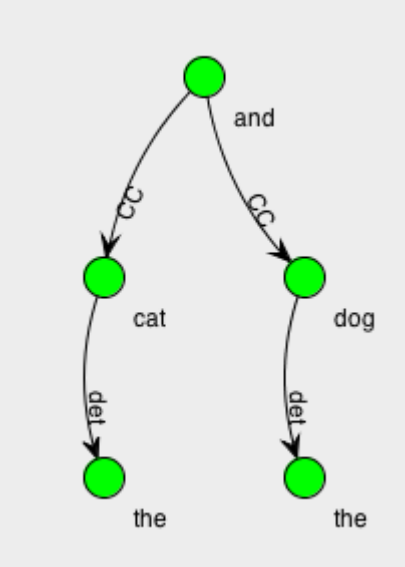


Figure 5.4: Post-processed Graph of Conjunction

Combine Multi-node Expressions

This step searches the graph and the original sentence for multi-word expressions. All the terms of these expressions are combined into a single node, representing the combined meaning. This processing step is relevant for some tasks and not for others; it can easily be toggled on and off. This step was first introduced for the evaluation described in Section 7.1.1.

Three techniques are used to identify multi-word expressions. The first is a greedy left-to-right search for the longest possible n-gram, in the source text, known to be in the dictionary. The second technique is to merge nodes that the parser connects with "`mwe`" dependency, which indicates a multi-word expression. The third technique is merging "`nn`" dependencies into the head node if the head is a proper name (POS tag = NNP). The three techniques are applied in order.[1] For example consider the sentence "The Easter Bunny as well as the Tooth Fairy like the United States." The original

[1]The application of all three technique aligned this representation more closely with Logic Form (LF) for comparison. Furthermore, they helped reduce the number of false matches during reading

dependency tree is shown in Figure 5.5 and the transformed in Figure 5.6. Note how *Easter Bunny* (which is in *LDOCE*), *Tooth Fairy* (which is not in *LDOCE*), *well as* (which has an "`mwe`"), and *United States* (which is in *LDOCE*) all become single nodes.

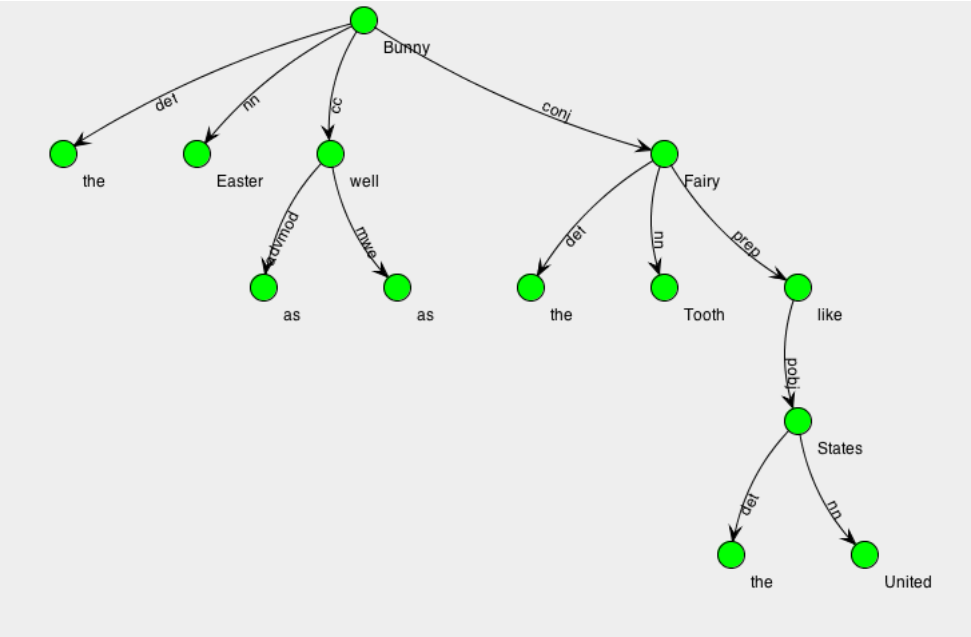


Figure 5.5: Example of Multi-node Expressions

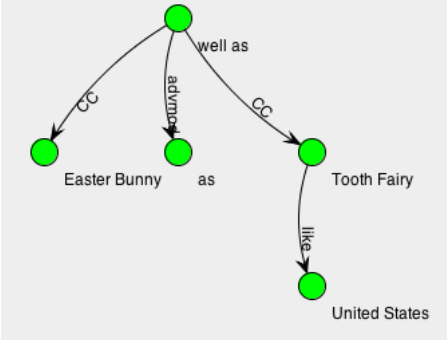


Figure 5.6: Example of Merged Multi-node Expressions

comprehension. The first technique is the most valuable, because the expression is found in the lexicon and can be labeled with a sense. The ordering of the second and third technique are not of concern as long as they are applied after the first technique.

Clausal Links

Many definitions will have subordinate clauses. This is often seen in the definitions of nouns where the subordinate clause describes the appearance, function, or common uses of the noun. We have seen this in the definition of $bicycle_{\text{noun}}$, which has a subordinate clause stating that it is something that you ride. It is important to connect the genus term from the head clause, or term being defined, with something in the subordinate clause. In the $bicycle_{\text{noun}}$ example, vehicle (or bicycle) must be connected as a dependent of ride. In the following section on creating attachments, we will see why this is useful.

The following rules are used to create clausal links, connecting the term in the head clause to the coreferent concept in the subordinate clause. In all cases, we start by finding the clausal dependency relation. Let h be the head of the clausal relation, rc be the clausal relation, and d be the target of the clausal relation (and the head of the subordinate clause).

Condition	Clausal Relation
if rc is of type rcmod	connect h and d by a clausal link
if h is a noun and d is a verb	create a new node n as child of d connected by a "dep" (generic) dependency, and connect n and h by a clausal link

Table 5.2: Rules for linking clauses

These two rules do not cover many examples of subordinate clauses, but they are helpful when processing noun definitions. Both rules are derived from observation of the definitions. Both lead to improvement in Creating Attachments and in later evaluations.

Consider the examples in Table 5.3; their graphical representations are shown in Figures 5.7 and 5.8. In example 1, there is a subordinate clause, whose subject is *that*, and *that* refers to the *animal*. In example 2, the subordinate clause describes an activity involving the bicycle (*ride*) but does not include a concept coreferent to *vehicle*, thus a proxy concept is added as a dependency of *ride*.

5.1.3 Marking Definition Type and Genus

Definitions come in different styles, as described in section 3.2. Some styles should be interpreted slightly differently than other styles, thus detecting the style is a first step in

Example	Term	Definition
1	anteater$_{\text{noun}}$	an animal that has a very long nose and eats small insects
2	bicycle$_{\text{noun}}$	a two-wheeled vehicle that you ride by pushing its pedals (pedal) with your feet

Table 5.3: Example of Definitions Requiring Clausal Links

this interpretation. The interpretation of the genus term varies significantly depending on the definition style.

In this work, definitions are divided based on three features:

1. Is the genus a single term or set of terms?
2. Is there a differentia?
3. Does this definition provide a usage context?

Table 5.4 contains sample definitions showing different definition styles and features. The terms marked in red underline are the genus terms as assigned by this system. The *differentia* is marked in *blue italics*. The usage contexts of definitions are marked in green.

Example 4 is a secondary definition, provided for the same sense as example 3. Examples 1, 2, and 8 all have multiple genus terms. Examples 1, 2, and 6 do not have differentia. Examples 5 and 6 have usage contexts[2] which indicate when to apply this sense. Usage contexts are converted into a subtree called a pattern.

All the definitions/senses that provide usage contexts should only be applied/labeled when the usage context (their pattern) is present in the MIG. One would not apply the definition of *run* found in example 6 to a horse, nor would someone apply the definition of *abandoned* found in example 5 to a child.

Definitions with a list of genus terms, but no differentia, should be interpreted as (partial or complete) extensions. Example 1 is a partial extension of *colour*, while example 2 is a complete extension for the term *bike*. Every concept in the extension is a subclass of the term.

A single term, such as in example 4, is considered the genus term, but with no clear interpretation. In example 4, *bike* is a superclass of *bicycle*, while in other situations, as seen in example 9, it may indicate a subclass or synonymy. The relationship between a definition that has a superclass and no differentia is unclear.

[2]A usage context is a specific text or subtree pattern in an MIG that indicates/signals a particular sense. This is different from the *context* that we have been previously discussing, which includes broader information, such as neighbouring sentences, knowledge about entities, and co-reference.

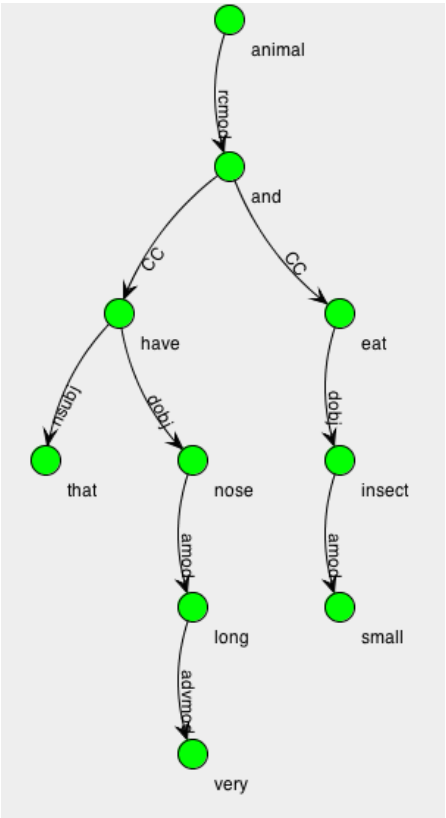


Figure 5.7: Parse Tree for *Anteater*

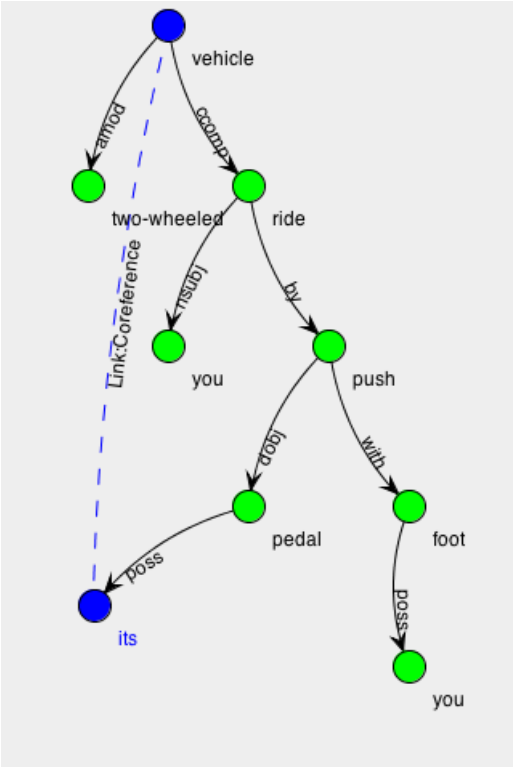


Figure 5.8: Parse Tree for *Bicycle*

While it is not common in *LDOCE*, some dictionaries will use several lexical entries for a term to list subclasses. *LDOCE* usually lists such subclasses in a single lexical entry, with a list of terms (see examples 1 and 2). The definition *machine* as a *vehicle* has no other definition for this sense; there is no information to specify the relation, unlike *bicycle* as *bike*, which has an alternate definition of *bicycle* (3) and the definition of *bike* is clearly a list of two subclasses (2). A separate discussion might be whether there should be a distinct lexical entry of *machine* as a *vehicle* and how it should be related to other definitions of *machines*. For the purposes of this discussion, we should note that because of the absence of other information, it is unclear how this sense and definition of *machine* relate to any others.

Definitions with a single genus term and a differentia can consider the genus term a superclass, as in example 3. Examples 7 and 8 each have two genus terms. Example 7 requires both clauses to be true, thus neither is technically a superclass, but they may be considered sub-events. Example 8 allows a utensil to be from either genus term, though neither is a superclass of all utensils.

Example	Term	Definition
1	colour	red, blue, yellow, green, brown, purple etc
2	bike	a bicycle or motorcycle
3	bicycle	a *two-wheeled vehicle that you ride by pushing its pedals with your feet*
4	bicycle	bike
5	abandoned	an abandoned building, car, boat etc *has been left completely by the people who owned it and is no longer used*
6	run	if a machine runs, it operates
7	eat	to put *food in your mouth and swallow it*
8	utensil	a tool or object *with a particular use, especially in cooking*
9	machine	vehicle

Table 5.4: Sample Definitions Demonstrating Different Styles from *LDOCE*

Classification of Definitions

Definitions can differ in style and structure, which requires different interpretations or processing of the definitions. We classify definitions along two major axes, whether or not they have differentia, and how many genera are used in the definition. Table 5.5 shows how definitions are classified by their structure. The table uses the term *single* to indicate that a definition contains a single genus term, *set* to indicate that a definition has more than one genus term, and *set + etc* to indicate that the list of genera ends in *etc* (suggesting that either list is longer or the list contains examples from the extension).

The classifications for definitions used in this research are *superclass*, *overlap*, *complete extension*, *partial extension*, *contextual patterns*, *properties*, and *?*. *Superclass* means that the concept has a single superclass (*i.e.*, any instance of this type are also of part of the superclass). *Overlap* indicates that instances of this type must also be a subtype of at least one of the genus terms, as in example 7 and 8. Example 7 defines eating as completing two actions, while example 8 indicates utensils are either tools or objects (or both). *Complete or Partial Extensions* are used when there is no differentia, indicating all concepts from the genera are of this type. The genera lists either all the subclasses (*Complete extension*) (2) or the list ends in *etc* and only some example subclasses are listed (*Partial Extension*) (1). Interpretation of a definition with superclass and no differentia is unclear (marked *?*); in this work, I treat the genus term as a super class. This leads to the interpretation that all bicycles are bikes (4) and all machines (of this sense) are vehicles (9).

Genera	with differentia	without differentia
Single	superclass	?
Set	overlap	complete extension (subclasses)
Set + etc	overlap	partial extension (subclasses)

Table 5.5: Interpretation of Genus Term by Definition Features

There are two additional types of definitions: Context Patterns and Properties. Context patterns are defined below in the subsection *Definitions with Contexts*, but in summary, they are composed of two parts: a usage context exemplifying usage and a definition. Properties are definitions which are often in the form of a question or a measurable quantity. For more information on Properties definitions see Section 4.2. These types of definitions are identified using the transformation system. Each transformation contains two key parts: the template and a transform. The transformation system matches a template (a subtree), which can contain restrictions on concept types, structure, relation types, and annotations, against a MIG. If the MIG contains a matching subtree then that subtree is transformed into a new subtree, as described by the transform. The transformations for detecting property definitions simply add an annotation (indicating that the definition is a property definition); nothing else about the MIG is changed. For the list property transformations (pattern and transform), see Appendix B, section *Property Transforms*.

Genus Detection Method

My method for genus detection is quite simple: a small number of weighted factors are considered and each concept is scored based on these factors. The concept with the highest score is considered the genus term.

The method is simple and accomplishes the goal with little effort or overhead, but certainly it could be made into a more formal task. More factors could be considered, feature/factor learning could be applied, and the most practical improvement would be learning feature (or factor) weights. None of these are applied in this work, as the genus extraction is a means to an end, and not the main focus of this research. All weights were manually determined, through trial and error. Applying overly complicated machinery, or over-developing a single aspect can easily distract from the goal of building and applying auto-frames. Furthermore, the results of the genus extraction are high, as we will see, and meet the needs of this research.

Understanding the factors requires a quick review of how definitions are pre-processed.

Recall that many definitions are not complete sentences, therefore the parse trees are often incorrect. By adding a prefix text to some definitions, many of the definitions were correctly parsed. Subsequently the concepts related to the prefix text were removed, leaving a correctly structured parse tree representing the definition. Revisit section 5.1.1 for more information.

The five factors considered when identifying the genus concept are as follows:

Root	The root node of a parse tree is often the genus term.
Dummy Subject	The node, which has the dummy concept as a subject, is often the genus term. This factor works effectively on definitions that required a prefix.
Part of Speech	The genus term tends to have a part of speech that agrees with the lexical unit being defined. *i.e.*, verbs expect verbs, nouns expect nouns, adverbs expect adverbs, however adjectives expect a verb or preposition which indicates how the description is related to the noun being defined.
X is used for	If a sub-graph is found representing a structure like "X is used for", then X is expected to be a term being defined and will be coreferent with a genus term.
X is used to mean	If a sub-graph is found representing a structure like "X is used to mean", then X is expected to be a term being defined and will be coreferent with genus term.

Table 5.6 contains sample definitions from each POS. The definitions are meant as examples of how the genus terms have a generally predictable POS: Nouns usually have nouns as genus terms; verbs usually have verbs; adjectives have noun modifiers, either in the form of verbs or adjectives, occasionally as propositional phrases; adverbs have verb modifiers, either in the form of adverbs or prepositional phrases.

Special considerations are made for concepts with factors connected through conjunctions; the weight of the factors of the conjoined concepts is moved into the conjunction concept. Thus the conjunction is considered the genus, if the children are likely genus terms. The part of speech factor considers the conjoined concepts; thus conjunctions, including their children, are acceptable genus terms, if the connected concepts are acceptable. This can be seen in Figure 5.9, which illustrates a parse tree for a definition

#	Term	POS	Definition
1	fast	Adj	moving or traveling quickly
2	scary	Adj	frightening
3	unkind	Adj	nasty, unpleasant, or cruel
4	bicycle	Verb	to go somewhere by bicycle
5	bicycle	Noun	a two-wheeled vehicle that you ride by pushing its pedals with your feet
6	fast	Adverb	in a short time
6	quickly	Adverb	fast
6	quickly	Adverb	for a short time

Table 5.6: Examples of Typical Genus Terms

of *close* from *Extended WordNet*. The root concept is a conjunction, which is treated as the genus concept.

Definitions with Usage Contexts

Definitions with usage contexts are usually realized in two ways in *LDOCE*, while no definitions with usage context were observed in *Extended WordNet* or *WordNet*. All usage context patterns include the word being defined and are composed of two parts: the usage context and the definition. The usage context is also referred to as the pattern.

One realization is an *if-then* structure, though technically the word *then* usually does not appear in the definition. The *if* portion contains the pattern, while the *then* portion contains the definition. This can be seen in example 1 in Table 5.7 and the definition of *run*. This type of pattern is most frequently used for verbs.

The second realization uses a passive voice with the pattern as the passive subject of the verb, and the remainder of the clause is the definition. This type of construction is used mostly for adjectives. This can be see in example 2 in Table 5.7, the definition of *abandoned*.

#	Term	POS	Definition
1	run	Verb	if a machine runs, it operates
2	abandoned	Adjective	an abandoned building, car, boat etc has been left completely by the people who owned it and is no longer used

Table 5.7: Examples of Definitions with Context Patterns from *LDOCE*

Usage context definitions are detected using the transformation system (described in

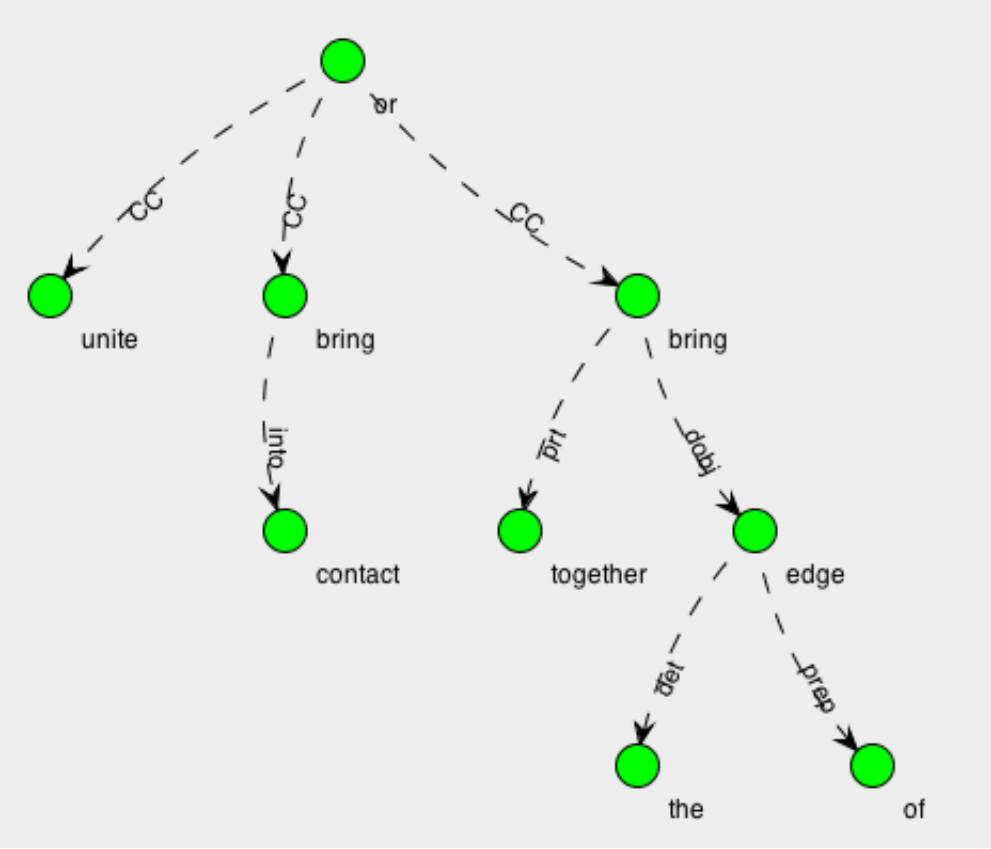


Figure 5.9: Example of a conjunction as the genus for the definition of *close*, verb, *Extended WordNet*.

Appendix B). A set of transformations (pattern and transform) are defined in Appendix B.2. The transformations for usage context definitions annotate the usage context (pattern), the definition, as well as remove the relations (clausal or otherwise) connecting these two parts.

A separate post-processing step tries to label coreferences between the pattern and the definition. In Example 1, *it* refers to the *machine*. In some cases the coreferences are not simple pronoun substitutions. This thesis does not include any coreference resolution beyond the pronoun resolution provided by CoreNLP.

5.1.4 Assigning High-Confidence Sense Labels

This step of processing only assigns sense labels to the terms in definitions in a few circumstances: the term and POS combination only has a single sense associated with them (*i.e.*, there are no homographs and the term is not polysemous); there exists a bi-

directional relationship between two definitions; the term is used[3] in the context section of the text pattern, thus the term is the one being defined; there is a common hypernym for a list of conjoined concepts.

A bidirectional relationship between two definitions holds when *Term 1* is defined using *Term 2* and *Term 2* is defined using *Term 1*. This relationship can be seen between the senses of *bicycle* and *ride* presented in Table 5.8.

As previously shown, definitions with usage contexts contain a pattern component that contains the word being defined. Thus, most context definitions refer to themselves in the context pattern. If the word being defined appears in the context pattern, we label it as itself as seen earlier in the examples found in Table 5.7.

Example	Definition
Ride noun_1 (*LDOCE*)	to travel on a *bicycle* or motorbike
Bicycle noun_{1a} (*LDOCE*)	a two-wheeled vehicle that you *ride* by pushing its pedals with your feet

Table 5.8: Definitions with Bi-Directional Relationships

When a set of terms is conjoined in a definition, they often have a common hypernym. When considering all possible senses of each conjoined concept, if there is a single set of senses (one sense for each concept) for which there is common hypernym, then the concepts are labelled with those senses. The search for a common hypernym is limited to three levels. All possible sense labels for all possible genus terms for each concept are considered. This can produce a large search space; limiting the search to three levels has provided manageable runtimes. Furthermore, three levels provides a balance between searching for common general terms and avoiding overly general terms which encompass many hyponyms. The restriction that there must be only a single set of sense labels should remove circumstances where an overly general hypernym is found because such a hypernym would usually result in many sense sets.

The first two techniques do not disambiguate many of the terms in definitions, but provide a sort of priming for the subsequent steps, which are iterative. 38% of the terms defined in *LDOCE* only have a single sense. 17% of the terms used in definitions have

[3]The term being defined appears in the definition with the same intended sense as the one being defined, but the term is used to exemplify the usage.

only a single sense. About 7% of senses have a bi-directional relationship that can be clearly disambiguated.

5.2 Connection Search

Connection search is a process that combines elements of both Word Sense Disambiguation (WSD) and Semantic Role Labeling (SRL). This process attempts to find the attachments between senses or auto-frames as was described in Section 4.4.3. The connection search process is intended to produce representations like those shown in Sections 4.4.3, 4.5.3, and 4.6.2.

In some ways, connection search resembles the idea of spreading activation; each concept is expanded (spread) to its potential definitions or auto-frames, and then there is a search for connections/attachments between adjacent concepts. This algorithm can be applied to lexical senses from a dictionary or to auto-frames. In fact, connection search is used for disambiguation of senses, in the construction of auto-frames, and the application of auto-frames. The method has three high-level steps that can be repeated. Each step is described in the following subsections.

Expansion - Each concept is assigned senses, and then the MIG is expanded to include their definitions or auto-frames. (Section 5.2.1)

Creating Attachments - Search for possible attachments between concepts that are connected by dependencies. (Section 5.2.2)

Weigh and Prune - Weigh each interpretation (combination of sense assignments and attachments), then prune low-weighted senses and attachments. (Section 5.2.3)

5.2.1 Expansion

The idea of expansion was introduced in Section 4.4.2 with a variety of examples. This section shows in detail how expansion was implemented, and the challenges associated with it. The purpose of expansion is to include the knowledge, implications, and commonsense information about each concept. Each concept is assigned senses, then the MIG is expanded to include the definitions (or auto-frames) of those senses. When expanding a concept to its senses, a *sense* link is created between the concept and the sense; an

interpretation restriction is created between all the senses.[4] When each sense is expanded to its definition(s), *genus* links are created between the sense and the genus.

Let us refer to graphs added during the expansion step as *expansion graphs*. The expansion graphs for a concept are the graphs that were added when expanding that concept.

The first iteration of connection search starts by expanding surface-level concepts, subsequent steps expand concepts that were added by the previous iterations of expansion. As seen in Sections 4.4.3 and 4.6.2, attachment and contextualization can require expansion of both genus terms and arguments, as demonstrated in the example of the female with a mustache.

Consider a sentence (surface graph to be expanded) that contains C concepts in it and the average concept has S senses. During the first iteration there are $C * S$ expansions. If the average sense has N concepts in its definition, then the second iteration will have $C*S*N*S$ expansions and the third one, $C*S*N*S*N*S$. A pattern becomes apparent based on the depth (n) (iteration of expansion), and the MIG will grow exponentially $C * S * (N * S)^{n-1}$ or $(O(c^n))$, where c is a constant.

Expansion has immediate performance implications related to lag in looking up each sense and definition in the knowledge base, as well as implications in subsequent steps. Since the number of expansions grows exponentially, even a small number of expansions and a short lookup time for definitions can cause long delays.[5] Expansion and connection search is normally limited to 2 or 3 iterations; this has been experimentally observed as a good balance of depth and runtime. However, expansion to an auto-frame likely only requires one iteration, since it already contains multiple graphs and the expanded definitions. Consider that if the auto-frame is constructed using two iterations of expansion, then inserting an auto-frame is like applying three levels of expansion in one iteration, because the auto-frame includes the core-senses and the two iterations of expansion.

Exceptions to expansion

There are a number of exceptions to sense expansion. Most of these exceptions are motivated by the need to find connections between concepts not in the dictionary, or because the terms are so general that almost anything might match them.

Only nouns, verbs, adverbs, and adjective are expanded as described above. The

[4] Links and restrictions are described in Section 4.4.2: *A Closer Look at MIG*.

[5] The small lag of a database lookup is normally trivial but can have a huge impact on this step. Having an in-memory cache of common senses and definitions can significantly improve the runtime.

words *someone* or *something* are not expanded, because they are very general. Pronouns are also not expanded: they act as a stand-in for nouns found elsewhere the text. The top 1% most frequent nouns or verbs in this dictionary's definitions are not expanded, as they are usually very general and would likely be near the top of any taxonomy or ontology. The choice of top 1% was determined experimentally for *LDOCE*.

Names are assigned senses based on the type of entity they represent. The names are not actually expanded. The knowledge base contains very little information about named entities. Named entity recognition is performed by the Stanford pipeline, which provides the type information.

Determining which senses should be assigned for each named entity type could be a long task, and it is dictionary-specific. Instead, lists of terms that could represent the type of the named entity were manually selected. For each type of named entity returned by the Stanford NER, a list of terms, which describe the NER class, was created. For example, the list for the person type contains the terms *person*, *human*, *someone*, and *who*. For a named entity, the appropriate list for that named-entity type is selected and all noun senses for the terms in the list are assigned to the named entity. In later processes, this enables creating attachments with a named-entity and attachment points in auto-frames.

A named entity may be assigned senses that are not truly representative of its type, but it is likely that the correct sense, will remain after the *Weigh and Prune* step of connection search. For example the *United States* might be assigned senses for country, location, and nation, but also incorrectly as city. It is important for the connection step that a correct sense is present, and incorrect senses should get removed during the *Weigh and Prune* step of connection search, because they are unlikely to have any connections.

The words and named entity types that have senses assigned but no expansion applied are listed Table 5.9,[6] as well as what word senses are assigned to them.

LDOCE and *WordNet* do not provide useful sense for the words *where, what, and who*, but this has not created any observed problems.

5.2.2 Creating Attachments

Searching for possible connections is similar to SRL that is primarily resolved through selectional restrictions. Given two surface concepts, $c1$ and $c2$, connected by a depen-

[6]The words in the category "person, someone" are actually detected by their part of speech (personal pronoun = PRP).

Word or Named Entity Type	Terms for which Senses are Drawn and Assigned
it	thing and object
I, you, he, she, we, they, us	person, someone
thing, something	something, thing
who	person, someone, organization, company, who, human
where	organization, place, somewhere, location, country, area, town, city, country, where
what	thing, object, action, event, what
when	date, time, when
Named Entity = [PERSON]	name, person, someone, human, town, city, country, location, who
Named Entity = [LOCATION]	name, organization, place, somewhere, location, area, town, city, country, where, what, who
Named Entity = [ORGANIZATION]	name, organization, company, what, who
Named Entity = [TIME]	date, time, when
Named Entity = [DATE]	date, time, when
Named Entity = [OTHER]	name, person, someone

Table 5.9: Senses Assigned to Named Entity Types

dency relation, r, the goal is to identify a link path between $c1$ and $c2$. Expansion adds links to *genus* terms and *attachment points*. The goal of this step is to find attachments between adjacent senses or auto-frames.

Consider for example the sentence "The computer is now running." *Computer* should connect to *run* through an attachment point for *machine*, as is found in the definition "if a machine runs, it operates". Attachments to other senses of *run*, such as *move* or *race*, should not be found because there would be no matching attachment point.

Consider for example the sentence "I went to the airport by car." *Car* should be attached to *went*, through a *drive* or *travel* attachment point from the *car* auto-frame. Another perspective on this is that *go* does not evoke a means of travel, while the noun *car* does evoke the ideas of *driving*, *traveling*, and ultimately *go*; thus the attachment point is in *car* and connects to *go*. *Go* does include an attachment point for a destination to which *airport* can be attached.

Recall that certain links in MIG indicate that concepts are coreferents. For example, the *genus, coreference, clausal, and example* links between two concepts in an MIG indicate that these instances refer to the same instance.

Defining the search space for attachments required considering the computation cost of different approaches. Expansion leads to an exponential increase in the number of concepts. Searching for attachments between all combinations of concepts from the expansion graphs requires factorial time, $O(n!)$.

Consider an approach attempting to search for attachments between all concepts in a sentence or clause; comparing all combinations of the attachment points in all of the auto-frames to one another is a factorial-time process, with a large n value. This approach is computationally impractical, although it was tested and led to finding incorrect or coincidental attachments between distant auto-frames.

Instead, only searching for attachments in concepts that were connected by dependency relations in the surface level graph leads to much smaller n. While this is still a factorial-time search, we have reduced the search space to just immediately connected concepts, and will avoid distant coincidental attachments.

If a surface concept $c1$ is dependent on a concept $c2$ by relation r, then the goal is to explain r by creating an *attachment* link between a genus concept from $c1$ or $c2$ and an attachment point from the other. For two concepts to be attached, they must be of the same sense. To attach the concepts *vehicle* and *car* actually requires *car* to be expanded to *vehicle* (and having a genus link) then attaching the two *vehicle* concepts.

A more natural way to consider this is that $c2$ should fill an *attachment point* (argument, slot, role, frame element) of $c1$ and should meet its selectional restrictions. As was demonstrated in the example in Section 4.4.3, a path connecting the concepts defines how these concepts are connected. The *mustache* and *teacher* were attached because the teacher is person and a mustache is hair above the lips of person.

When an attachment is made between concepts $c1$ and $c2$, the search space for attachments for $c1$ is expanded to include the immediate children of $c2$ and vice versa. This is done because auto-frames that are large tend to include more attachment points than just the immediate dependencies of a concept. Consider for example the sentence "We drove quickly to the hospital in the ambulance." The parse tree for this is shown in Figure 5.10. *Hospital* can be attached as the destination of *drive*, while *vehicle* is a child of *hospital*, but it can be attached as the *vehicle* being *driven*.

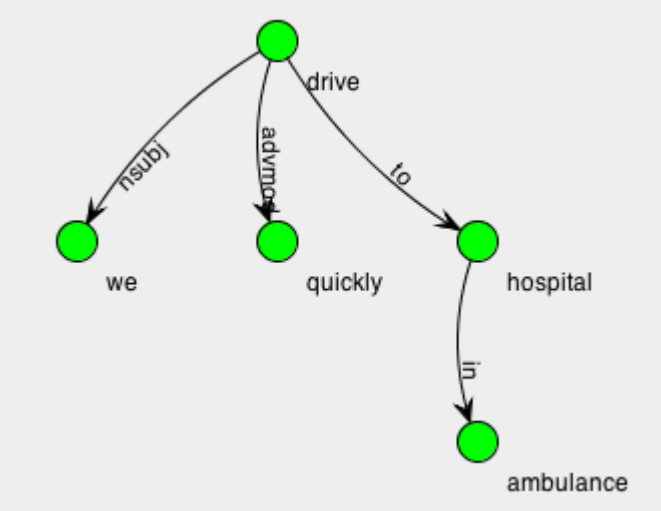


Figure 5.10: Example Parse Tree with Indirect Attachment

Attachment Search Algorithm

Given a surface relation r, for which we are trying to find an attachment link, let h be the head of the dependency and d the dependent. Recall that attachment points for an auto-frame are all non-genus concepts. The attachment point for lexical senses from the dictionary are any non-genus terms in the definition. Let the set of *attachment points* be $\{ap\}$.

For a given d (a dependent of h), create a set of coreference concepts ($\{d_{coref}\}$) which should include coreference via genus, coreference, and clausal links. Whether processing auto-frames or lexical senses from a dictionary, $\{d_{coref}\}$ contains all of the genus concepts (thus generalized type information) as well as information from other references to d.

If any of the concepts in $\{d_{coref}\}$ are of the same sense as those in $\{ap\}$, then create an attachment link between these concepts. Algorithm 5.1 formally defines this process. The algorithm is composed of three nested loops. The outer loop iterates over surface concepts, let us say there are s surface concepts; s is a function over the size of the input data. The next loop iterates over concepts only one edge away from the surface concept. Let us say on average there a adjacent concepts. The inmost loop iterates over concepts that are coreferent with the current adjacent concept. Let us say on average there are r coreferent concepts. This algorithm is cubic time, $O(s*a*r)$. Now, a is likely reasonably small (*i.e.*, 3-5) and not likely to vary widely; if we treat a as a constant, then the algorithm becomes quadratic. There is only one variable that is a function of the size of the input (s), so the algorithm should scale linearly ($O(s)$) with input data.

The method $getCoreferent(d)$ returns all concepts that are coreferent to d by traversing all links that indicate coreference and returning the set of concepts that were visited.

Algorithm 5.1 Creating Attachments

```
for c ← each surface concept do
  {ap} ← getAttachmentPoints(c)
  for d ← each dependents(c) or head(c) do
    {d_coref} ← getCoreferent(d)
    for d_coref from {d_coref} that has the same sense as an ap from {ap} do
      createAttachment( ap, d_coref )
    end for
  end for
end for
```

The purpose of this process is to find attachments between adjacent auto-frames or senses. These attachment links act as assigned roles, allowing later processing to compare syntactically different, although semantically the same, expressions to one another. Attaching senses allows for the next step in connection search (Prune and Weigh) to filter unconnected senses in a manner similar to the Lesk WSD algorithm[7] (Banerjee and Pedersen, 2002).

As previously illustrated with the example "I went to the airport by car ", at times dependents, such as the *car*, can have attachment points (*travel*) that attach to the head (*go*). Thus, a dependent d also searches its head h to see if it matches an attachment point. These types of attachments tend to occur when a genus concept is a noun and there is another clause expressing its usage. Recall the clausal links from Section 5.1.2, which connected *anteater* as the subject of *eat* and *bicycle* as a dependent of *ride*.

Consider the example from Section 4.4.3 in Figure 4.3 repeated here for convenience as Figure 5.11. The previous expansion step would have added the graphs containing the definitions of *teacher* and *mustache*, but the attachment link would not exist. Thus the list of attachment points for *teacher* in this example would be *teach* and *job*, while for *mustache* the list would be *grows*, *lip*, *man*. An attachment can be made between *man* and *human* if they have a common sense.

Note that if multiple concepts (that are not coreferent) can satisfy a single attachment point in an auto-frame, then a mutual exclusion interpretation restriction is needed in the MIG, to enforce that only one of these attachments is valid in any single interpretation.

[7]Lesk assigns senses by weighing the amount of overlap between the glosses of the senses in a context, while these attachments indicate overlap between senses and auto-frames.

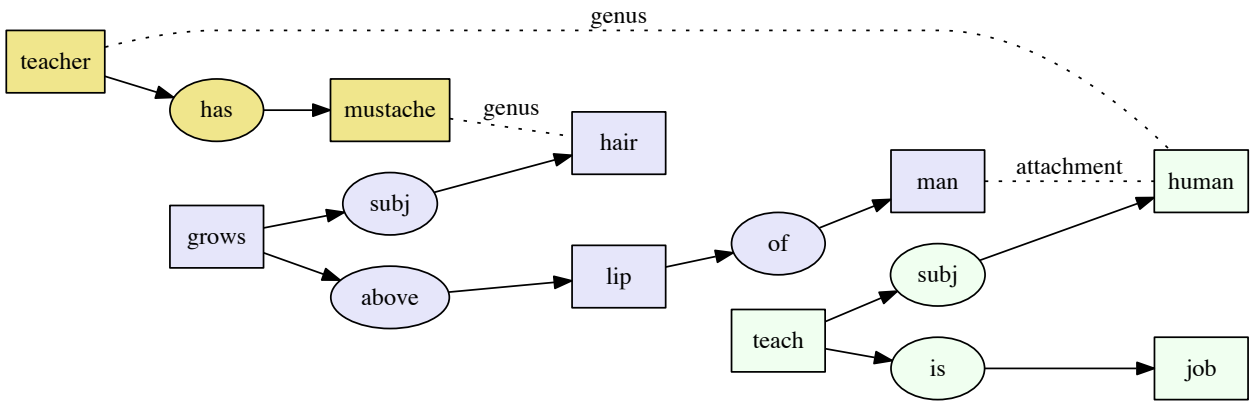


Figure 5.11: Example CG with Modulations: *The teacher has a mustache*

Adjectives and Adverbs

This research finds *attachment* links between verbs and nouns. Many adjectives and adverbs are defined in terms of what they modify and not with a genus term; thus definitions need to be processed as an upwards search. Adjectives and adverbs are usually the dependents of the concept they modify.

Table 5.10 has definitions for adjectives and adverbs that are defined by the word modified. Words between angle brackets (*e.g.*, <example>) are placeholder words for the word modified by the definition. Words in red are the words being modified, words in blue are modifiers, and words in green are heads of relations with the modified words.

Note that sample definitions 1 and 5 include words which could be considered a genus term and would be coreferent with the adjective being defined. Only one (1) includes something representing the word modified, but most can easily have a placeholder added that represents the word being modified. Three (1, 2, and 3) include concepts in the definition which would be heads of the word being modified.

5.2.3 Weigh and Prune

The purpose of the weighing and pruning step is to score and rank different interpretations, that is, sets of assigned senses and attachments, and then filter unlikely (low-weighted) interpretations. There can be many approaches to such a task, from different weighting schemes to different acceptance or rejection criteria. A simple rule-based ap-

#	Word	Definition
1	Abandoned *LDOCE*, adjective	an abandoned building, car, boat etc has been left completely by the people who owned it and is no longer used
2	Hairy *LDOCE*, adjective	<creature> having a lot of body hair
3	fast *LDOCE*, adjective	<thing> moving or traveling quickly
4	fast *LDOCE*, adverb	<action> in a short time
5	fast *LDOCE*, adverb	<action> soon and without delay

Table 5.10: Examples from *LDOCE* demonstrating the difficulties involved in connecting adjective and adverb definitions

proach is used in this proof of concept. Certainly the process could be improved with statistical methods and machine learning. There are likely many techniques used with statistical parsing that could be applied here.

As with the previous steps, consider the computational cost of weighing interpretations. Expansion has increased the MIG's size exponentially, creating attachments had a combinatorial cost, and this process must weigh and prune interpretations. Interpretations are sets of assigned senses and assigned attachment links validated by interpretation restrictions in the MIG. Considering all possible sets of sense assignments and attachment links leads to a combinatorial problem with factorial processing time $O(n!)$. The number of interpretations to consider grows with the number of nodes, the number of expansions applied, the number of attachments created, and the average number of senses per concept.

A number of approaches were attempted for this step, such as top-down scoring and bottom-up scoring. Processing times as high as 1h per sentence were observed. This suggests either that n is high and the number of combinations is extremely high, or the processing time per step is not small enough. To reduce the search space and lower the computation cost, a greedy approach was taken. The greedy algorithm presented below

and shown in pseudocode in Algorithm 5.2 turns a combinatorial problem into $O(n * s)$, where n is the number of concepts in the MIG and s is the average number of senses per concept.

Weighing and pruning of senses and attachments is done only with a local context. The selection of senses and attachment links is done for each concept independent of other concepts. That is, weighing senses and attachments only considers the problem of choosing the highest-weighted senses and attachments for this concept, without regard for the impact of these choices on other concepts or the interpretation as a whole. By limiting the search to the local maxima, we are no longer considering how all combinations of assignments affect one another. Instead we consider each of the n concepts independently and then consider the s assignments for each concept. This results in a complexity of $O(n * s)$. However, since MIG still supports multiple interpretations, we need only prune unlikely interpretations, while closely ranked interpretations may remain. Thus even if the correct/intended sense and attachment were not the most highly ranked, they may still remain in the MIG.

Theoretically, once the number of interpretations has been reduced significantly by using this greedy independent approach, then a separate process could weigh and prune any remaining interpretation as a whole.

Algorithm for Weighing and Pruning

The weight of each graph is calculated using the weight of each concept in the graph, the weight of each concept is calculated using the weight of its senses, and the weight of each sense is calculated using the weight of the expansion graphs. This forms a recursive postfix algorithm (bottom-up) for weighing a graph, as it depends on the calculated weights of the expansion graphs. This process starts with the lowest concepts, that is, the nodes in the deepest expansion graphs. Since the weight of graphs is calculated using attachment links and attachments only connect different surface level concepts or their expansion graphs, then the weight of each graph, concept, and sense is calculated using the number of connections with other surface-level concepts.

The process has been broken into 4 steps:

1. Process the attachments for each concept
2. Process each sense of a concept
3. Prune senses
4. Weigh the graph

The following sections describe each of the steps, then the entire algorithm is defined in pseudocode (Algorithm 5.2).

To help illustrate the process, the description of each step will refer back to Figure 5.12. This figure has been very simplified for this example; the definitions are concise but not precise. Labels have been removed from dependency relations. Blue boxes represent concepts and blue arrows represent dependency relationships. Red arrows are genus links, connecting a concept to the genus term of an expansion graph. Green arrows are attachment links, connecting to concepts, which would have been attached during the step of creating attachments.

The original sentence was "John deposited money in the bank." In this example, *money* has one sense and it has not been expanded to conserve space. *Deposit* was expanded to two senses: "put money in bank" and "mineral found in ground". *Bank* was expanded to two senses: "ground beside water" and "organization that manages money".

1. Process the attachments for each concept.

The implementation considers *attachment* links as evidence that a sense connects with other senses and may be the appropriate sense. There are many factors to consider when creating a weighting scheme: word or sense frequency, characteristics of attachment links or link paths, what surface dependency relation is being modeled, weighing according to the type or depth of an attachment point.

The weight of an *attachment* link will be the Document Frequency (DF)[8] of the linked word. DF was chosen because it will weigh less common senses more highly; if a sense is uncommon but fits a context, it will be weighted highly. And words that are common will be weighted lower, because they will fit more contexts.

The use of the dictionary as a corpus is less than ideal, as well as the use of frequency of words instead of senses. It would instead be more appropriate to have the frequency with which a word or sense is evoked in a corpus (*i.e.*, the frequency with which a sense appears in an auto-frame or an expansion graph). For example, the word *object* may not appear often in any corpus, but many words would evoke it and it would appear in many auto-frames. *Object* should be treated as a high-frequency word, which can easily be matched in many contexts and thus attachments involving *objects* should be ranked lower than other less frequent words.

[8]In this case DF stands for Definition Frequency, but it is more important that the fundamental idea is understood. $DF = n_t/N$ where N is the total number of documents and n_t is the number of documents that contain the term.

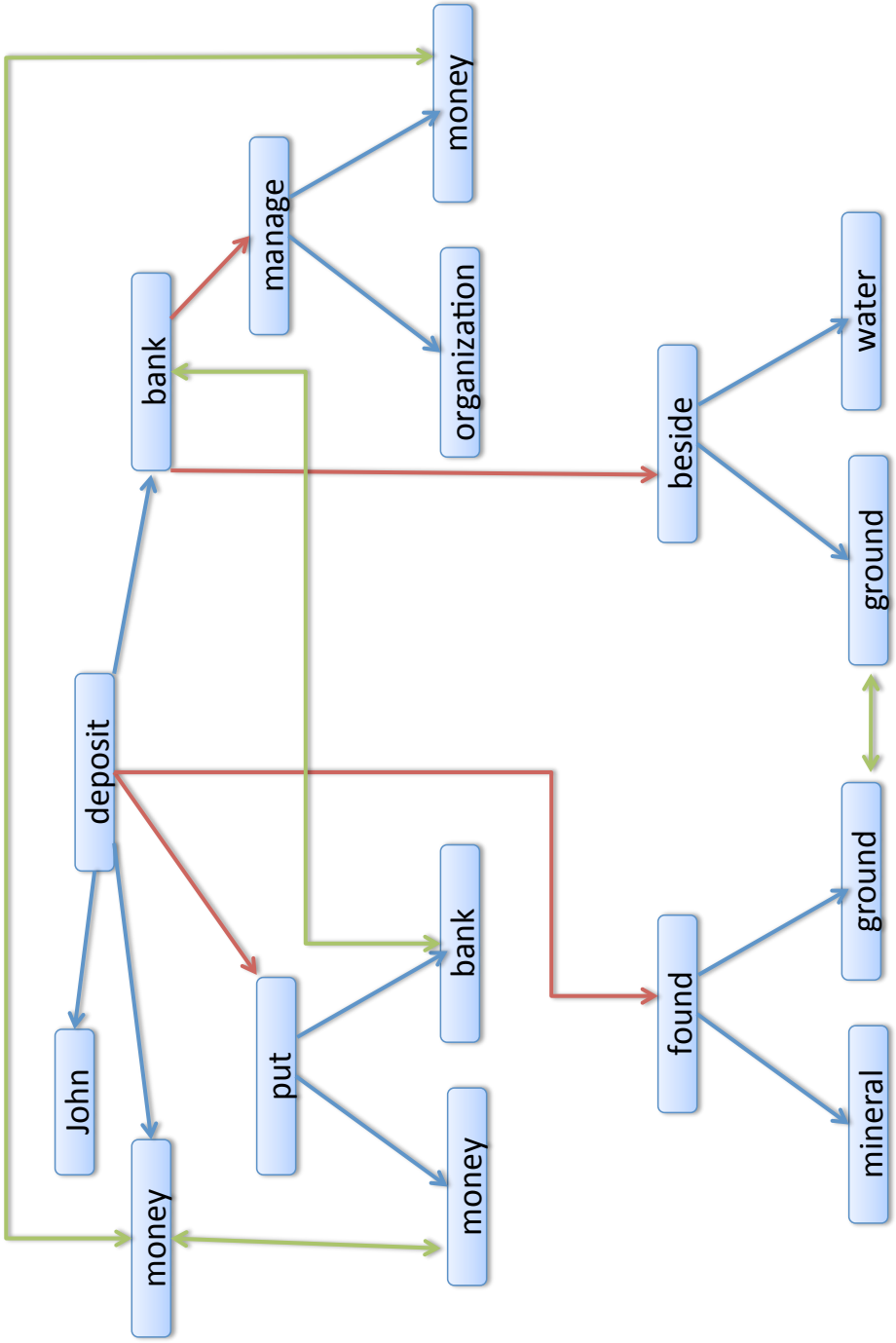


Figure 5.12: Example of Weighing and Pruning a MIG

Often definitions are constructed so that their syntactic arguments will match those of the words they define. To increase the weight of *attachment* links (and thus interpretations), which have matching syntactic structures, weights are scaled depending on how the syntactic relations match. The rules used in scaling a link weight are as follows.

1. If a particular attachment link models the same syntactic relation as the surface relation *subject, direct object, indirect object, prepositional phrase*, then the full weight is applied.
2. If a particular attachment link models a generic relation (*dep* dependency) at either the surface level or within an expansion graph, then the weight is scaled to 80%.
3. If a particular attachment link models a *subject, direct object*, or *indirect object* relation at the surface level and within a definition, then the weight is scaled to 80%.
4. If none of the rules above is applied, then the link weight is scaled to 60%.

Consider Figure 5.12. This step only processes attachments and assigns weight to the sense for which the attachment is made. Consider for example the attachment between *ground* and *ground*; the sense of *ground* as in *land* would have a weight, and not the sense of *ground* as an *electrical ground*. The attachment connecting the *bank*s would only connect *bank* as in financial institution, even though one of these *banks* shows a possible expansion to *riverbank*. Thus senses for the concepts *money*, *bank*, and *ground* are assigned weights at this step.

2. Process each sense of a concept.

Not all concepts and senses will have attachment links, thus some concepts will have a weight of 0. During this step, we assign weights to all concepts. The weight assigned to a concept will be the highest weight from any of its attachments.

Again, consider Figure 5.12. Weights will be assigned to each concept in the expansion graphs. Many concepts will have 0 weight: *mineral, found, beside, water, put, organization*, and *manage*. *Ground, money, and bank* will be assigned the maximal weight of their links. Since each only has one link, this will be the weight of that link. Note the maximal value is chosen because it is locally representative of the most likely interpretation and is the least likely attachment to be pruned. In addition, combining the weights of many attachments would involve testing if the attachments are mutually exclusive, which would increase the the cost of the computation, and it would not account for the fact that some of the attachments may be pruned.

3. Pruning Senses

Once sense weights have been assigned, pruning is done to senses below a relative threshold of the most highly-weighed sense. This is intended to remove less likely senses. When removing senses, any attachments associated with these senses and expansion graphs are also removed. Pruning only occurs if the most highly-weighed sense is above a particular absolute threshold. This minimum absolute threshold filters cases where there are no strongly attached senses.

The relative threshold is intended to keep many highly-weighted senses, but filter out unlikely senses. The absolute threshold only allows pruning to occur when there exists at least one highly-weighted sense.

The absolute threshold used to allow pruning has been set to 0.05. The relative threshold used to filter less likely senses is 0.8. Selecting this relative threshold controls a balance between including the right sense and including too many wrong senses. These values have been selected through manual testing and evaluation of the resulting interpretations.

The algorithm is as follows: The sense with the highest weight is selected, let the weight be w. If w is below the absolute threshold of 0.05 then no pruning is done. Otherwise, w is multiplied by relative threshold 0.8 to produce the filtering threshold, f. All senses with a weight less than f are filtered.

Consider our example; during the first iteration where the expansion graphs are processed, the only senses to be pruned would be the unrealistic senses mentioned in step 1. *Ground* would prune the sense *electrical ground* and *bank* would prune the *riverbank* sense from the sentence "put money in bank".

During the second iteration[9] where the surface level graph is processed ("John deposited money in the bank"), the senses *put money in bank* and *organization manage money* would be highly weighted (by step 4). The alternative senses *mineral found in ground* and *ground beside water* would be filtered because of their significantly lower weight.

4. Weighing a graph

When weighing a graph, the first step is getting weights for each concept, which is done during steps 1, 2, and 3. The weights of all concepts in the graph are added together, with the exceptions of conjunctions where only the highest branch is taken. Consideration could be made to normalize the weights, as summing them can cause them to rise above 1.0. The challenge with normalization is scaling the number of attachments by attachment points. Some definitions may only provide one attachment point, while

[9]Recall that the prune and weigh algorithm is a bottom-up algorithm.

another provide a dozen; normalization should somehow account for the likelihood of one attachment point matching versus several attachment points. After some investigation, no solution was found, so no normalization has been applied.

In general, more attachments should lead to higher weights. When the weight of the graph is propagated to its parent sense, a scaling of 0.8 is applied to account for the attachments connecting to a more generalized sense.

Again, consider Figure 5.12. The weight of each expansion graph would be calculated, based on the weight of each concept. "Put money in bank" would likely be highly weighted having two attachments, where other graphs have only one. Now the weight of each graph is propagated to the sense it represents in the next level of the graph (that would be the surface graph, in this case "John deposited money in the bank"). The weighing process begins again at step 1, but for the surface graph. On this second iteration, during step 1 *bank* as in organization would be highly weighted because it has weight coming from the expansion graph and from an attachment to "put money in bank".

Declarations for Algorithm 5.2

Let $DF(c)$ return the DF value for concept c.
Let $x.weight$ be the weight of concept, sense, or link x
Let $senses(c)$ return the possible senses for concept c
Let $links(x)$ return the attachment links associated with concept or sense x
Let $subGraphs(c)$ return the auto-frames/expansion graphs for concept c
Let $minThresPrune$ be the minimal sense weight to consider for pruning (0.05)
Let $acceptThres$ be the relative threshold used to accepting or pruning senses (0.8)
Let $graphWeight(g)$ return the weight of graph g
Let $prune(s)$ remove the sense s and its associated expansion graphs and attachments
Let $maxWeight(a)$ return the weight of the attachment link in a with the highest weight

Algorithm 5.2 Weigh and Prune Concepts

```
for each concept c in graph G do
  # Assign a weight to each link based on DF
  for each link l from links(c) do
    l.weight ← DF(c)
  end for
  # Assign a weight to each sense, based on the max weight of the links
  for each sense s of senses(c) do
    s.weight ← maxWeight(links(s))
  end for
  # Get the weight of an expansion graph and propagate weight up to the sense
  for each sense s of senses(c) do
    # Decay the weight coming from subgraphs by 0.8
    ew ← 0.8 * graphWeights(subGraphs(s))
    if ew > s.weight then
      s.weight ← ew
    end if
  end for
  # The weight of the concept is the max weight of the senses
  c.weight ← maxWeight(senses(c))
  # Prune senses
  if maxWeight(senses(c)) > minThresPrune then
    minAccept ← maxWeight(senses(c)) * acceptThres
    for each sense s of senses(c) do
      if s.weight < minAccept then
        prune( s )
      end if
    end for
  end if
end for
```

5.2.4 Constructing Frames

This section describes how senses will be merged into auto-frames and what additional senses will be included in an auto-frame. Sentences are divided into three classes: core, evoked, and secondary evoked.

Core senses are the senses that an auto-frame is modeling. If auto-frames are considered coarse-grain senses, then these are the senses that are grouped together.

Evoked sentences (or senses) are the senses that an auto-frame includes because they help us understand the usage and implications of the auto-frame. Evoked senses could be considered additional information/sentences related to the auto-frame.

Secondary Evoked sentences (or senses) are senses that are less commonly related to core senses but in the presence of particular indicators could be intended. This class includes subtypes of the core senses.

Core Senses

When should different words evoke the same auto-frames, or which senses for the same word should be in the same frame? Synonyms should be included in the same frame. What about near synonyms? How near would they have to be? What about words with definitions that are a single word, a genus with no differentia, such as *purchase* as *buy*, *bicycle* as *bike*, or *machine* as *vehicle*? These are difficult questions for which there is no single right answer for all circumstances. This research aims to implement a practical solution based on observed patterns and on intuition.

Construction of an auto-frame will begin with a single core non-derived sense, to which related senses will be added. Each sense should only be a core sense for one auto-frame. Derived senses should be included as core senses in the same auto-frame as the base sense, such as *bank*, as in *a building containing a bank*, and *bank*, as in *financial institution*.

Dictionaries sometimes have different senses for words with different arguments but the same genus term and mostly the same implications; some dictionaries will contain distinct definitions for transitive and intransitive verbs. In *LDOCE* this can be seen with some of the definitions of *travel* in Table 5.11. It would be valuable to merge these senses into a single frame, as they share the same fundamental idea about *going* and the differences are small, secondary, and not mutually exclusive. Further, recall that coarse-grained sense can improve the performance of some systems on NLP tasks. This would also reduce the number of auto-frames, thus simplifying auto-frame sense selection.

#	Word	Definition
1	travel *LDOCE*, verb	to go from one place to another, or to several places, especially to distant places
2	travel *LDOCE*, verb	to go a particular distance or at a particular speed
3	travel *LDOCE*, verb	to move at a particular speed or in a particular direction
4	travel *LDOCE*, verb	to go from place to place to sell and take orders for your company's products
5	travel *LDOCE*, verb	to go very fast

Table 5.11: Definitions of *travel* from *LDOCE* showing common genus term or differentia

Two criteria were used for merging core senses into an auto-frame:

- derived senses were included as core senses;
- definitions for the same word with the same genus term are included as core senses. (For example, 4 of the senses of travel have *go* as a genus term).

Evoked Senses

Determining what should be evoked by a frame can be difficult, and may depend on the knowledge base's purpose. In this research, I consider two questions: *What is true or known about a frame? What might be true about a frame?*

The two questions above can be reformulated into more practical questions:

1. What concepts, senses, or statements are required to define the auto-frame?
2. What concepts, senses, or statements are defined using the auto-frame?

Consider the first reformulated question. The concepts and senses referred to by a definition must be true of the auto-frame. All the terms used in the definitions of core senses are expanded to their definitions. Thus the auto-frame includes (as evoked knowledge) the definition for each term in core sense.

Hypernym/Hyponym relations are the most used relations in WordNet. A genus term can effectively capture this relationship. Because of the importance of this relation,

expanding the genus hierarchy several times can be helpful. Genus terms are expanded to two levels.[10]

Expansion of a genus term allows for inclusion of its attachment points, which may be properties, arguments, or common sense implications. This was previously shown with *source* and *destination* roles for the word *ride* (genus term of *travel*) (see Section 3.5.1).

Likewise, *something moving* should inherit the property *speed*. Recall that some definitions are marked as properties, and what they modify has also been marked. Thus, if a property definition exists for one of the core senses, or its genus concepts, then it should be included in the auto-frame.

The second reformulated question, "what concepts, senses, or statements are defined using the frame", relates to semantic priming and what ideas, memories, and experiences are evoked by a word (Neely, 1976).

At times, an expression may suggest to a reader a particular circumstance or subtype related to an auto-frame. For example: "John was attacked by a large fish in the ocean." Many people would expect or assume the fish is a shark. Or "The buyer walked away satisfied." We might expect that the buyer purchased something and the purchase resulted in his satisfaction.

To allow for connection search to find and connect these ideas, their definitions must also be included in the auto-frame. If their definitions are disconnected from the context (there are no attachments), then connection search should prune them away.

When these additional senses are added to an auto-frame they will be referred to as *secondary evoked senses*. Any definition, which refers to a core sense for a given auto-frame, is then added to that auto-frame as a secondary evoked sense.

Secondary evoked senses can add a lot of information to the auto-frame, which can increase processing time, yet the more the difficult problem is "how can such additional information be evaluated?" It would likely require psychological or cognitive research data to model which distant associations human readers normally make quickly.

In summary, auto-frames include, as evoked senses, the following:

- Definitions for the genus concepts (expanded to two levels)
- Definitions of properties that modify the genus concepts
- Definitions of words from the core senses.
- Definitions that use (contain) one of the core senses (such as subtypes and parts)

[10]Expansion to two levels was chosen based on observation. It strikes a balance between expanded senses to include meaningful information, yet avoids overly general definitions, and limits the computation runtime.

Modulated Genus Terms

In some definitions, there are identifiable terms that may function as the genus if the terms were to be used with a different part of speech than originally intended. These additional genus terms may be marked as *genus* concept, as they are a possible modulated interpretation of the definition in some contexts.

For example, consider the definitions in Table 5.12. The words in red represent the actual genus term and words in blue represent words that could be genus terms, if the word were used as a different POS. $Bicycle_{\text{noun}}$ is defined as being something you ride, thus it seems fitting that inflecting *bicycle* as a verb would refer to this action, *riding*. A verb sense of *ride* specifically involves a bicycle, which could act as a possible modulated noun genus term for this sense of *ride*. While the genus term for $bicycle_{\text{verb}}$ in *LDOCE* is not technically *ride* as suggested above, the definition implies *riding*, which is the activity of *going* on a bicycle. Furthermore, *go* is the genus term of *travel*, and *travel* the genus term of *ride*, thus the suggested genus term and dictionary-supplied genus are related.

Word	Definition
Bicycle *LDOCE*, noun	a two-wheeled vehicle that you ride by pushing its pedals with your feet
Bicycle *LDOCE*, verb	to go somewhere by bicycle
Ride *LDOCE*, verb	to travel on a bicycle or motorbike

Table 5.12: Some expected example sentences for the bicycle frame from *LDOCE*

The example shows the potential for modulating genus terms by selecting terms from other parts of speech.

Rules for Modulated Genus Terms

A simple rule-based approach to finding alternate genus terms is used in this research, although it is expected that this could be better done using corpus-based methods.

Alternate genus terms are key terms in a definition, with different parts of speech than the word being defined, which the definition requires, and are usually specific to

the domain or context. Consider the verb sense of *bicycle* in Table 5.12: *somewhere* is not an alternate genus term because it is too general, while *bicycle* is fairly specific.

Given a noun definition *Def* for term T, and asuming there does not exist a verb definition for T, I use the following rule. If *Def* has a subordinate clause where the head verb/predicate, V, takes T, the genus term, or something coreferent to it as a dependent, then V is treated as a possible genus term.

Given a verb definition *Def* for term T, and where there does not exist a noun definition for T, I use the following rule. If there exists only one noun dependent on the *genus* verb, which is not in the top 1% most frequent nouns used in the dictionary, then the noun is an alternate genus term. Alternate genus terms are not marked when there is more than one candidate, although conjunctions are marked as a single candidate.

Constructing Auto-frames: Procedure

Before starting the construction of the auto-frames, it is beneficial to apply some WSD. Connection search can function as a form of WSD prior to building auto-frames, but it is also an integral aspect of building auto-frames.

Starting from a single non-derived sense, S, the steps to produce an auto-frame are as follows:

1. include senses for the same lexical units and identical genus terms
2. include derived senses
3. mark senses as processed
4. include lexical units (terms) for each of the core senses as terms for the auto-frame
5. attach included senses using connection search and expand senses
6. include properties of the genus concepts

Each of these steps is detailed below.

Include senses for the same lexical units and identical genus terms

Given a set of terms, $\{T\}$, which evoked the auto-frame, examine other definitions for each term in $\{T\}$, and if they have a genus term in common with the auto-frame, include them. For the previous example, this step would now include the multiple senses of *travel* with the genus term go. By linking the genus terms and running connection search, the two definitions will have common elements linked and expanded.

Let $\{S\}$ be the set of senses included into this auto-frame.

Include derived senses

Include all senses derived from a sense in $\{S\}$ that are not yet in the auto-frame. These senses should now be included in $\{S\}$.

Mark senses as processed

Each sense in $\{S\}$ should be marked as processed; these do not need a separate auto-frame.

Include terms for each sense

For each sense in $\{S\}$, include any terms that evoke these senses into $\{T\}$.

Attach included senses using connection search and expand senses

First, create known connections between senses, such as attaching a parent sense to the derived sense. Create a link between common genus terms that were added in the first step. Now, run connection search to find more connections and common elements between these senses. Connection search should be configured to expand genus terms two levels deep and other terms once.

Include properties of genus concepts

Find all the definitions of a *property* type which modify any genus concepts now in the auto-frame or any superclass of a genus concept.

5.3 Using Auto-frames: Entailment and Reading Comprehension

For auto-frames to be of some value, there must be a method of applying them to tasks. Yet the knowledge representation, MIG, differs from other common knowledge representations. For example, it supports multiple interpretations allowing terms to be only partially disambiguated. Auto-frames also differ significantly from other resources in that they are large graphs of information, instead of a sense and role inventory.

Subsection 5.3.1 describes a method for applying auto-frames to a limited task of entailment (narrowed to comparing predicates/auto-frames and roles/attachment points). This entailment system is used to determine if a sentence entails the answer to a question. Subsection 5.3.2 explains the method of answering reading comprehension questions.

5.3.1 Entailment

The entailment method has similarities with graph matching or with comparing predicate-argument structure. One MIG is considered the text (or known information/knowledge),

while the other MIG is the hypothesis (or query); thus entailment involves testing if the query is known or implied by the text.

For each surface concept in the query, there must exist a concept in the text with which it can be matched. That is to say, each concept explicitly stated in a query must exist in the text either explicitly or implicitly (in an expanded graph). Furthermore, the concepts must be of the same sense or coreferent with something of the same sense. Because auto-frames include expansions, particularly for genus terms, comparison of super-classes is achieved by comparing concepts connected by genus links. Finally, for each surface relation, an attachment should exist that connects the related surface concepts or their sub-graphs. A parallel attachment must exist in the hypothesis; that is, it must connect the concepts from the hypothesis matching the ones in the text using the same attachment point.

Consider the following example.

query: *John gave Mary money*
text: *John bribed Mary.*

For reference, the definitions of *contribute, bribe* and *pay* have all been listed in Table 5.13. In the text, *bribe* would be expanded to a graph similar to *John paid Mary money to persuade her.* Then *paid* would be expanded to *John gave Mary money for something.* "John gave Mary money" (query) matches with the expanded (evoked) understanding of the text.

A different query, *John contributed money to Mary*, would evoke *John gave Mary money*, but it would not entail the text using this method, because nothing evoked by the text matches the surface sense *contribute*. The important distinction is that the senses of the surface level concepts in the query must exist in the evoked MIG of the text. The auto-frame for *bribe* includes the definition of *give*, thus they entail, but the auto-frame for *give* does not include the definition for *contribute*.

Since textual entailment is primarily about "ordinary understanding"[11] (Voorhees, 2008), it seems reasonable to allow these to match. *Contribute* expands to *giving*; to allow for partial matching, we could allow matching of expansion graphs in the query instead of just the surface graph. This partial matching would allow "John contributed money to Mary" to match with "John bribed Mary". This partial matching has been included in this system.

[11]See Section 3.11.1 for more details.

Word	Definition
contribute	to give money, help, ideas etc. to something that a lot of other people are also involved in
bribe	to pay money to someone to persuade them to help you
pay	to give someone money for something

Table 5.13: Example definitions for *contribute, bribe* and *pay*

A number of unit tests were used to validate the entailment system under the conditions of an evolving code base and knowledge base. The tests were added as examples of expected behavior, whenever the entailment system was changed to add new functionality. Table 5.14 shows the sentences tested for entailment as unit tests. "Entails" indicates if the sentences are expected to entail. "Mirrored" indicates that bi-directional entailment was tested; "no" simply means that bi-directional entailment was not tested, but makes no claim about whether the alternate direction should entail. "Success" indicates whether the system passed or failed the test.

Note that the incorrect entailment of *toilet* and *store* happened because a toilet can be a room or building and a store is building. Likewise, the other failures are caused by the knowledge found in or absent from *LDOCE* and as a consequence, the knowledge base.

5.3.2 Reading Comprehension

In reading comprehension tests, a text is read then questions about the text are answered to demonstrate understanding. Reading is implemented as parsing and transforming the text, then applying connection search to add auto-frames, create attachments, and pruning away any disconnected (or low-ranked) parts of the auto-frame. The entire story is stored in a single large MIG; because this is a single MIG, coreference links and attachments can be maintained across sentences. Coreference across sentences exists where the parser identified them.

Questions are restructured so that they are better suited to entailment in the original text. The complete list of transformations can be found in Appendix B.2.

The transformations remove the *WH* part of the question, often replacing it with a concept to facilitate matching. In addition, a concept is marked as the expected answer.

Entails	Mirrored	Success	text	hypothesis
yes	no	pass	man	someone
yes	no	pass	man	runner
yes	no	pass	man	person
yes	no	pass	John	person
yes	no	pass	London	place
yes	no	pass	to travel	go
yes	no	pass	John in the band	Who in the band
yes	no	pass	John is in the band	Who is in the band
yes	yes	pass	John travelled to the store	John went to the store
yes	yes	pass	John bought a guitar	John purchased a guitar
yes	yes	pass	John paid his teacher.	John bribed his teacher.
yes	no	pass	John cooked his steak.	John cooked his food.
yes	no	pass	John fried his steak.	John cooked his meat.
no	no	pass	John travelled to the store	John ate the store
no	no	pass	John like guitars. He likes to shop. John eats food.	John ate the store.
no	no	fail	John travelled to the store	John went to the toilet
no	no	fail	John wrote a book	John wrote a song
yes	yes	fail	John wrote a book	John created a book
yes	yes	fail	John wrote a book	John wrote a story

Table 5.14: Sentences used to validate the entailment system

Consider the question "Who is John". This could entail all references to *John*, although there could be many possible candidate answers, few of which would describe *John*'s relevance. Instead, the query graph is restructured into the form "John is person", and the entailment system searches for text that describes *John* as a person, or as a subtype of person, such as president, sailor, or writer. The word matching *person* is expected to be the answer. The patterns were developed in an ad-hoc manner, while attempting to re-formulate questions so that answers could be found.

Once the question has been parsed and reformulated, it is also run through connection search to assign auto-frames and expand the query MIG. The query MIG and the text MIG are given to the entailment system. The system attempts to find a sentence which matches the context of the query. It returns multiple matches in the same order in which they were matched. It attempts to match surface level concepts first, then processes deeper concepts.[12] The first match tends to have the closest matching senses (because

[12] "Deeper concepts" refers to concepts within expansion graphs

they were closer to the surface), while later matches require traversing one or more expansions to find a matching sense.

The system returns the sentence which matched the query and a mapping between query nodes and text nodes. Using this mapping, a query concept, which was marked as the exact answer in the transformation step, can be mapped to an answer in the text.

The results of reading comprehension evaluation and examples of the data can be found in Section 7.3 *Extrinsic Evaluation: Reading Comprehension.*

5.4 Summary

This chapter introduced the methodology for parsing and processing definitions, connecting words and senses in context, and applying MIGs to reading comprehension. The first step was to build a knowledge base, obtaining good parse trees and annotating the knowledge. Subsequently, the most important algorithm of this research was introduced, connection search, which is used to expand words to their senses and then attach them to their context. Connection search also includes a pruning step which removes low-ranked senses and attachments from the MIG. The construction of auto-frames was explained, which also depends on the use of connection search. Lastly, the methodology for applying MIGs to reading comprehension was introduced. In the next chapter, we will see the system design and discuss some key challenges related to designing and implementing these algorithms.

Chapter 6

System Overview

This chapter describes the overall design of the system for extracting auto-frames and key operational details of some of the components. It should be noted that one of the largest challenges in implementing this system was keeping the runtime of components to less than a week; after optimization many components ran in a few hours instead of several days. This chapter describes a number of the design decisions and implementation details that decreased the runtime from several weeks to days or hours depending on the task.

6.1 Overview

The implementation is composed of eight high-level modules: MIG, Knowledge Base, Parser Server, Post-Parsing, Connection Search, Creating Auto-frames, Entailment, and Remedia. Figure 6.1 shows these modules and their dependencies. There is another component not included in this model, which is the graphical knowledge base browser and parse tree viewer; it enables viewing the output of the parser and various post-parsing operations, as well as browsing data in the knowledge base. This browser is described in Appendix D.

The MIG module contains the implementation of the knowledge representation, which all other modules depend on. MIGs can be stored in memory or saved in the knowledge base. The knowledge base is designed to store sentences and multiple versions of MIGs (from parser server, post-parsing, labeled, and auto-frames) for these sentences. The knowledge base can also store lexical units, senses, definitions[1] and all the associations

[1]Definitions are stored as sentences which are attached to a sense.

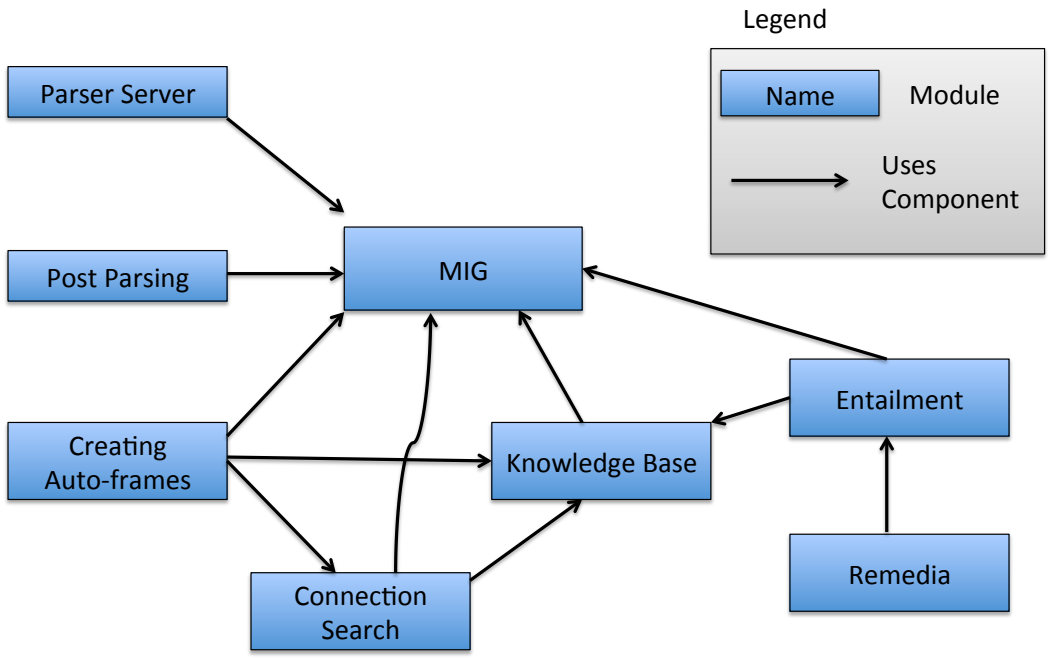


Figure 6.1: The System's Modules

between these concepts.

The parser server is a separate process, which manages pools of parsers, keeps them alive as needed, and restarts failed parsers. Post-parsing, which was defined in Sections 5.1.2 and 5.1.3, transforms Multiple Interpretation Graphs (MIGs) returned by the parser into a more useful representation and annotates things like genus concepts and definition types. Parsing and post-parsing are distinct steps so that errors in the representation can be traced to the parser or the transformation process.

Connection search is the implementation of the key algorithm in this research, which is used for constructing and applying auto-frames. Creating auto-frames is the process of selecting and merging senses into an auto-frame. This process relies on accessing the knowledge base and on the connection search algorithm. The description of the

connection search algorithm and the creation of auto-frames can be found in Section 5.2.

Entailment is a simple ad-hoc system used to match query MIGs to text MIGs. This system is then applied to the reading comprehension task (Remedia Component). These systems have been described in Section 5.3.

6.2 MIG

An MIG contains a list of surface graphs (any graphs that exist before expansion). A map of link paths[2] to graphs used within the MIG (*i.e.*, the surface graphs and any graphs added by expansion). This allows rapid lookup of any expansion related to a specific concept or path.

An MIG contains a list of all links within it, such as coreference, clausal, or genus links. An MIG also contains a set of interpretation restrictions, which themselves contain interpretation sets. See Section 4.4.2 for more information on MIG, links, link paths, and interpretation restrictions.

Lastly, the MIG contains some caches to enable rapid lookup, such as a cache of all relations and concepts from all the contained graphs.[3]

6.3 Parser

This system uses the Stanford Pipeline from CoreNLP (de Marneffe et al., 2006; de Marneffe and Manning, 2008). There were a number of difficulties related to the parser's performance and stability. It should be noted that the parser is reasonably fast, parsing most definitions in about half a second, though some take closer to one second on a 2.8 GHz Core i7 processor. Loading a new parser takes approximately 15 seconds. These are very reasonable performance times, however, when parsing 65000 definitions 2 to 4 times each, the runtime becomes quite long. A quick calculation assuming single-threaded processing, the minimum parse time of 0.5 seconds, only 2 versions of each definition (with and without prefix), and no overhead, reveals that it would take about 18 hours to parse all sentences.

[2] A link path is an ordered list of links. In this case the links are only genus and predicate-argument links, which connect concepts to their expansions.

[3] An earlier implementation of MIG used a single large graph to which each new expansion was cloned and added. This created large graphs that were difficult to traverse, required a significant amount of memory, were slow to prune, and had a host of other management problems. The current design minimizes redundant information by re-using graphs within the MIG.

Beyond the total runtime there were a few other concerns with the parser:

1. The parser would occasionally crash, which would require a restart of the parser and would not necessarily release resources.
2. The parser would exhaust all memory (out-of-memory error).
3. The parser would hang, stall, or have extremely long runtimes.

An out-of-memory error would seem to be easily fixed by adding more memory, but this would often lead to the third consequence, a stalled parser. Ending a stalled parser thread is possible, but the resources would not always be released, possibly leading to an out-of-memory error. Also, note that once a process is out of memory, the process cannot recover, so it terminates.

The solution was to run a parser server from which parsing could be requested; the server would always be available and could hide failures. The server delegated parsing a small set of parsers running as separate processes. If a parser process took too long, it could be terminated and a new one started; likewise if conditions 1 or 2 occurred, the parser could be terminated and restarted. The server managed all these conditions while the main system would simply get back a dependency parse tree.

Figure 6.2 illustrates all the parser processes. The system process has many threads, which independently connect to the parser server and request text to be parsed. The parser server sends these requests to the parsers, which it manages. When a parser process completes parsing, it returns the result to the server, and then the server returns this to the system.

6.4 Post-Parsing

Details of parsing and post-parsing have been described in Section 5.1. Parsing, post-parsing, and saving MIGs to the knowledge base are done together in a multithreaded system. Parsing is a CPU- and memory-intensive task, while post-parsing requires little time or resources, and saving to the knowledge base is primarily I/O bound. Parsing, post-parsing, and saving on a 2.8 GHz 8 core system, using 4 parsers and 10 threads takes 415 min. This processing time has been reduced from the original unoptimized time of several days.

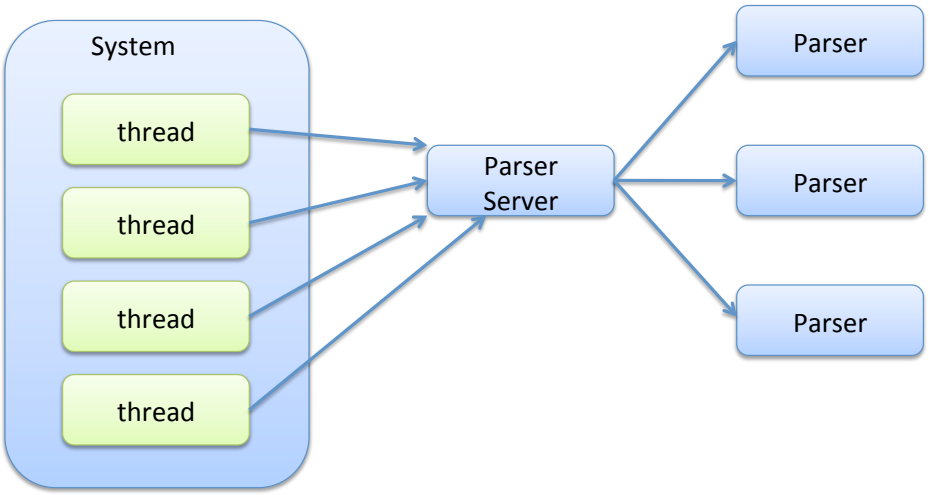


Figure 6.2: Parsing Model

6.5 Knowledge Base

The resources used and experimented with during this research were parsed into MIGs. Rapid searches through these MIGs can be useful for research and task-based work. Recall that in Section 5.1.4 definitions defined using one another (bi-directional definitions) were used for disambiguation. This task initially took close to a week to run, because for each definition many other definitions needed to be searched. The complexity of the task was $O(n^2)$ where n is the number of definitions in the knowledge base. Using the database query system, this task requires less than 45 minutes.

To facilitate rapid searches, the MIGs are stored in an indexed MYSQL database. The database contains documents,[4] sentences, MIGs, senses, and terms. Terms (from dictionaries) may be associated with one or more senses and senses may be associated with one or more sentences (which are expected to be the definitions and/or examples). Sentences may be associated with many graphs; the expectation is that different processes (transformations and knowledge extraction) may produce different graphs.

A number of methods exist for rapid graph searches (Milo and Suciu, 1999; Williams et al., 2007; Zhang et al., 2007; Escolano et al., 2011; Yan et al., 2004), though most focus on following paths. The method presented here is simple, intuitive, easy to imple-

[4]The document structure was used to support Question-Answering (QA) and to support the use of different knowledge bases. The majority of the development included parallel development of the *WordNet* and *Longman Dictionary of Contemporary English* (*LDOCE*) knowledge bases.

ment, and is not limited to path-based searches. It provides performance much faster than sequential search, though it is not appropriate for real-time systems. Appendix C describes how good search performance was achieved.

Table 6.1 shows the search times for a few sample queries. The queries are from the *LDOCE* database, containing 50,000 graphs. The queries are presented in natural language, but are submitted in a form similar to:

```
[concept] -> (relation) -> [concept] ->(relation) -> ...
```

The underscores "____" are used to mark concepts or relations with unrestricted types in the query. Queries 6 and 7 each have an anonymous concept and relation. Queries 8 and 9 represent potential worst-case searches, though they are not realistic queries; no practical query will contain only "the" (the most frequent word in the database) and anonymous concepts and relations (which require joins with complete tables). Searches with less frequent or more specific concepts should be faster.

	Query	Search Time	results
1	cat	409ms	58
2	the	1521ms	21651
3	sonic_screwdriver	19ms	0
4	animal like mouse	662ms	6
5	____ in house	2110ms	80
6	Tricorder ____ ____	34ms	0
7	____ ____ person	2434ms	1222
8	The ____ the	9339ms	10
9	The ____ ____ ____ the	824ms	0

Table 6.1: Search time of sample queries in *LDOCE* graph database

The expectation is that during knowledge extraction it will be beneficial to find all graphs where a pattern exists. For example, consider finding all animals (or things) that have fur. Another use could be to determine if in a particular corpus certain concepts are frequently, or with high probability, related to other concepts. For example, consider finding what foods and how frequently each food is eaten by an animal. Another use would be to determine the set of all things that satisfy a requirement: for example, consider finding the set of all things that can be "driven".

6.6 Connection search

The original implementation of connection search took between 3 minutes and 1 hour to process a single sentence, but has since been optimized to less than 1 minute per sentence. A processing time estimate based on the Remedia data would suggest that the average time per sentence is less than 45 seconds.

The main bottlenecks in connection search were the constant queries for senses, MIGs, or auto-frames from the knowledge base. Each queries had a very small lag (delay) measured in milliseconds, but due to the number of queries, this delay would accumulate to a longer time. By adding client-side caching of these values, the runtime was drastically reduced. The more efficient implementation of MIG also improved the performance of connection search.. Connection search requires traversing the MIG and expands it by adding in auto-frames or definitions multiple times. The original implementation would duplicate the definitions and auto-frames during each expansion, but the current implementation allows the graphs to be shared. The sharing of graphs reduced the runtime of adding to the MIG, traversing of the MIG, removing of graphs from the MIG, and the overall memory footprint of the system. Running connection search for the purposes of Word Sense Disambiguation (WSD) on all of *LDOCE* with two levels of expansions takes about 3 days to complete.

6.7 Creating auto-frames

The procedure implemented in this module, described in Section 5.2.4, is composed of two simple parts: select core senses and evoked senses. Connection search is applied to connect senses and expand some of the concepts and evoked senses. The merging of senses in this algorithm is fast (measured in hours), however the connection search component takes days to complete. Because of this long runtime, auto-frames are usually created in a lazy manner, as needed.

6.8 Entailment

Four different entailment systems were implemented throughout the course of this thesis. Two early prototypes used for question-answering were written in Prolog. The lexical knowledge base and sentences would be converted into rules and facts, then the program

would attempt to prove entailment. While many good things can be said about the use of XProlog, these implementations were too slow to be practical.

The third system was written in Java using insight gained from implementing the systems in Prolog. This system was intended to be applied to reading comprehension instead of question answering. This system still proved slow, but it was faster than previous implementations and did not require changing the format of the knowledge base.

The current implementation is the evolution of the third system. Managing the runtime of the entailment system has been achieved by multi-threading the task and considering multiple interpretations simultaneously. The entailment system stores states, which describe partial matches between the query and the text MIGs. Multiple threads are used to match and update these states, until there is a complete match (a sub-graph of the text that matches the query graph) and the system stops.

Previous entailment systems were slow because they searched for matching sub-graphs in each individual interpretation in the MIG. The current system allows matches to include multiple interpretations simultaneously. Consider the sentences "John went to the store" and "John went to the shop". There are several senses for "go" which could apply in this context, thus there are several parallel interpretations, which could match. However, entailment does not require testing each interpretation separately, it is enough to know that the concept "go" matches for a set of senses and thus a subset of interpretations. Likewise, there may be one or more senses for which "store" and "shop" could match, but for successful entailment there must exist a set of one or more interpretations where all the concepts and attachments are matched.

6.9 Remedia

The Remedia module loads the individual stories and questions. For each question, it uses the entailment system to find a sentence containing the answer in the text. It then verifies that the sentence is correct against the answer key. Running this process on 58 documents (∼300 questions) takes 70 minutes.[5]

[5]Using 2.8 GHz Core i7 processor with 16 GB of memory.

6.10 Java Virtual Machine

There were a number of performance optimizations that could be made just by choosing the right settings for the Java Virtual Machine (JVM). In Java 7, the 32-bit JVM was approximately 20% faster than the 64 bit version. This was particularly noticeable during parsing. Oracle confirms the performance difference.[6] Two things can be done to mitigate this issue: the parser and processes that use less than 4GB of memory can run in 32-bit mode, or the JVM can be set to run with the " -XX:+UseCompressedOops" flag which should mitigate the performance loss in 64-bit mode.

Another noteworthy JVM setting, applicable to most modern machines, is the "-XX:+UseNUMA" flag. It enables use of the NUMA (Non-Uniform Memory Access) architecture on the CPU. This system is run on a Core i7 processor, which has a NUMA architecture; this flag can boost performance by up to 40%.[6] The NUMA architecture allows multi-core systems to access memory at different speeds. In particular, memory local to the core can be accessed more quickly than memory local to another core.

Other important settings which can significantly affect performance are the choices of garbage collectors (young and old). While this topic could require quite a bit of discussion, the summary is that single threaded garbage collection is generally acceptable for single threaded processes. Parallel garbage collection is ideal to maximize throughput. Concurrent garbage collection seems appealing, though it is actually slower than parallel garbage collection, but it is probably best suited to situations that need quick response time and not optimal throughput.

6.11 Summary

This brief chapter focused on the design and challenges of implementing this system. The proof of concept system went through numerous iterations of refinement and experimentation. Many of the implementations seemed to have promising results but impractical runtimes. The high computational complexity of the algorithms would compound any small runtime delay into long days or weeks of processing. Many of the performance mitigation techniques are described in this chapter, as well as the design of some of the re-usable components. The development effort and skill required for this proof of concept was far greater than expected. The next chapter evaluates auto-frames comparing them

[6] `http://docs.oracle.com/javase/7/docs/technotes/guides/vm/performance-enhancements-7.html`, last accessed on April 26, 2016.

with *FrameNet*, and then applies the knowledge base to reading comprehension.

Chapter 7

Evaluation

It can be difficult to evaluate a knowledge representation. What defines a successful representation? This chapter is divided into two parts: an intrinsic evaluation of auto-frames and an extrinsic evaluation of auto-frames, as applied to the task of reading comprehension.

The intrinsic evaluation starts with converting post-processed graphs into Logic Form (LF) (Rus, 2004) and validating structure and processing initially done on the definitions. We continue the intrinsic evaluation with an inspection of some auto-frames and comparison with *FrameNet* frames.

The extrinsic evaluation applies auto-frames and connection search to the task of reading comprehension, using the Remedia data set. A number of configurations are considered. The system shows moderate precision, yet low recall. The results show that the use of connection search increases precision and that carrying multiple interpretations into the task can increase recall. It is also shown that, while the entailment and reading comprehension system are ad-hoc, they show improvement over a simple lemmatized and dependency-based evaluation.

7.1 Intrinsic Evaluation

The intrinsic evaluation begins with an empirical evaluation of parsing and preprocessing of the dictionary definitions. While this evaluation does not concern auto-frames, it suggests a success of all the effort of preprocessing the definitions and attempts to improve and restructure definitions. Subsequently, genus detection is evaluated, which is also measured empirically and has good performance. Selected auto-frames are reviewed

and compared with *FrameNet*. Lastly, the modulation examples from Section 4.6.2 are reviewed to verify that the system achieved its design goals.

7.1.1 Evaluation: Comparing against LF

It is difficult to evaluate tdirectly he auto-frame representation proposed in this work. There is no gold standard available for this representation. Furthermore, any comparison to related work is difficult, because some of the resources are unpublished and the representations differ significantly. However, LF is a simple representation with gold-standard resources. The post-processed graphs can be simplified to LF: mapped onto LF by a series of transformations. Appendix A.1 describes LF in detail, while Appendix A.2 describes the transformation (mapping) process.

This provides a comparison of our intermediate knowledge representation to the knowledge representation in (Rus, 2004). Rus's LF representation is highly simplified and ignores many issues. A number of criticisms of LF, beyond just the over-simplification, can be found in Appendix A.3.

This comparison is of limited value, because LF lacks so many interesting details, and because this is an intermediate representation in this extraction process. The comparison does highlight that the overall structure is consistent with the one Rus had annotated for his task. This suggests that the basic units of an auto-frame, the graphs which represent statements or propositions, are well structured.

Gold Standard Data

Senseval 3 hosted a task for Logic Form Identification (LFi).[1] Participants were to extract LFs for a set of sentences. The task produced an evaluation of various LFi methods, in addition to training and test data sets for future research (Rus, 2004).

The system from the University of Sydney transformed dependency parses into logic forms (Anthony and Patrick, 2004). This system selected arguments for each predicate by consulting an ordered list of dependency relations; the identifier of the source or target of the first matching relation from the list was used as the argument. It used a similar transformation-based approach to convert parse trees into LF. This approach is fast and easy to implement, though it depends on the insight of the implementer and not a machine-learning process.

[1] LFi is the acronym used at Senseval 3 for the task. See Rus (2004).

Bayer et al. (2004) from MITRE also used a dependency parser, but added several layers of logic processing to the parse tree. Given their results, the additional layers of information did not seem to provide significant improvement. The authors provide clear criticisms on the effectiveness of LF; in particular they point out deficiencies in the representation of adverbs that modify adjectives. Furthermore, they point out the inconsistencies of representing noun-noun compounds differently from adjective-noun expressions.

Ahn et al. (2004) from the University of Amsterdam also started from a dependency parse tree to create the logic forms. Their system traverses the tree to create the appropriate predicate for each node. They note that some of their system's errors occurred because of inconsistencies between the training and the test data, or because some phenomena were not well represented in the training data.

Rus's (2001) dissertation explains his method, which starts from syntactic parse trees. The part-of-speech tagging and parse trees are constructed through a voting scheme of multiple agents, to increase accuracy. The resulting parse trees are simplified so that fewer rules are required for transformation (*i.e.*, more generalized rules are used); otherwise as many as 10,000 rules may be needed. The part-of-speech tags are simplified, plurals become singular (*i.e.*, NNS $\rightarrow$ NN) and verb tenses are ignored (*i.e.*, VBG $\rightarrow$ VB, VBZ $\rightarrow$ VB, etc.). The parse tree is simplified, mapping complex structures into simpler ones. Finally, a set of general transformations are applied to convert the simplified tree into a LF. Rus's work is different from others, in that he starts from POS tags and constituency trees. A voting scheme among many parsers was used to produce highly accurate POS tags and parse trees.

Wenner (2007) from Lund University presents a system that obtains reasonable performance, but with lower complexity than other systems; this system was developed after the Senseval 3 competition. The system uses MXPOST (Ratnaparkhi, 1996) for POS tagging, MALTPARSER (Nivre et al., 2006) for dependency parsing, and WordNet for morphological parsing.[2] The results were comparable to those at Senseval 3, though not all metrics were published.

Evaluation

The syntactic logic form is evaluated using the data (and scripts) from the Senseval 3 Logic Form Identification (LFi) task. There are 50 development examples and 300

[2]Many WordNet interfaces and packages include a morphological parser for stemming and lemmatization.

sentences in the test set. The system developed and compared in this section was created only considering the definition of LF and the development set.

There are two levels of evaluation: argument level and predicate level. Argument level requires the hypothetical and expected predicates to match, but only evaluates which arguments match. The evaluation script does not require the argument identifier (the textual realization of the argument) to match, but instead evaluates if the argument is connected to the correct predicates. The argument-level evaluation measures both precision and recall by taking the total number of correctly identified arguments and dividing it by the total number arguments, either assigned or expected depending on metric: precision or recall, respectively. Predicate-level evaluation requires that the predicate and all arguments are correctly assigned. Thus predicate-level precision and recall are the number of correctly assigned predicates divided by the total number of assigned or expected predicates, respectively.

Argument level should be considered optimistic as it rewards partially correct predicates, while predicate level requires all arguments to match for the predicate to be considered correct.

Table 7.1 repeats the results of the teams from Senseval 3, plus one system that did not participate in Senseval, and includes the results from this work. The results from this research are good in all metrics, usually the second highest by a significant margin. The difference between the f-measures (argument level and predicate level) of this work and the next best system is statistically significant when tested with a t-test at a 95% confidence level. Given that this system took less than one month to develop, the results are acceptable. The same results are also shown as charts in Figures 7.1 and 7.2.

	Argument Level			Predicate Level		
Group	Precision	Recall	f-score	Precision	Recall	f-score
University of Amsterdam	0.729	0.691	0.709	0.819	0.783	0.801
Language Computer Corporation	0.776	0.777	0.776	0.876	0.908	0.892
MITRE	0.734	0.659	0.694	0.839	0.781	0.809
University of Sydney	0.763	0.655	0.705	0.839	0849	0.844
University of Lund	-	-	0.705	-	-	0.844
This work	0.762	0.740	0.751	0.875	0.884	0.879

Table 7.1: Comparison of syntactic representation against Senseval 3 Logic Form Identification task

The results on the development data were high, but imperfect. The development results are presented in Table 7.2 with the cause of each error discussed below. Most of

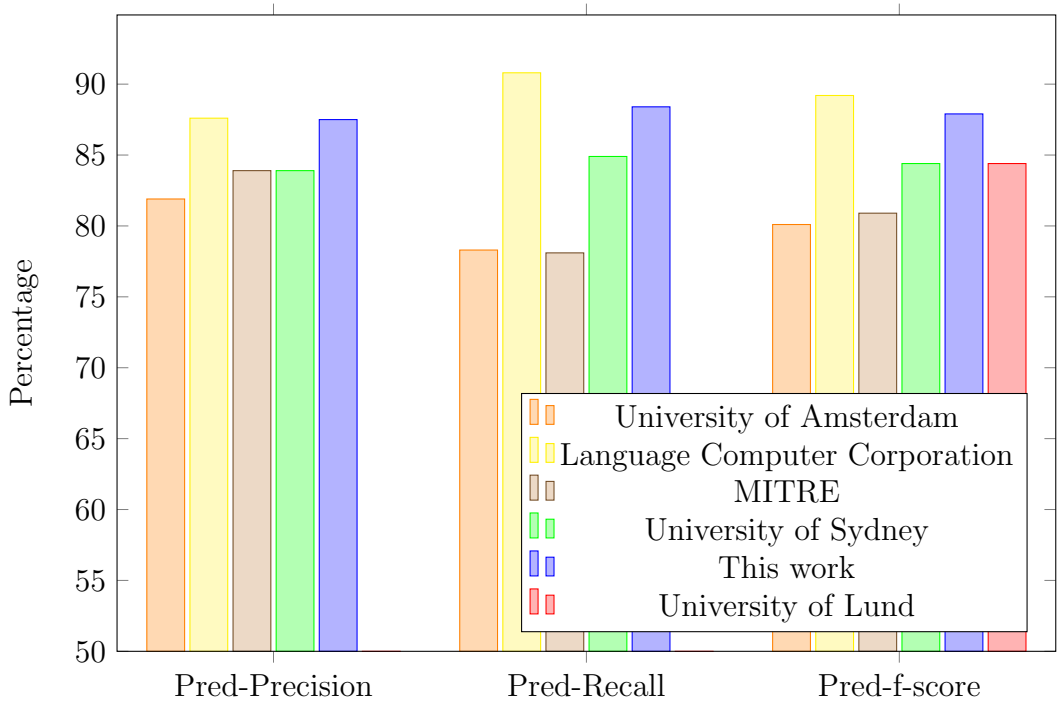


Figure 7.1: Predicate-level comparison of mapping to LF

the errors occurring in the development data are small disagreements that do not change the semantics or the logic of the representation. Below is a detailed analysis of the errors made by this system on the development data.

	Argument Level		Predicate Level	
Group	Precision	Recall	Precision	Recall
This work	0.843	0.805	0.923	0.904

Table 7.2: Results on development data from Senseval 3 Logic Form Identification task

POS Tagging Errors The 8 parts-of-speech tagging errors were outside the scope of this effort (*i.e.*, they come from the parser), though in some cases the problem can be detected and corrected. In all but one case they do not affect the interpretation; ultimately the predicates and arguments match except the POS, which is not actually included or considered in the test data.

Collocation or Noun Compound Errors These were 7 disagreements between the system and the evaluation data about whether a series of words is a collocation, a noun compound (complex nominal), or just an adjective followed by a noun.

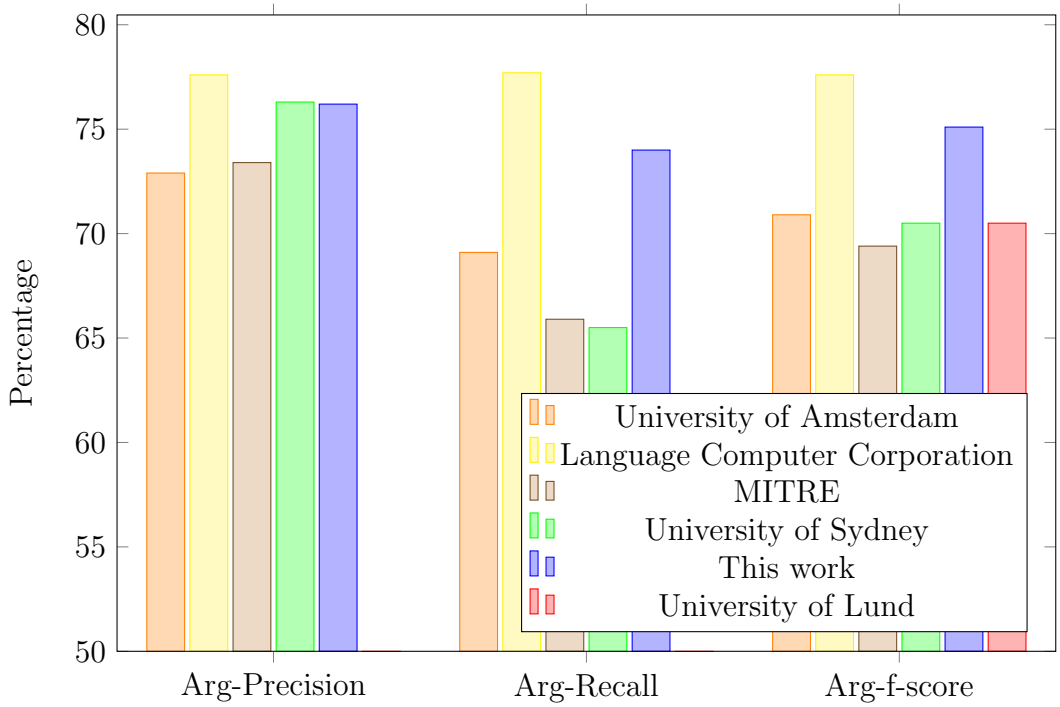


Figure 7.2: Argument-level comparison of mapping to LF

Collocations, noun compounds, or adjective and noun pairs all produce a single argument representing the joint concept. The differences are in how the concept may be broken into smaller concepts. It seems inconsistent and sometimes arbitrary that the LF representation has different representations for each of these multi-word expressions.

Lemmatization Errors There were 4 lemmatization errors, which seem to be errors in the development data. Four verb predicates have not been lemmatized, while all others have been. Unless there is a special reason why these are not lemmatized, nothing can be done about these errors.

Preposition Compound Error There was one error where a preposition compound was not detected. The expression "as if" should have been represented as "as_if(x1)". Our system considered "as" to be an adverb and "if" was stripped out as a determiner.

Relational Error The relational error was a difference in the assignment of a relation (*i.e.*, an argument to a predicate). The dependency tree did not attach a relation to the same concept that the development data expected. This error was either

from the parser or the development data, but either way was outside the scope of this work.

Case Difference This was a trivial difference where this system failed lemmatization of the word "Englishmen" and made the letters lower case.

Argument Count Error There should be two arguments to the word "that", but our system produced no arguments. It considered the word a determiner, which should be stripped out, but as it was the subject of a verb, it remained as a concept/predicate. The rules in the system were unable to process it.

7.1.2 Genus Detection Evaluation

In *Extended WordNet*, there exists a LF for each gloss. Each LF is annotated with a quality identifier: gold, silver, normal. It is not entirely clear what each indicator means. Since some LFs were generated by hand, some were manually corrected, and yet others were fully automatically generated, we could assume this progression matches that of the quality. Whatever the true meaning of each quality indicator, it is clear that the gold quality should be of the highest confidence.

The LF data from *Extended WordNet* can be used as a training, development, or evaluation set for genus detection. Each synset (sense) in *Extended WordNet* has an LF of the form:

[head word predicate] →[LF of definition]

such as

```
house:NN(x1) →dwelling:NN(x1) serve:VB(e1, x1, x26) as:IN(e1, x2)
living:NN(x2) quarters:NN(x3) for:IN(x2, x4) one:JJ(x4) more:JJ(x4)
family:NN(x4)
```

which is the LF for the definition of *house*: a dwelling that serves as living quarters for one or more families.

The genus terms for verbs and nouns can easily be found by matching the identifier of the head predicate with a predicate in the definition. In the previous example, *dwelling* is the genus for *house*. In the case of adjectives and adverbs, it is easier to find the relation connecting it to the noun or verb being modified. This can be seen in the following example, where the adjective *fuzzy* modifies some noun with an identity of $x1$. In the definition the modified noun never appears but $x1$ is the subject of the verb *cover*.

```
fuzzy:JJ(x1) →cover:VB(e1, x1, x3) with:IN(e1, x2) fine:JJ(x2)
light:JJ(x2) hair:NN(x2)
```

I used the first 1000 gold quality synsets from each part of speech (POS) (nouns, verbs, adverbs, and adjectives) from *Extended WordNet* as development data for the genus detection algorithm. Most of the examples in *Extended WordNet* are of incomplete sentences, requiring prefixes; thus the resulting method was developed for this incomplete sentence style of definitions and not necessarily for other styles. The evaluation set used all synsets with gold quality LFs, including the synsets from the development set. It should be noted that many definitions have multiple genus terms, often connected by a conjunction (as seen in the previous section and illustrated in Figure 5.9)

As a result of differences in how multi-word expressions are handled, the evaluation only requires matching one word in a genus term that is multi-word expression. For the genus evaluation, I do a two-level evaluation: synset level and term level. Term level compares terms for matches. If a synset has no genus term then it makes no contribution to a term level evaluation. A synset with two genus terms (either in different sentences or in a single sentence connected by a conjunction) will contribute twice to the evaluation. The term level evaluation produces a confusion matrix, which indicates the number of terms correctly detected (True Positives), terms missed (False Negatives), and the terms incorrectly detected (False Positives).

The synset level compares terms, but evaluates all genus terms for the synset as a whole. The synset-level evaluation is an accuracy-like measure which indicates the percentage of synsets had correctly assigned genera. The evaluation metrics used for each level of evaluation are described in Table 7.3.

Table 7.4 shows the evaluation results of the development and testing datasets for all metrics. It also includes division by POS and totals for the test data. We can see that the error ratios are quite consistent between the development and the test data, meaning that the development data are probably quite representative of the test data. Table 7.4 may be a little overwhelming, thus the test results are also presented in a bar chart (Figure 7.3) of percentages. As it is clear from the bar chart, there is a particular weakness in detecting the genus of adjectives. Other than with adjectives, the genus term detection has very high precision.

Metric	What it means
SC	(Synset Correct) all genus terms where correctly identified for a given synset
SPC	(Synset Partially Correct) some genus terms where correctly identified for a given synset
Total	Total number of synsets processed
GTC	(Genus Term Correct) the number of terms that matched (*i.e.*, true positives)
GTW	(Genus Term Wrong) the number of terms that were unmatched (*i.e.*, false positives)
GTM	(Genus Term Missed) the number of terms that had no word for comparison (*i.e.*, the number of detected genus terms was less than the number in *Extended WordNet*) (*i.e.*, false negatives)
GTX	(Genus Term Extra) the number of extra terms detected (*i.e.*, the number of detected genus terms was larger than the number in *Extended WordNet*) (*i.e.*, false positives)

Table 7.3: Explanation of Metrics for Genus Detection Evaluation

Data	SC	SPC	Total	GTC	GTW	GTM	GTX
Dev Nouns	752	759	1000	741	8	249	2
Dev Verbs	711	729	1000	816	52	295	3
Dev Adjective	518	719	1000	763	286	694	98
Dev Adverb	832	891	1000	985	89	226	77
Test Nouns	19557	19850	26114	19659	262	6650	105
Test Verbs	9162	9534	13485	11259	816	4447	49
Test Adjective	7387	9475	14298	9354	4691	9531	1861
Test Adverb	3219	3355	3664	3700	215	558	138
Test Totals	39325	42214	57561	43972	5984	21186	2153

Table 7.4: Evaluation of genus detection against Gold LFs in *Extended WordNet*

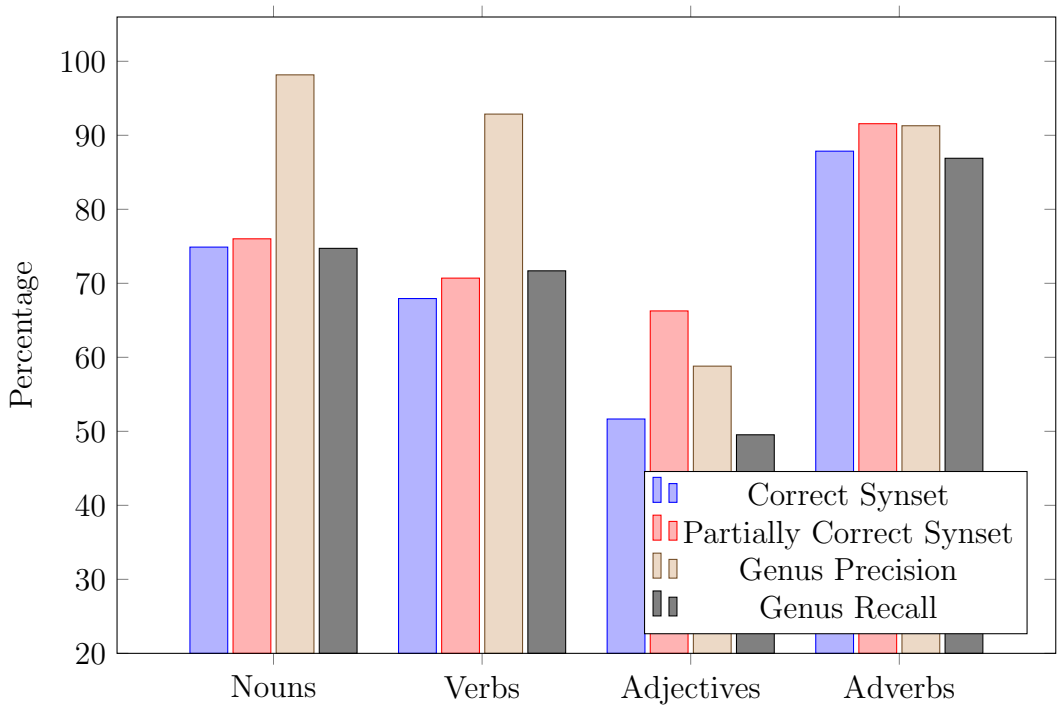


Figure 7.3: Percentage evaluation of genus detection against Gold LFs in *Extended WordNet*

7.2 Inspection of Auto-frames

This section examines a number of auto-frames and contrasts them with *FrameNet*. The goal of this review is to understand the strengths and weaknesses of auto-frames and their construction. Furthermore, I can evaluate to what degree the original expectations and goals of the research were met. This understanding may also suggest where improvements are needed.

It is obviously not possible to consider all words and meanings neither when developing a general methodology, nor during the evaluation. I will consider several examples which I feel are sufficiently representative.

The first auto-frame I will examine is *bicycle*. It has been consistently considered throughout the thesis, it has been used as an example of contextualization, and there are corresponding frames in *FrameNet*. I will then quickly examine the auto-frames for *piano*, *airplane*, and *bribe*.

Next, *bank*: it is a well known and often used example of a polysemous word and a homograph; one can observe the division of senses into auto-frames for this word. Homographs are expected to have their own auto-frames, while polysemous and polylexic senses are expected in the same auto-frame.

The last review in this section will consider a random selection of lexical units from *FrameNet* and *Longman Dictionary of Contemporary English* (*LDOCE*) (auto-frames). In this review, I will consider the generality or specificity of the frames and see if they contain comparable frame elements. I will conclude this comparison by highlighting how *LDOCE* and auto-frames have many more senses and lexical units than *FrameNet*.

7.2.1 Bicycle

Tables 7.5-7.7 contain a list of the core senses, evoked senses, and secondary evoked senses of *bicycle*.[3] Words that share colour (are not black) are coreferent between the different sentences. Only the words that were identified as connected have been coloured.

Table 7.6 does not contain a sense for *press* or *pedal* even though we would expect them to be evoked from the definition of *bicycle*. When the connection search process does not find any highly weighted senses, usually a sign that there are no attachments, and there are many possible senses (greater than 5), then the expansions are not included, although the complete set of senses are assigned to the word. In future, such ambiguity

[3]To review the distinction between these types of senses in an auto-frame see Section 5.2.4.

may be better resolved with Word Sense Disambiguation (WSD), or by assigning auto-frames within the definitions of auto-frames, instead of the present senses within auto-frames.

Recall that secondary evoked senses are senses that are defined using a core sense. In this case, they include *bicycle* in their definition. During evaluation and review, these senses did not improve the attachment of auto-frames to any contexts. Furthermore, if the term for the definition/sense were used (*e.g.*, *handlebar*), the association with *bicycle* would be obvious. The original goal of secondary evoked senses was to allow the contextualization of general concepts to more specific concepts. For example, *bicycle* (the auto-frame) and *steering* would attach through the sense of *handlebars*.

The *bicycle* auto-frame has over 50 secondary evoked senses. Inclusion of these senses in tasks, such as question-answering, significant affected performance, as it much increases the search space for connection search, which already has a high computational complexity. Thus, they were not included in the reading comprehension evaluation. Furthermore, in early test cases, the secondary evoked senses never actually connected to any concepts in a context; this may suggest that this type of connection and contextualization is rare.

Because auto-frames contain many senses and definitions, and there are connections between them, they can be very hard to visualize. Many links will intersect with other links and the image can be hard to read. The easiest and most accurate representation of an auto-frame is presented in Tables 7.5 and 7.6. In these tables, we can see which senses and definitions are included and how they are attached.

Core Senses

Example	Word	Definition
1	$\text{Bicycle}_{\text{noun}}$	a two-wheeled vehicle that you ride by pushing its pedals with your feet
2	$\text{Bicycle}_{\text{noun}}$	bike
3	$\text{Bicycle}_{\text{verb}}$	to go somewhere by bicycle

Table 7.5: Core lexical senses for the *bicycle* auto-frame

The attachment points for the auto-frame *bicycle* are found in Table 7.8. A number of interesting attachment points are included in the auto-frame because of the evoked senses of *travel*, for example distance, speed, origin, destination, and direction.

Core Evoked Senses

Example	Word	Definition
4	$\text{Ride}_{\text{verb}}$	to travel on a bicycle or motorbike
5	$\text{Ride}_{\text{verb}}$	a journey on a horse or bicycle, or in a vehicle
6	$\text{Ride}_{\text{verb}}$	to travel in a bus, car, or other vehicle
7	$\text{Vehicle}_{\text{noun}}$	a thing such as a car, bus etc. that is used for carrying people or things from one place to another
8	$\text{travel}_{\text{verb}}$	to go a particular distance or at a particular speed
9	$\text{travel}_{\text{verb}}$	to move at a particular speed or in a particular direction
10	$\text{travel}_{\text{verb}}$	to go from place to place to sell and take orders for your company's products
11	$\text{travel}_{\text{noun}}$	to go very fast

Table 7.6: Evoked senses for the *bicycle* auto-frame

Recall that attachment points do not function like traditional slots/roles; instead, they are concepts in expansions graphs, which may be connected with neighboring auto-frames. For example, *push, pedal*, and *foot* are less like features or properties to be assigned (as many roles are); instead, these attachment points allow nearby concepts to be attached to a more complete representation. Consider "John pedaled so fast, that his bicycle went like the wind." Since *bicycle* has a *pedal* attachment point, the two clauses can be connected.

There are also a number of unexpected attachment points in the *bicycle* auto-frame, for example *sell, take, order,* and *product* (These can be found in the lower half of the table). These attachment points come from the inclusion of a sense of *travel* that seems related to a traveling salesman. This sense was not meant to be included. This demonstrates an error in the creation of auto-frames (inclusion of an unexpected sense), but this sense had two attachments to *go* and *place*. These attachments were weighted low, but high enough to get the sense included in the auto-frame.

Secondary Evoked Senses

Example	Word	Definition
12	backpedal$_{\text{verb}}$	to pedal backwards on a bicycle
13	beat-up$_{\text{adjective}}$	a beat-up car, bicycle etc., is old and in bad condition
14	bike$_{\text{verb}}$	to ride a bicycle
15	brake$_{\text{verb}}$	to make a vehicle or bicycle go more slowly or stop by using its brake
16	carrier$_{\text{noun}}$	a metal frame that is fixed to a bicycle or other vehicle and holds bags etc.
17	coaster$_{\text{noun}}$	a brake on some types of bicycle that works by moving the pedals (pedal) backwards
18	cyclist$_{\text{noun}}$	someone who rides a bicycle
19	handlebars$_{\text{noun}}$	the bar above the front wheel of a bicycle or motorcycle that you turn to control the direction it goes in
...	...	...

Table 7.7: Related senses for the *bicycle* auto-frame

Comparison with the FrameNet frame

FrameNet contains two frames evoked by *bicycle*, one as a verb and another as a noun. These frames are *vehicle* and *operate_vehicle*. Each of these two frames is evoked by a number of terms related to many different vehicles, such as *buggy, bus, cab, canoe, ship, tank, warplane, drive, parachute, paddle, etc.*. Thus *FrameNet* does not contain frames or information specific to bicycles, only their general super class *vehicle*.

First consider the *vehicle* frame. Its core frame element is *vehicle*, that is the particular type of the vehicle (*e.g.*, car, plane, boat, Boeing 747, etc.). Core frame elements are determined by the creators of *FrameNet*, and represent elements critical in the understanding of frame. The non-core frame elements are found in Table 7.9. Most of the frame elements are generic properties of *objects*. In fact, only *means_of_propulsion* and *itinerary* seem specific to *vehicles*.

The *use* frame element for a *bicycle* should be filled by transportation, travel, or riding, which is already captured in the auto-frame. The *descriptor* frame element could easily be satisfied by *LDOCE*'s definition *a two-wheel vehicle*, although this frame element could be intended to accept adjectives, such as *beat-up* or *new*. Even *means_of_propulsion* is easily satisfied by *LDOCE*'s definition: "*... by pushing its pedals with your feet*". Since

Descriptive Name	Associated Words
wheels	two-wheeled
vehicle	vehicle, bike, bicycle
ride	bicycle, ride, travel, go, move
rider	you, <subject>
$\text{pedal}_{\text{verb}}$	push
$\text{pedal}_{\text{noun}}$	pedal
foot	foot
destination	somewhere
speed	speed
distance	distance
direction	direction
sell	sell
take	take
order	orders
product	products

Table 7.8: Attachment points for the *bicycle* Auto-frame

the auto-frame for *bicycle* includes elements that satisfy, *means_of_propulsion*, *use*, and *descriptor*, it is either more descriptive or at least comparable on these frame elements.

Frame Element Name	Description
Descriptor	This frame element identifies a characteristic or description of the Vehicle.
Itinerary	The frame element tells about the time and route of the service of the transportation device
Means_of_propulsion	The Means_of_propulsion designates how the vehicle is moved.
Possessor	This frame element denotes the Possessor of the Vehicle
Use	The task which the Vehicle is put to. Normally, this includes the transportation of some kind.

Table 7.9: Non-Core FrameNet Frame Elements for the noun *bicycle*

The *possessor* frame element has no equivalent in the bicycle auto-frame. This is primarily due to the fact that attachment points are not like arguments. With an auto-frame, one would expect the possessor to define the attachment to the possession. The same situation would exist with an adjective filling the *description* frame element. The adjective would select the head and define its relation to it.

Consider the *FrameNet* frame for *operate_vehicle*. The core frame elements are found

in Table 7.10. The *driver* frame element is comparable to the attachment point *person* that is inserted as the subject of most verb definitions. The *driver* element would likely take the same argument as would be attached to *person* who rides or travels on a bicycle. There is no equivalent to *Area* in the auto-frame because it is a locative adjunct. The *Goal* frame element compares with *destination* attachment point found in the definition of *travel*. The *Path* frame element has no equivalent attachment point; perhaps there should be a *path* attachment point in *travel*, but it does not appear in any of the definitions in *LDOCE*. *Source* compares favorably with some attachment points in *travel or go*, which we would call the *origin*. The *vehicle* frame elements would be a bicycle and it is the same as *vehicle* attachment point.

Frame Element Name	Description
Area	This frame element is used for expressions which describe a general area in which motion takes place when the motion is understood to be irregular and not to consist of a single linear path. Locative setting adjuncts of motion expressions may also be assigned this frame element.
Driver	This is the being, typically human, that controls the Vehicle as it moves
Goal	Any expression which tells where the moving object(s) ends up as a result of the motion expresses the frame element Goal. Some particles imply the existence of a Goal which is understood in the context of utterance.
Path	Any description of a trajectory of motion which is neither a Source nor a Goal expresses the frame element Path. This includes directional expressions and "middle of path" expressions
Source	Any expression which implies a definite starting-point of motion expresses the frame element Source. In prepositional phrases, the prepositional object expresses the starting point of motion. With particles, the starting point of motion is understood from context.
Vehicle	This is the means of conveyance controlled by the Driver. It may move in any way or medium.

Table 7.10: Core FrameNet Frame Elements for the verb *bicycle*

Table 7.11 lists the non-core frame elements for *bicycle* as a verb. There are twenty non-core frame elements. None of them match attachment points well. They tend to be for expressions, such as adjuncts, adjectives, adverbs, relative clauses, etc, which would instead have an attachment point for the head concept they modify. This is a

key distinction between *FrameNet* frames and frame elements as compared with auto-frames and attachment points: frame elements exist for all possible descriptive features of a frame, while this is not true of auto-frames. One auto-frame may define itself as attached to another auto-frame, even if in a dependency tree it is the dependent. The dependent auto-frame effectively defines its own attachment point to the head auto-frame. This was seen in Section 5.1.3 in the discussion of genus terms for adjectives and adverbs. However, the same effect can be seen in how auto-frames for nouns sometimes contain attachment points for the verbs that normally take them.[4]

FrameNet frames seem very general, while auto-frames are specific to the word's meaning (this is demonstrated again in Section 7.2.6). *FrameNet* uses two different general-purpose frames for bicycle: one as *verb* and another as *noun*; or *operate* and *vehicle*. In contrast, the auto-frame combines both frames, allowing the meanings to shift between the two and the noun/object to evoke the activity. *FrameNet* does not even have a connection between the two frames for *vehicle* and *ride*. Auto-frames were designed so that *a bicycle* evokes the idea of *riding*, and conversely the idea of *riding* evokes the idea of a *bicycle or vehicle*. This allows the noun sense of the *bicycle* to modulate a word like *travel* or *go* to mean *ride*.[5]

Frame Element Name	Description
Circumstances	This FE indicates the Circumstances under which the operation of the Vehicle takes place.
Cotheme	This is a second moving object, expressed as a PP headed by with.
Degree	The extent to which the Vehicle crosses a boundary or the extent to which the Vehicle completes a course.
Depictive	The state of the Driver or Vehicle during the Transportation.
Distance	Any expression, which characterizes the extent of motion, expresses the frame element Distance.
Duration	The length of the time interval during which the Driver operates the Vehicle.
Event	An event that the Driver is also participating in while directing the Vehicle.
Explanation	The reason for which the Operate_vehicle event occurs.

[4]See Section 7.2.1 for examples where *bicycle* provides the *ride* attachment point for the verb *go*.
[5]This is demonstrated in Section 7.2.1.

Frame Element Name	Description
Frequency	"How often" the Operate_vehicle event occurs.
Manner	Any expression, which describes a property of motion which is not directly related to the trajectory of motion, expresses the frame element Manner. Descriptions of steadiness, grace, and other things count as Manner expressions.
Means	The action performed by the Driver by which the Driver achieves directed Transportation.
Particular_iteration	Expressions marked with this extra-thematic FE modify a non-iterative use of the target, and indicate that it is conceived as embedded within an iterated series of similar events or states.
Place	The Place describes the location of a more specific motion (which has Source, Path or Goal).
Purpose	The state-of-affairs that the Driver tries to bring about by the way that they direct the Vehicle.
Result	Result of the Driver's action.
Route	The Route is the road or path that the Vehicle regularly travels on.
Speed	The Speed is the rate at which the Vehicle moves.
Time	The Time when the Driver operates the Vehicle.

Table 7.11: Non-Core FrameNet Frame Elements for the verb *bicycle*

Contextualized Bicycle Examples

Consider the contextualization examples regarding *bicycle* from section 4.6.2. For convenience, the sentences are repeated in Table 7.12 as examples 1-3. Examples 4-7 are similar sentences, which should evoke similar or comparable graphs.

Each of these sentences has been processed with connection search and auto-frames and are presented in Figures 7.4 to 7.10. When visualizing a Multiple Interpretation Graph (MIG), if more than three senses are selected for a concept, they are hidden to keep the image compact. In the visualization, circles (nodes) are concepts, black solid arrows are dependency relations, black solid arrows starting with the text "Link:" are genus or sense links, and dashed lines are other links. Colour (except green) is used to indicate potentially coreferent concepts, as are dashed colored lines. Links are indicated

Example	Sentence
1	Boris oiled his bicycle.
2	Boris cleaned his bicycle.
3	Boris rode his bicycle.
4	Boris put oil on his bicycle.
5	Boris traveled someplace on his bicycle.
6	Boris bicycled to the store.
7	Boris rode to the store.

Table 7.12: Example modulations of *bicycle*

with arrows labelled with the text "Link:" followed by the link type. Links can also be a form of coreference, but the visualizations do not always correctly colour these concepts.

We can see in almost all of the sentences, except 4, that connection search has done a good job of attaching and pruning the concepts. Furthermore, each set of similar sentences ({1,4} and {3,5,6,7}) has very similar expansions and attachments. In Example 1, *oil* was assigned the *put oil on a machine* sense, which was attached to *bicycle* as a machine.

In Example 4, *oil* was also assigned *put oil on a machine* sense, however the surface level *put* was attached to the *put* in the sense of oil. *Bicycle* and *machine* were similarly attached. Due to space constraints, an incorrectly selected sense of *put* was cropped out of the figure: the sense referred to putting a politician into office. This political *put* sense seemed to attach well with *put* from *oil*.

In Example 2, many senses of *clean* remained and because it is impractical to visualize many senses, these expansion graphs have been hidden. The problem with selecting a sense of *clean* in Example 2 is that many senses of *clean* have an *object* or *something* attachment point, which can connect with *bicycle*. With the exception of the *object/something* attachment point, there are no other attachments to help weigh the senses. While this may seem problematic, it is acceptable and supported by the design of MIG, which allows concepts to have multiple senses and interpretations when they are ambiguous.

In Example 3, we see many connections between *bicycle, travel, ride, vehicle*, and *bike*.

These senses were so strongly attached that the correct senses were selected. Examples 5 and 6 have similar MIGs, with similar senses and attachments. Example 5 demonstrates how attachment points are different from roles, in that *bicycle* as a noun attaches to and contextualizes *travel* though the *ride* attachment point.

Example 7 is interesting because on the surface it seems so similar to examples 3, 5, and 6, yet there is not enough information to select senses or attachments. As before when there are many senses for a concept, they have been hidden to keep the visualization compact. A deeper investigation reveals that *ride* was expanded to many senses and most remain due to lack of attachment with other concepts. *Ride* could refer to riding a bus (as a passenger), a horse, a motorcycle, a bicycle, an elevator, or to move/float on waves. This demonstrates the value of supporting multiple interpretations, when there is not a single clear sense.

For performance reasons, senses are weighted independently of the whole interpretation; this sometimes leads to two attachment points being attached to the same concept (under different interpretations, but there is mutually-exclusive (XOR) interpretation restriction).

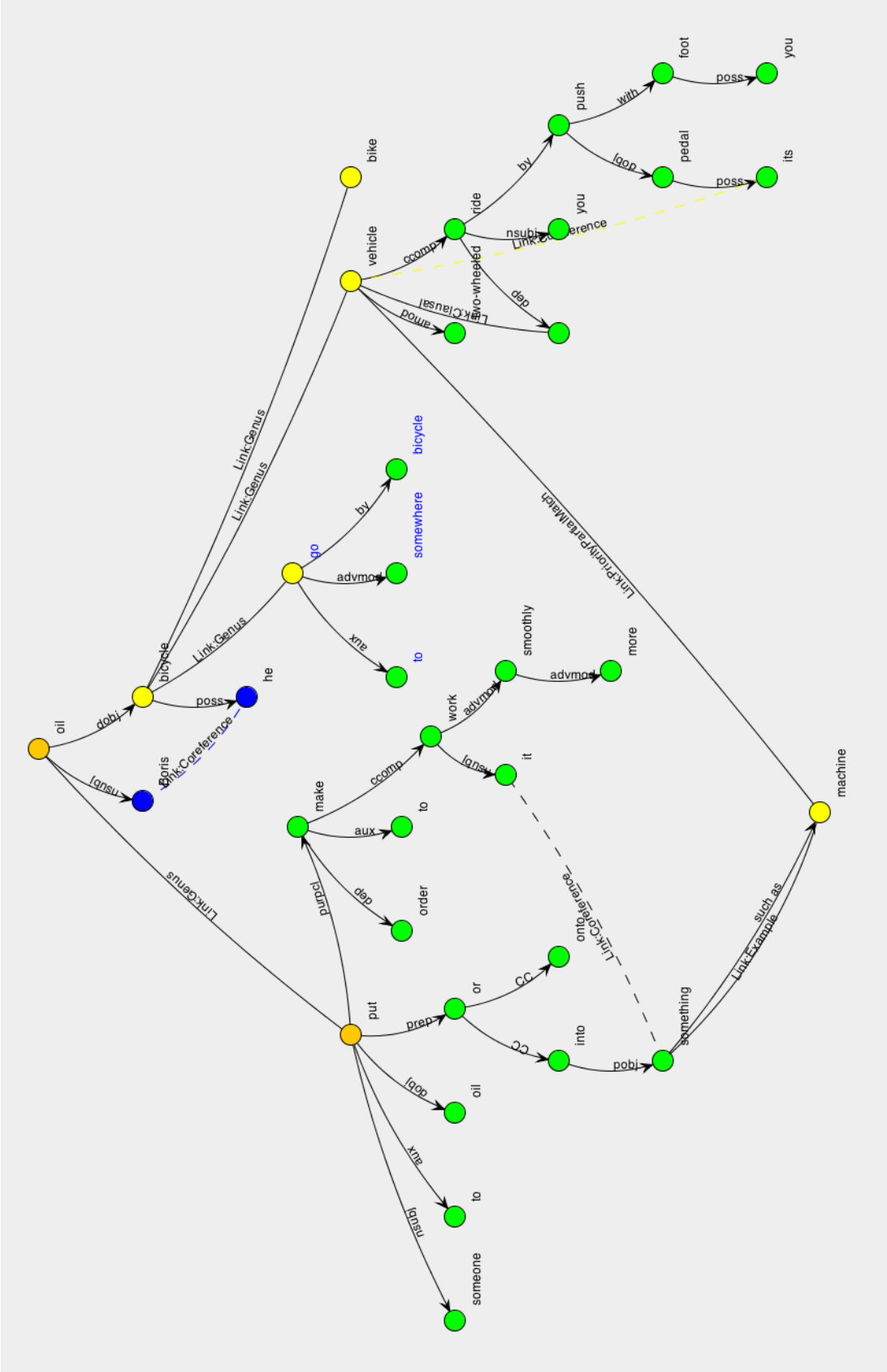


Figure 7.4: Contextualization Example 1 of “Boris oiled his bicycle”

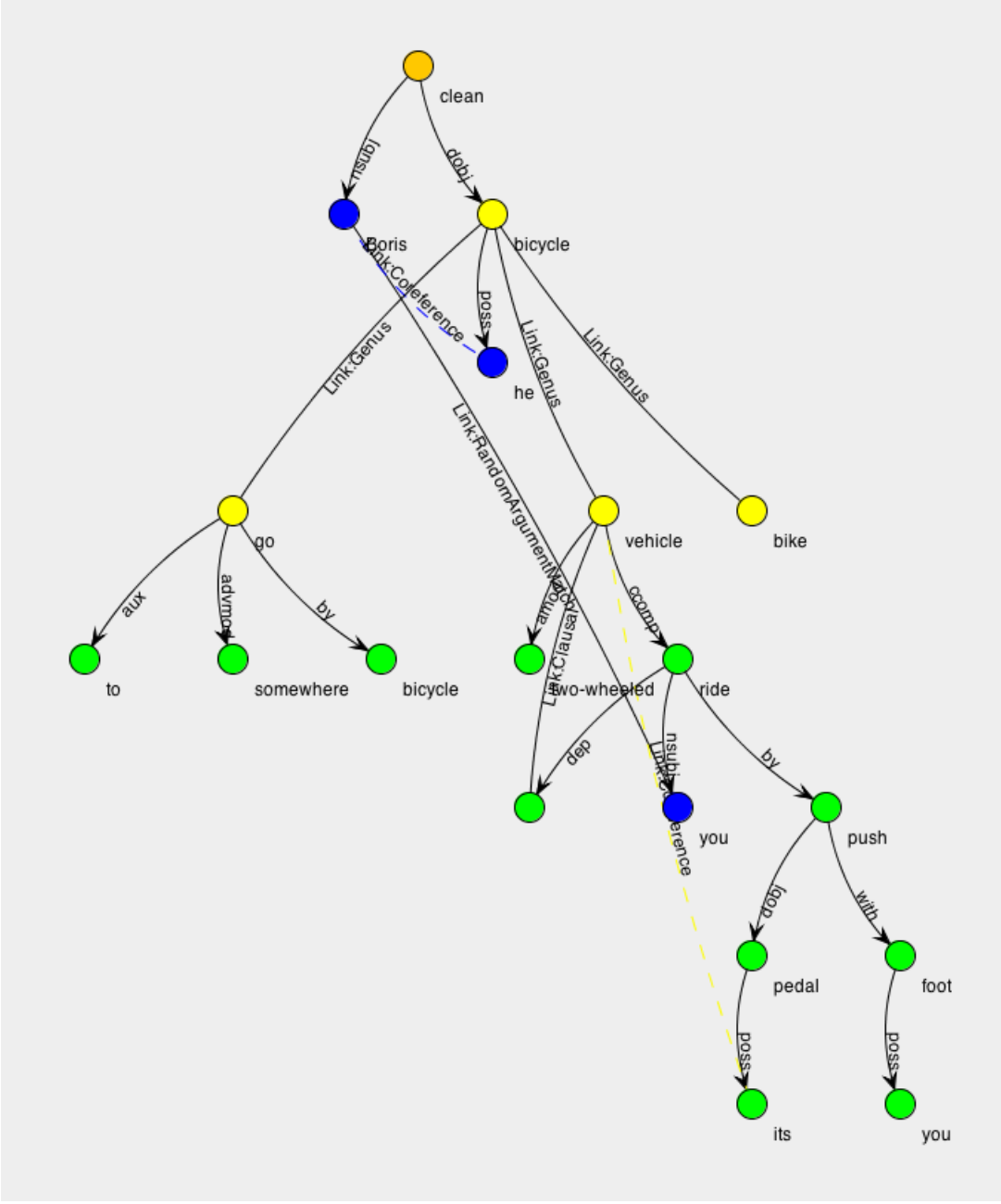


Figure 7.5: Contextualization Example 2 of "Boris cleaned his bicycle"

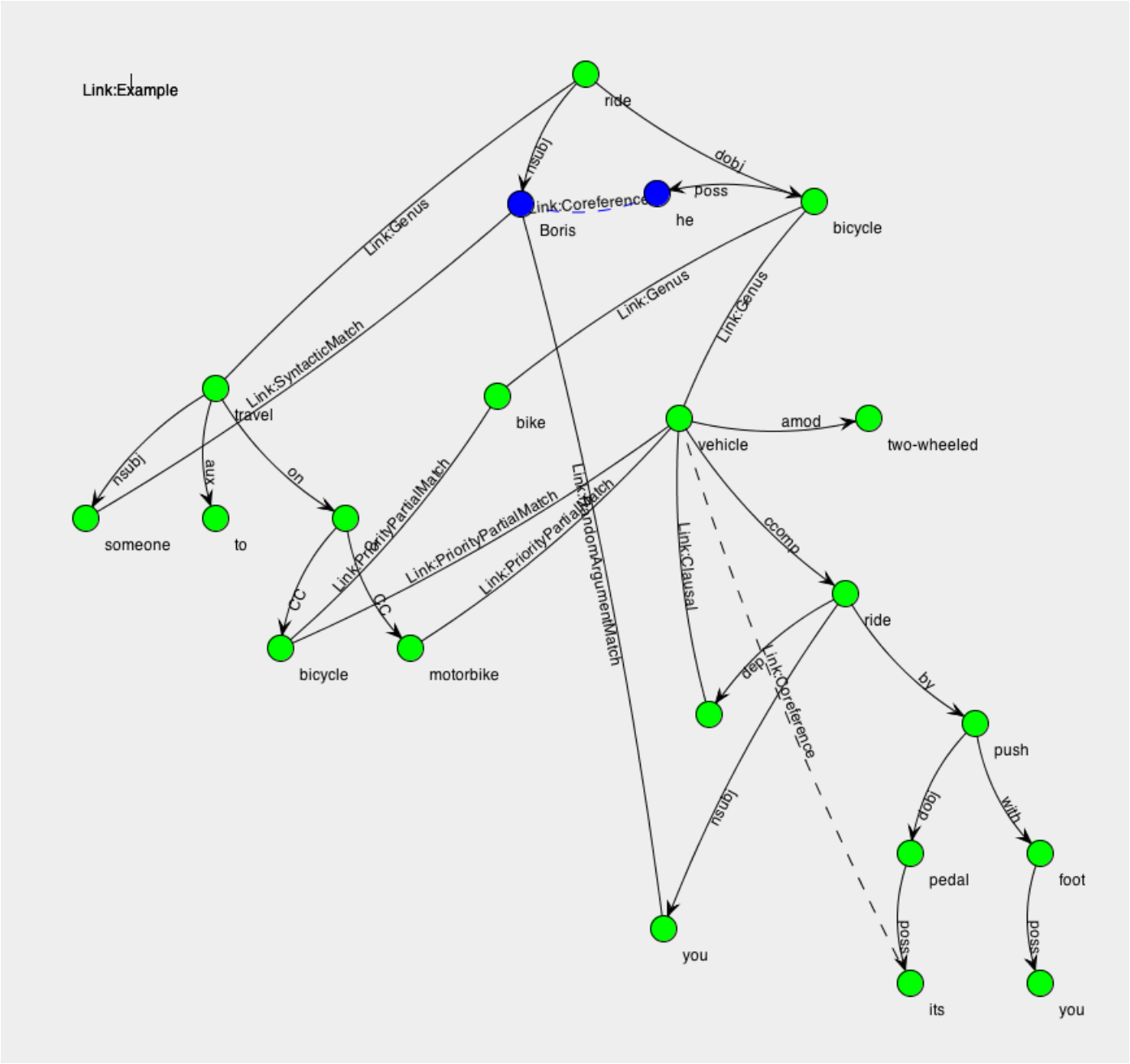


Figure 7.6: Contextualization Example 3 of "Boris rode his bicycle"

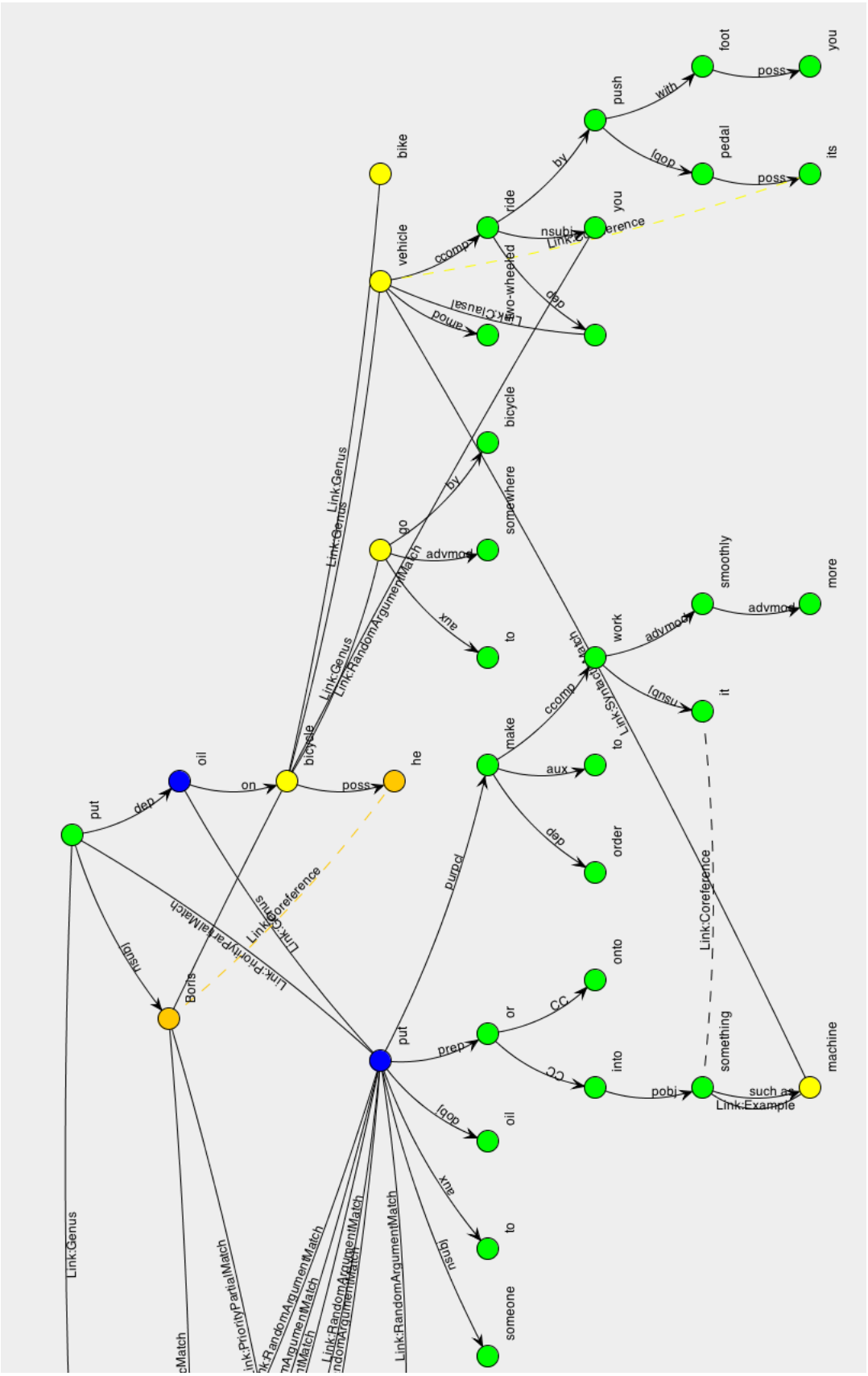


Figure 7.7: Contextualization Example 4 of “Boris put oil on his bicycle”

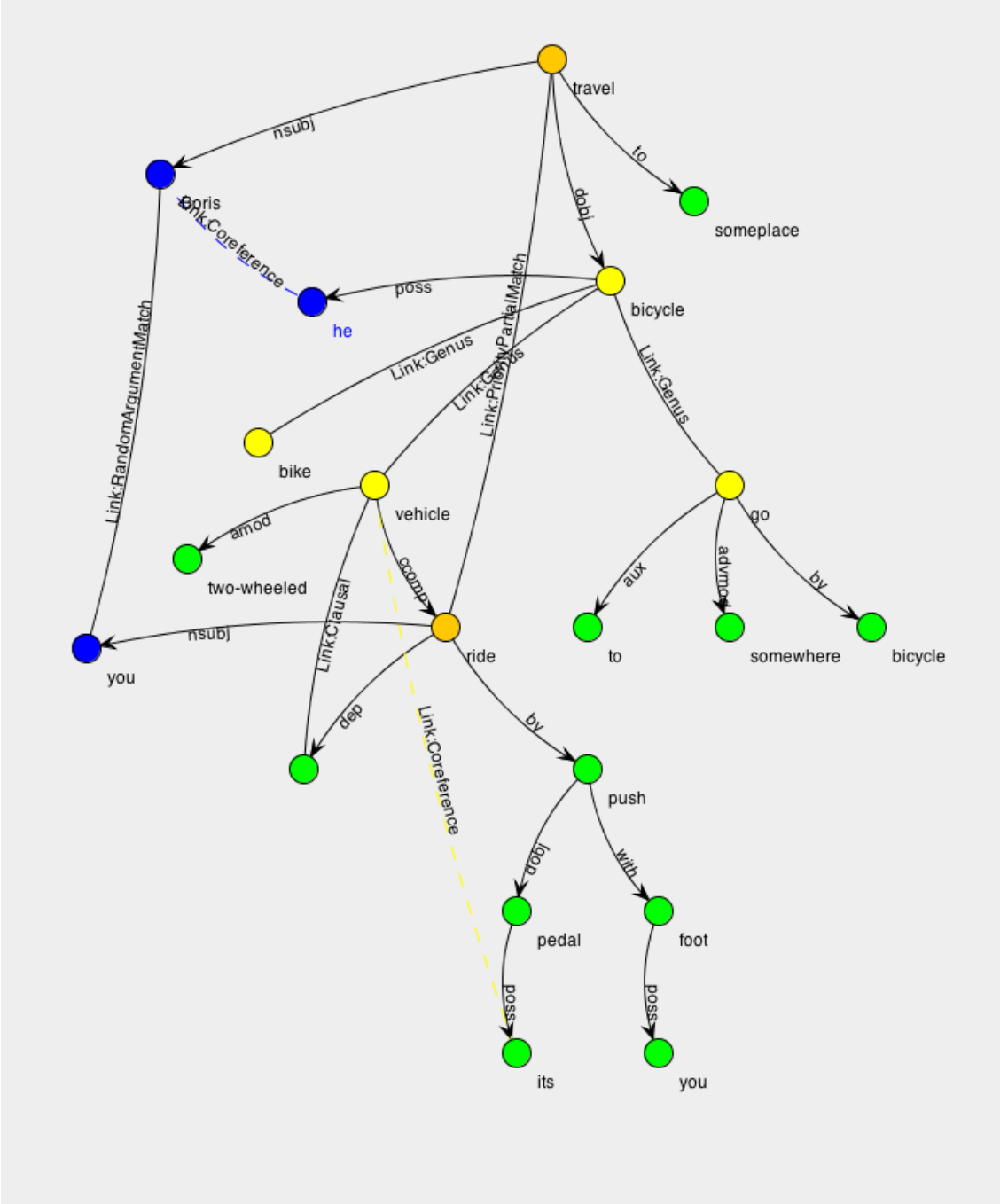


Figure 7.8: Contextualization Example 5 of "Boris traveled someplace on his bicycle"

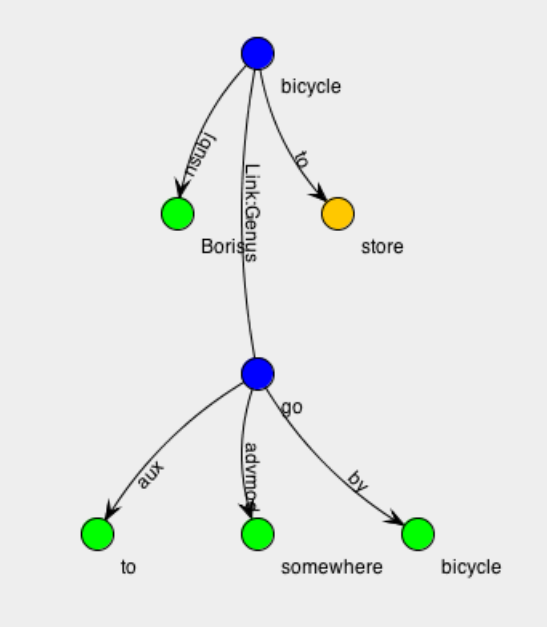


Figure 7.9: Contextualization Example 6 of "Boris bicycled to the store"

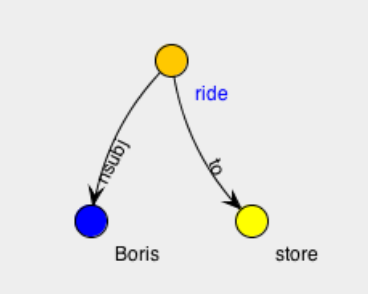


Figure 7.10: Contextualization Example 7 of "Boris rode to the store"

7.2.2 Piano

Let us examine the auto-frame for *piano*. We will limit our review to the core senses, evoked senses, and attachment points. Table 7.13 contains the only core sense for *piano*. Table 7.14 lists the evoked senses for *piano*. Some of the evoked senses seem accurate and related to *piano*, but the sense of *sit* is incorrect. Furthermore, many expansions were not included in the auto-frame because connection search could not prune the senses sufficiently. Some terms, which were not expanded or included in the auto-frame, are:*front, press,* and *keys*.

Core Senses

Example	Word	Definition
1	$\text{Piano}_{\text{noun}}$	a large musical instrument that you play by sitting in front of it and pressing the key

Table 7.13: Core lexical sense for the *piano* auto-frame

Core Evoked Senses

Example	Word	Definition
2	$\text{Musical}_{\text{adjective}}$	connected with music or consisting of music
3	$\text{Instrument}_{\text{noun}}$	an object such as a piano, horn, violin etc., used for producing musical sounds
4	$\text{Instrument}_{\text{verb}}$	musical instrument
5	$\text{Play}_{\text{verb}}$	to perform a piece of music on a musical instrument
6	sit_{verb}	to do a job, go to a meeting etc. instead of the person who usually does it

Table 7.14: Core evoked senses for the *piano* auto-frame

Figure 7.11 depicts the contextualization of "Boris played piano". We can see that the auto-frame for *piano* has attached to the word *play*. However, the wrong sense of *play* is selected because *Boris* attaches to many attachment points in the auto-frame selected for *play*. This demonstrates a problem with the weigh and prune algorithm.

Descriptive Name	Associated Words
musical	musical
instrument	instrument, piano
pianist	you
play	play, perform
press	press
key	key
produce	producing

Table 7.15: Attachment points for the *piano* Auto-frame

This problem is caused by weighing each concept independently.[6] The same concept can be attached to many attachment points under different interpretations, however each attachment may contribute to increasing the weight of the sense.

Figure 7.12 depicts the contextualization of "Piano music is the best". In this example, all the senses are correctly selected and the correct attachment is made between *music* and *play*.

In *FrameNet*, *piano* only evokes the *noise_makers* frame. This frame is a general-purpose frame with many lexical units like *piano, guitar, bell, alarm* and *siren*. The only core frame element is *Noise_maker*, the device actually making the noise, that would normally be the lexical unit. This frame is only related to *artifact*, and the non-core frame elements are very general properties of created objects/artifacts. The *FrameNet* frame contains no context specific information about a piano, or even any relation to music or instruments.

[6]Recall that concepts are weighed independently to reduce the search space and the runtime of the system.

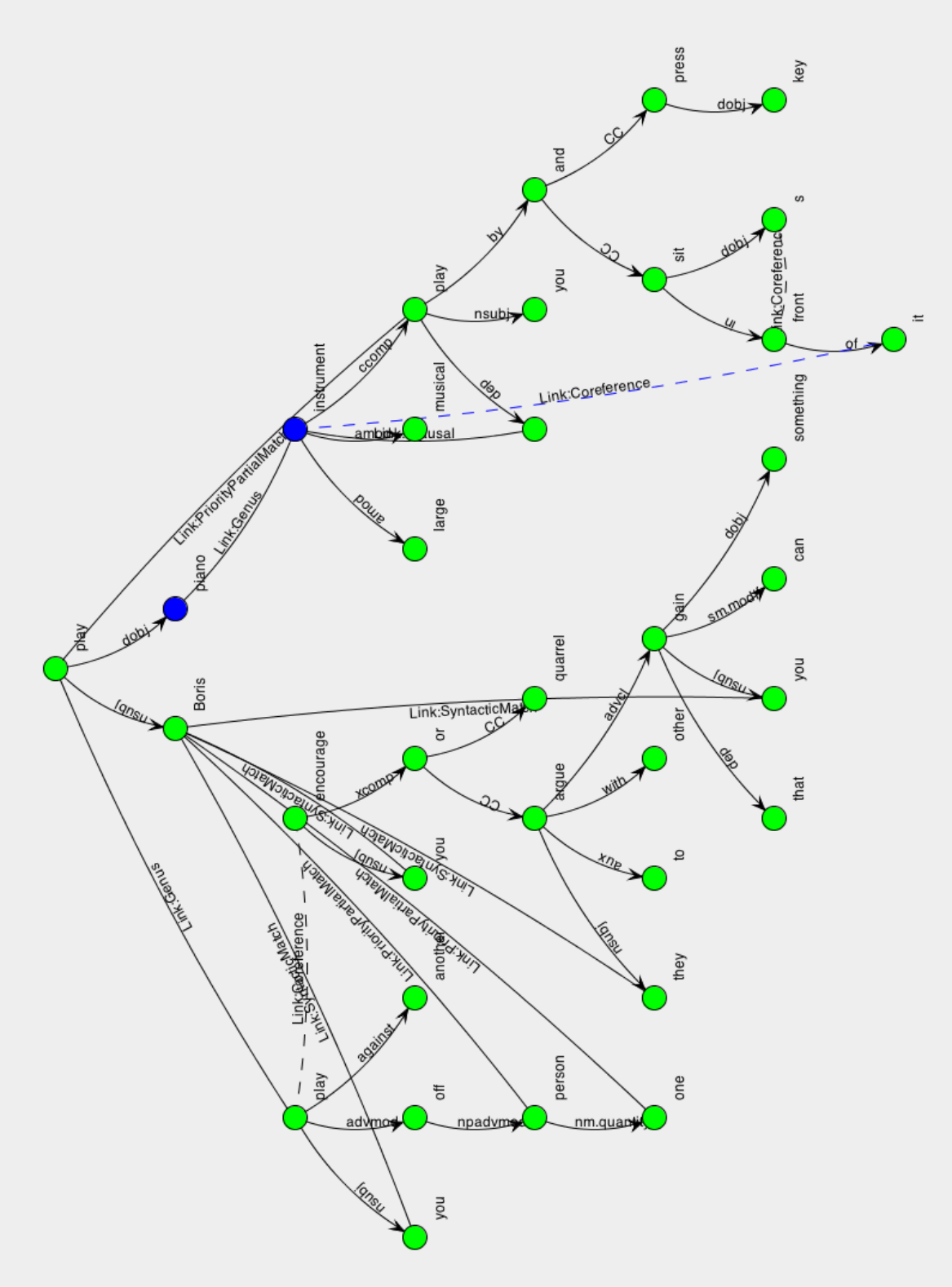


Figure 7.11: Contextualization Example of “Boris played piano”

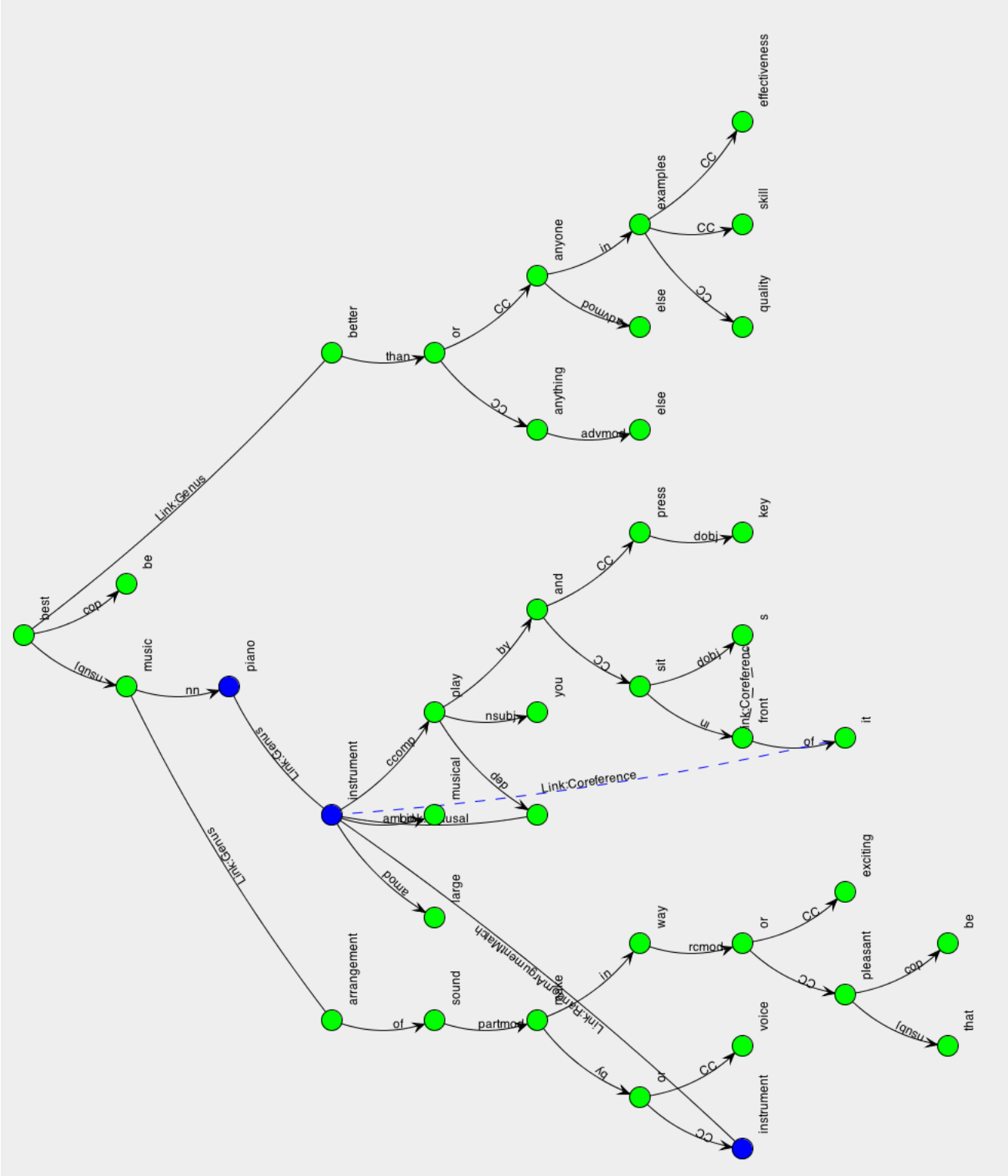


Figure 7.12: Contextualization Example of "Piano music is the best."

7.2.3 Airplane

The core and evoked senses for the auto-frame of the word *airplane* are presented in Tables 7.16 and 7.17. As with the other auto-frames, we find that some senses are correctly attached and assigned while others are not. The sense in example 8 is not related to an airplane and is a poor choice of sense for engine. With this auto-frame, the missing senses are more concerning that those present in it. There are several senses of *fly* related to planes, carrying passengers, and being in the air that are not present. One would expect the senses for *aeroplane* to be a core sense. The airplane sense of *plane* is not even included. From this auto-frame we can see that related senses are being added to the auto-frame, but the construction of auto-frames is still missing important senses.

Core Senses

Example	Word	Definition
1	airplane$_{noun}$	a flying vehicle that has one or more engines
2	airplane$_{noun}$	aeroplane

Table 7.16: Core lexical senses for the *airplane* auto-frame

Table 7.18 lists the attachment points for the auto-frame *airplane*. These attachment points satisfy some of the core frame elements previously shown for the *Vehicle* frame in *FrameNet*.

In *FrameNet*, *airplane* would evoked same *vehicle* frame as *bicycle*. This frame contains no *airplane*-specific information.

Core Evoked Senses

Example	Word	Definition
3	aeroplane$_{\text{noun}}$	a flying vehicle with wings and at least one engine
4	fly$_{\text{verb}}$	to control a plane through the air
5	fly$_{\text{verb}}$	to cross an area of water in a plane
6	vehicle$_{\text{noun}}$	a thing such as a car, bus etc. that is used for carrying people or things from one place to another
7	engine$_{\text{noun}}$	piece of machinery with moving parts that changes power from steam, electricity, oil etc., into movement
8	engine$_{\text{noun}}$	a vehicle that pulls a railway train

Table 7.17: Core evoked senses for the *airplane* auto-frame

Descriptive Name	Associated Words
vehicle	airplane, vehicle, thing, aeroplane
engine	piece, vehicle
fly	fly, control, cross
wing	wing
air	air
water	water
area	area
carry	carry
cargo	people or things
source	place

Table 7.18: Attachment points for the *airplane* Auto-frame

7.2.4 Bribe

The core and evoked senses for the *bribe* auto-frame are presented in Tables 7.19 and 7.20. The attachment points for the *bribe* auto-frame are shown in Table 7.21. This auto-frame has many of the senses that we would expect, such the senses for *pay*, *dishonest* and *persuade*. However, there are two senses of *pay* that seem better suited: "to give someone money for the job they do" or "to give someone money for something you have bought, or for something they have done for you". It is unclear why these senses did not get included in the auto-frame. However if auto-frames were labeled with other auto-frames as sense (instead of *LDOCE* senses), then all these senses of *pay* would have been included, since they are a part of a single auto-frame.

Core Senses

Example	Word	Definition
1	bribe$_{\text{verb}}$	to pay money to someone to persuade them to help you, especially by doing something dishonest

Table 7.19: Core lexical sense for the *bribe* auto-frame

Core Evoked Senses

Example	Word	Definition
2	pay$_{\text{verb}}$	to give someone the money that you owe them
3	pay$_{\text{verb}}$	repay
4	pay$_{\text{verb}}$	to give someone all the money you owe them
5	persuade$_{\text{verb}}$	to make someone decide to do something, especially by repeatedly asking them or telling them reasons why they should do it
6	dishonest$_{\text{adverb}}$	not honest

Table 7.20: Core evoked senses for the *bribe* auto-frame

In *FrameNet*, there is no *bribe* frame, but there is a *pay* frame. The core frame elements of the *pay* frame are *buyer, seller, money, goods,* and *rate*. These concepts all have matching concepts in the *bribe* auto-frame, except *rate*.

Descriptive Name	Associated Words
bribe	bribe, pay, give, repay
persuade	persuade, make
money	money
recipient	someone
help	help
goal	something
dishonest	dishonest

Table 7.21: Attachment points for the *bribe* Auto-frame

7.2.5 Bank

FrameNet only contains three senses for *bank*. The frame Relational_natural_features is also evoked by *summit*, *shore*, *footholl*, and more. The frame Businesses is also evoked by *chain*, *mill*, *shop*, and more. The frame Reliance_on_expectation covers the sense *to bank on*.

Section 4.5.1 stated that the generality of *FrameNet* can be both a strength and a weakness. Each of the frames listed above has generic frame elements. Since each frame is very general, they offer little to differentiate between the lexical units that evoke them. Because the frames and frame elements are generic, they are not comparable to attachment points for the auto-frames, which are more detailed and specific.

"Bank" is a well known example of a polysemous word. Let us examine how the *LDOCE* senses were divided into auto-frames. The *LDOCE* senses for *bank* are found in Table 4.5, repeated here as Table 7.22. Senses 1, 2, and 3 are derived; thus, they must be related to at least one of the other senses. However the system, in particular connection search, was only able to prune down the derived senses association to senses 5-7.

Senses 5-12 each had their own auto-frame. Other than the derived senses, most of the senses are independent. The auto-frames for senses 5-7 included the derived senses. Since senses 5-12 are independent, it is expected that each would have its own auto-frame.

Reviewing the auto-frames for bank reveals very little about the performance of the algorithms.

7.2.6 Comparison of Lexical Units

This section compares five lexical units each from *FrameNet* and auto-frames. The high-level comparison is summarized in Table 7.23. The upper half of the table shows

Number	Definition
1	a local office of a bank
2	to put or keep money in a bank
3	to keep your money in a particular bank
4	to arrange something into a pile or into rows
5	a business that keeps and lends money and provides other financial services
6	the money in a gambling game that people can win
7	a slope made at a bend in a road or racetrack to make it safer for cars to go around
8	to make a plane, motorcycle , or car slope to one side when turning
9	to cover a fire with wood, coal etc. to keep it going for a long time
10	to depend on something happening or someone doing something
11	a large pile of earth, sand, snow etc.
12	land along the side of a river or lake

Table 7.22: Senses for Bank in *LDOCE*

lexical units, nouns and verbs, randomly selected from *FrameNet*, the lower half – from auto-frames; they appear in the *Term* column.

The *Sense* column lists the intended senses of the lexical units. The *AF* column says whether a similar auto-frames existed. The *FN* column makrs the existence of a similar frame in *FrameNet*. *FN Name* is the name of the *FrameNet* frame. The *Gen.* column shows if this is a general frame for many senses or specific frame for this term. The *LUs* column shows the number of lexical units that evoke this frame in *FrameNet*. The second number in the *AP/FE* column indicates the number core frame elements in the *FrameNet* frame, the first number – how many of those could be satisfied by an attachment point in the auto-frame.

The frame *Perception_experience* did not have an exact equivalent in the auto-frames. The *perceive* as in *understand* auto-frame had some overlapping elements, but did not capture the exact same meaning. It can be noted that only 2 of the 5 core frame elements could be satisfied by this inexact comparison of frames.

There was no comparable auto-frame for the sense of *touch* defined in *FrameNet*.

For the auto-frames that matched the senses of the *FrameNet* frames (*slouch, per-*

Term	Sense	AF	FN	FN Name	Gen.	LUs	AP/FE
$\text{slouch}_{\text{verb}}$	lazy posture	Y	Y	Posture	Y	25	2 / 2
$\text{perceive}_{\text{verb}}$	understand	Y	Y	Perception_experience	Y	13	2 / 5
$\text{perceive}_{\text{verb}}$	notice	Y	Y	Becoming_aware	Y	28	2 / 3
$\text{laugh}_{\text{verb}}$	make noise	Y	Y	Make_noise	Y	105	3 / 3
$\text{hinge}_{\text{verb}}$	depends on	Y	Y	Contingency	Y	14	2 / 2
$\text{touch}_{\text{noun}}$	a small amount	N	Y	Quantified_mass	Y	58	- / 3
$\text{outflank}_{\text{verb}}$	attack from behind	Y	N	-	-	-	-
$\text{forgather}_{\text{verb}}$	meet a group	Y	N	-	-	-	-
$\text{psycho}_{\text{noun}}$	crazy	Y	N	-	-	-	-
$\text{rage}_{\text{verb}}$	feel angry	Y	N	-	-	-	-
$\text{illogical}_{\text{noun}}$	not sensible	Y	N	-	-	-	-

Table 7.23: Comparison of Lexical Units

ceive, laugh, and *hinge*), the core frame elements had comparable attachment points. This suggests that auto-frames also capture the core relationships that *FrameNet* frames capture.

None of the randomly selected lexical units from *LDOCE* had equivalents in *FrameNet*. The auto-frame evoked by $rage_{noun}$ was related to feeling angry; the only *FrameNet* frame for $rage_{noun}$ was *Fire_burning*, which was not at all similar. None of the randomly selected lexical units had matches in *FrameNet*; this was not a surprising result, given that the auto-frames have about 47,000 lexical units and *FrameNet* has "more than 10,000 word senses".[7]

In Table 7.23, all of the randomly-selected lexical units from *FrameNet* belong to very general frames, not specific to the term. *Laugh* evoked the very general frame *Make_noise*, which has over one hundred lexical units. As previously noted, this can be seen both as a strength and weakness of *FrameNet*: many lexical units will evoke the same frame, making comparison easy by producing a uniform representation. However, there seems to be a loss of information in representing *laugh* as *making noise*. Auto-frames have a specific sense for laugh, which is related to jokes and amusement, but also includes senses related to making noise with one's voice. Even though auto-frames do not create the same uniform representation for many different expressions of making noise, each auto-frame will usually contain some overlapping senses, and can thus be compared using the simple entailment system described in Section 5.3.1.

[7]https://framenet.icsi.berkeley.edu/fndrupal/about – accessed April 2016.

Lexical Unit Comparison Conclusions

This comparison verifies that many of the lexical units in *FrameNet* are very general and not specific to the word or sense that evokes the frame. It would appear that the core frame elements for most *FrameNet* frames are captured in auto-frames.

7.3 Extrinsic Evaluation: Reading Comprehension

The reading comprehension evaluation uses the Remedia data set, which was first described in section 3.11.3.

7.3.1 Data

Most of the texts are about four paragraphs long, with approximately 15 sentences per text. The following example, taken from the Remedia data set (rm2-1.txt), is about Christopher Robin. The text and questions for this document are shown in the following box. The evaluation used the Remedia corpus with reading levels for grades 2 and 3. Questions for grade levels 4 and 5 were not included in the evaluation because they are significantly different. To answer them would require discourse analysis and features not in this ad-hoc solution.

1989 Remedia Publications, Comprehension/5Ws2
Storybook Person Found Alive!

(ENGLAND, June, 1989) - Christopher Robin is alive and well. He lives in England. He is the same person that you read about in the book, Winnie the Pooh.

As a boy, Chris lived in a pretty home called Cotchfield Farm. When Chris was three years old, his father wrote a poem about him. The poem was printed in a magazine for others to read.

Mr. Robin [sic] then wrote a book. He made up a fairy tale land where Chris lived. His friends were animals. There was a bear called Winnie the Pooh. There was also an owl and a young pig, called a piglet. All the animals were stuffed toys that Chris owned. Mr. Robin made them come to life with his words. The places in the story were all near Cotchfield Farm.

Winnie the Pooh was written in 1925. Children still love to read about Christopher Robin and his animal friends. Most people don't know he is a real person who is grown now. He has written two books of his own. They tell what it is like to be famous.

1. Who is Christopher Robin?
2. What did Mr. Robin do when Chris was three years old?
3. When was Winnie the Pooh written?
4. Where did young Chris live?
5. Why did Chris write two books of his own?

Example from Remedia: rm2-1.txt

Note: The author of *Winnie the Pooh* is A. A. Milne, not "Mr. Robin". I used and transcribed the Remedia data without any editing of the content.

To answer these sample questions, some abstraction is needed, though finding the sub-graph that matches the query usually gets the reader very close to the answer. Table 7.24 contains some examples of questions and the location of the answer in the text. The sub-graph common between the query and the answer text is highlighted in yellow.

To answer question 1 does not just require finding something matching *Christopher Robin*, but a sentence that describes who he is. Question 2 requires the understanding that Mr. Robin is Chris's father and some normalization of the re-ordered expression.

Question	Answer Text	
1	Who is Christopher Robin?	He is the same person that you read about in the book, Winnie the Pooh.
2	What did Mr. Robin do when Chris was three years old?	When Chris was three years old, his father wrote a poem about him.
3	When was Winnie the Pooh written?	Winnie the Pooh was written in 1925.
4	Where did young Chris live?	As a boy, Chris lived in a pretty home called Cotchfield Farm.
5	Why did Chris write two books of his own?	He has written two books of his own. They tell what it is like to be famous.

Table 7.24: Baseline results on factoid questions from TREC 2004 Question-Answering (QA) task

Question 3 is straightforward matching of predicates and selecting a date for when the event/predicate occurred. Question 4 calls for the understanding that *young Chris* and *Chris, as a boy* are equivalent. Question 5 requires some discourse analysis to understand how the second sentence is connected to the first.

7.3.2 Results

The primary evaluation metrics used are human sentence accuracy and human sentence precision. These are consistent with most other evaluations done on this corpus (Hirschman et al., 1999; Charniak et al., 2000; Riloff, 2000; Wellner et al., 2006). "Human sentence" refers to a sentence in which most humans claim to have found the answer. Some questions have no human sentences, although all the questions do have an expected answer. In rare cases there is more than one human sentence, indicating the answer can be found in either sentence.

Human sentence accuracy is the percentage of questions where the selected sentence matches the human sentence. Human sentence precision is the number of selected sentences containing the answer that agree with human selection divided by the number of questions answered.

Human sentence precision seems an appropriate evaluation of this work, as it measures

whether a provided answer is correct; the system is not expected to answer all questions, but the answers it provides are expected to be correct. High-precision symbolic systems can either be combined with other high-precision system to improve coverage or with lower-precision (high-coverage) systems to maximize f-measure.

Figure 7.13 plots the overall results on the Remedia data with some baselines for comparison. This figure shows the number of responses for each metric (not the percentages). *Answered* is simply the number of questions answered. *Correct* is the number of questions, which returned the exactly right textual responses. *HS@1*, *HS@2* and *HS@5* are the number of correct human sentences returned within the first response from the system (HS@1), within the first two responses (HS@2), and the first five responses (HS@5). Since no effort was made to prioritize or sort sentences which match the question/query, it is seems reasonable to consider the responses at 1, 2, and 5. A refined system would attempt to select the optimal answer, but no such technique is applied here. In fact, very few questions had 5 or more matching sentences, and these were usually vague questions like who is X, where X appears often in the text.

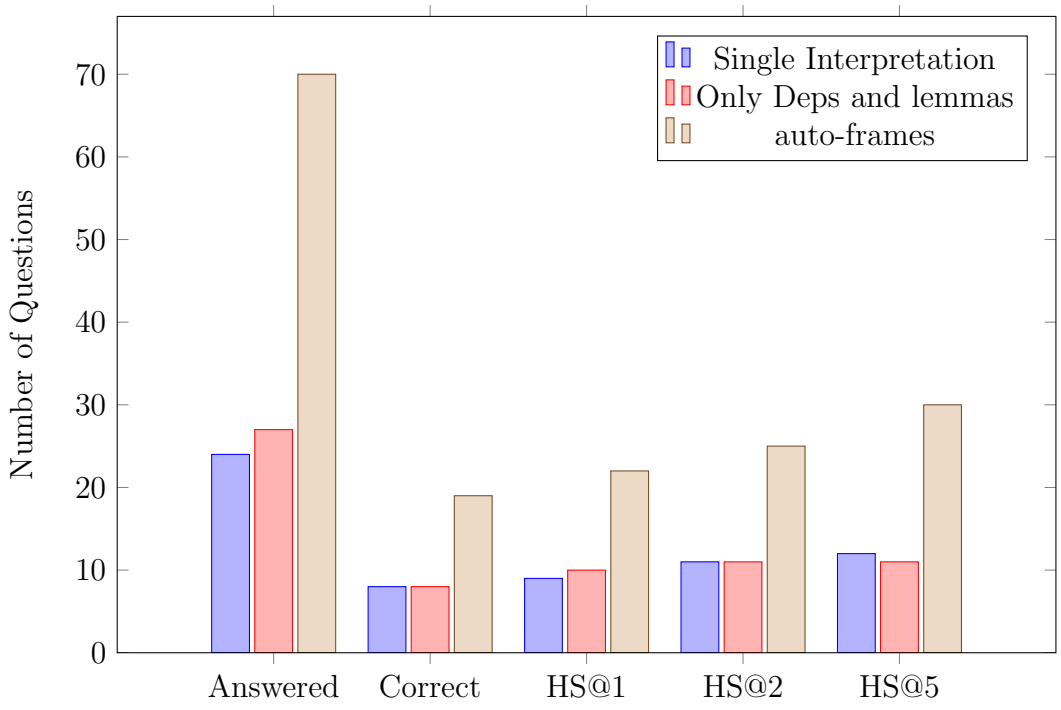


Figure 7.13: Recall on Remedia reading comprehension and comparison of the different knowledge representations

Figure 7.13 shows the number of Remedia questions correctly answered using auto-

frames. For comparison, the success rate of the same algorithms but only using dependency relations and lemmatized-term matching is included. Without the use of auto-frames and attachment points, about half as many questions are answered. This suggests that matching is improved because of the abstraction and generalization that auto-frames and attachment points provide. While the overall methodology is far from refined, the application of auto-frames improves answer selection over the random baseline[8] of 5% and over the baseline of just using lemmas and dependencies.

Recall that by design MIGs and auto-frames allow multiple interpretations in the representation to support the idea of ambiguity. Figure 7.13 also includes the results of only using a single (highest ranked) interpretation, instead of leveraging the support for multiple interpretations. The addition of multiple interpretations allows the system to find answers to more questions.

Figure 7.14 shows the precision and recall when using a single interpretation, multiple pruned interpretations, and all interpretations. Single interpretation uses auto-frames, but only allows a concept to be assigned a single sense, and prunes all but one sense from the auto-frame. All interpretations are kept by disabling the pruning step in connection search. The inclusion of multiple interpretations improves performance particularly on recall without significantly impacting precision. Pruning unlikely senses, using connection search, shows improved precision with some impact in recall. The auto-frames in Figure 7.13 contained all interpretations.

From Figure 7.14, it can be observed that pruning low-weighted branches improves precision. These results demonstrate the benefits of having multiple interpretations.

The observed recall is low. Using all interpretations, only 70 questions were answered of the 305 questions. A major cause of the low recall may be related to the limited set of questions supported by the ad-hoc system. The system was only designed to support a small set of question types: *who, when, where* and a few variants of *what*, while the reading comprehension text includes a number of other types of questions. Table 7.25 shows the number of questions of each type and the system's performance on each type of question. The evaluation metrics are based on multiple pruned interpretations at 5 responses.

The previously presented results were all based on an automatic evaluation of the precision, recall, and accuracy of the system on the Remedia data. I reviewed the results and found a number of insightful metrics. It is noteworthy that on occasion the evaluation module failed to recognize that the sentence supplied by the system correctly matched

[8]It is based on the chance of randomly selecting a sentence which contains the answer to the question.

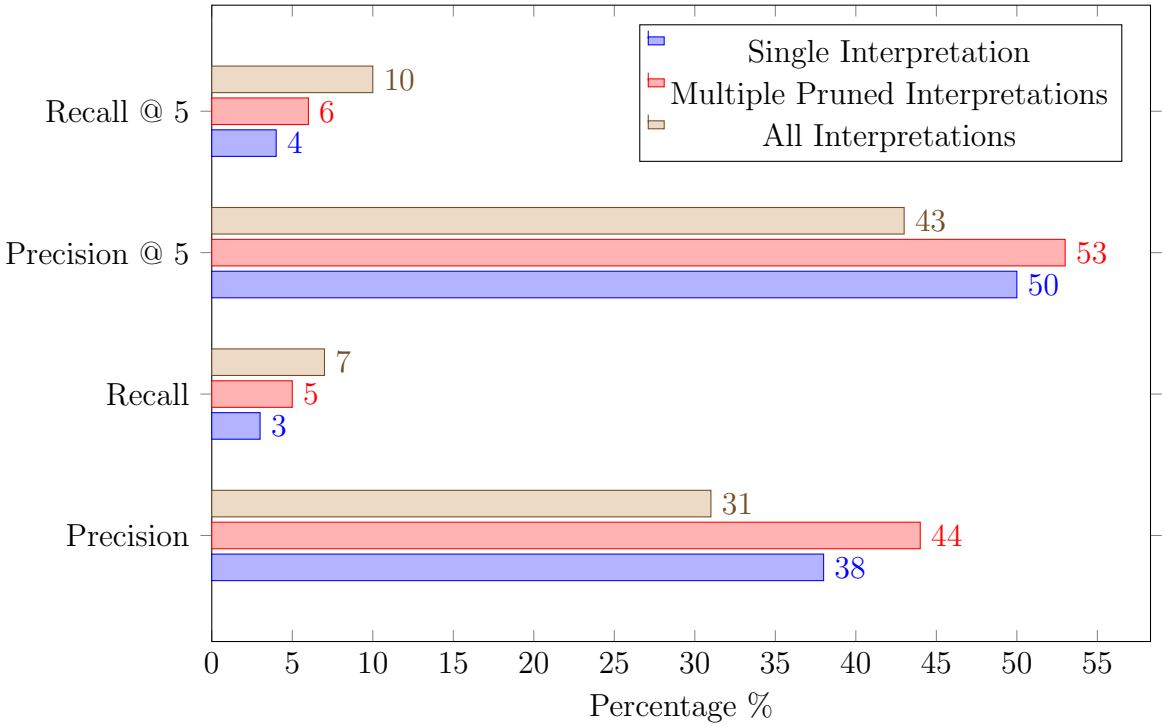


Figure 7.14: Impact of Multiple Interpretations on Remedia

	Total	Answered	Correct	Human Sentence Accuracy	Human Sentence Precision
Who NAME (find description)	51	29% (15)	12% (6)	14% (7)	0.47 (7)
Who description (find name)	6	50% (3)	0% (0)	33% (2)	0.67 (2)
What Did _ Do	3	67% (2)	33% (1)	33% (1)	0.50 (1)
Where	59	8% (5)	7% (4)	5% (3)	0.60 (3)
When	58	9% (5)	5% (3)	7% (4)	0.80 (4)
What Is	47	9% (4)	2% (1)	2% (1)	0.25 (1)
What is the name of	3	0% (0)	0% (0)	0% (0)	NaN (0)
Other	78	0% (0)	0% (0)	0% (0)	NaN (0)

Table 7.25: Break down of question types and system performance on questions from grades 2 and 3 in the remedia data

the human sentence. This was primarily caused by differences in sentence splitting. Secondly, a number of the sentences contained the correct context for the answer, but due to referential ambiguity, humans selected the answers in other sentences. In these responses, the correct context was found, but the correct answer was not given because the system did not select the correct response.

Table 7.26 contains examples of sentences which match the query context but do not contain the answer to the question.

Question	Where did he win four medals?
System	Today Jesse won his fourth gold medal in the Olympics.
Question	Who reached the top of Mount Everest?
System	After hours of fighting the snow and ice they reached the top called the summit.
System	When they got to the top of mount Everest the two men had climbed 29 028 feet.

Table 7.26: Examples of sentences which match the queries, but do not contain the answer

A manual evaluation of all interpretations increased the number of human sentences matched at one answer (HS@1) to 28, HS@2 to 31, and HS@5 to 36. The precision thus increases to 0.41, 0.45, and 0.53 respectively The correct context was found for 44 questions (at 5 answers), producing a precision for the correct context of 0.65. Recall for HS@1, HS@2, HS@5, and correct context @ 5 was 0.09, 0.10, 0.12, 0.14 respectively.

Hirschman et al. (1999) achieved human sentence accuracies between 28% and 36% percent using a variety of bag-of-words approaches. The results are much better than 5% score expected from random selection. More recently, Charniak et al. (2000) and Wellner et al. (2006) obtained human sentence accuracies as high as 41%. Grois and Wilkins (2005) used reinforcement learning and examples in a learning system to obtain result of 48% human sentence accuracy. Riloff (2000) used a rule-based system to obtain 40% human sentence accuracy. By comparison, this system obtained under all interpretations at 5 answers 10% human sentence accuracy and on the supported questions 20%.

However, these results are not unreasonable given that this research was exploratory and focused on a new knowledge extraction and representation system. Moldovan and Rus (2001) showed (on a different data set) that LF could be used to answer 52 out of

351 questions, though only 25 questions were correctly answered.

The results of this system are very weak by comparison, although they suggest potential in such a knowledge base and the ideas demonstrated here. Furthermore one would expect that by using a more mature reading comprehension system, refinement or optimization of many of the configurable values, and task specific improvements, these results could be further increased. Recall could be increased by adding specific support for questions like "What is...", "What is the name of the ...", and "Why did ..." This would improve the system's coverage of the questions as much as 42%. Improving answer selection could remove the need to test the top results at 1, 2, and 5 answers. Pruning has shown improvement in precision, yet it is a very unrefined algorithm.

7.3.3 Review of Remedia Answers

If we examine some of the results from the reading comprehension data, we can see that both contextualization and expansion of concepts to their auto-frames assist in answering a number of the questions. This demonstrates the functionality of the representation and methodology. Some of the results from the reading comprehension evaluation are presented in Table 7.27.

More than half the answers provided by the auto-frame based system are correct, although a large portion of questions remain unanswered. The system achieves high precision but low recall, as expected: not all question types were supported and the system is unrefined.

In this comparison, the OpenEphyra[9] question-answering system was used as a comparison. This system is derived from the Ephyra system (Schlaefer et al., 2006) used at Text REtrieval Conference (TREC) in the QA track (Schlaefer et al., 2007). OpenEphyra is a statistical system, which relies on co-occurrence, redundancy, and patterns to correctly identify the right answer.

Consider the following review of a small test set of 10 questions. This system answered 4 correctly, 1 partially correctly, and no answer was provided for the remaining 5 questions. Precision of almost 100% is achieved and a recall of 50%. OpenEphyra answers 8 of the 10 question, 3 are answered correctly, 1 partially correctly, and 4 incorrectly. A precision of almost 50% and a recall of 40% were obtained.

The following tables provided information on each question that the auto-frame based system answered correctly. The expected answers are shown as correct answer, while the

[9] `http://www.ephyra.info/`, `http://sourceforge.net/projects/openephyra/`

source text shows the fragment of the document where the answer can be located. Experimental answer is the answer returned by the auto-frame-based system. OpenEphyra's answer is the answer from the OpenEphyra question answering system. Discussion describes any challenges that would lead to correctly or incorrectly answering the question.

Reading Comprehension Sample Results - Table 7.27

Question	Who is Christopher Robin?
Correct Answer	the person you read about in Winnie the Pooh
Source Text	He is the same person that you read about in the book, Winnie the Pooh
Experimental Answer	He is the same person that you read about in the book, Winnie the Pooh
OpenEphyra's Answer	Chris
Discussion	Correctly answering this type of question is a matter of selecting a sentence which describes Christopher Robin as a person ("Christopher Robin is [person]").
Question	When was Winnie the Pooh written?
Correct Answer	1925
Source Text	Winnie the Pooh was written in 1925
Experimental Answer	1925
OpenEphyra's Answer	1925
Discussion	Answering this question simple matching the context of the question with something in the text, then extracting the date. OpenEphyra also correctly answers this question.

Reading Comprehension Sample Results - Table 7.27

Question	Where did young Chris live?
Correct Answer	Cotchfield Farm in England
Source Text	As a boy, Chris lived in a pretty home called Cotchfield Farm
Experimental Answer	England Or a pretty home called Cotchfield Farm
OpenEphyra's Answer	pretty home called Cotchfield Farm
Discussion	Ideally this question requires the understanding that *young Chris* refers to *Chris as a boy*. This system select ones of the two answers listed above[10], which are partially correct, because it does not consider temporal context or account for the adjective *young* in the query.
Question	Who is the captain of the sailboat?
Correct Answer	Thor
Source Text	The captain's name is Thor
Experimental Answer	Thor
OpenEphyra's Answer	Kon Tiki
Discussion	The source text does not specify that Thor is the captain of a sailboat, but one sense of *captain* has a *ship* attachment point. It is implied in this sense that there exists a *ship* with which the *captain* is associated. In *LDOCE*, *ship* and *sailboat* are not connected as hypernyms, both may be connected as types of *vehicles* or types of *boats* (depending on the sense of *ship*). Here we see evocation of implied knowledge, modulation, and partial matching playing a role in selecting the correct answer. Discourse analysis could have connected the implied *ship* associated with the *captain* to other references to ship in the text.

[10]The responses from different runs may differ due to internal timing and other circumstantial factors.

Reading Comprehension Sample Results - Table 7.27

Question	What is the name of the boat?
Correct Answer	the Kon Tiki
Source Text	The name of their boat is the Kon Tiki
Experimental Answer	Kon Tiki
OpenEphyra's Answer	Kon Tiki
Discussion	This is a straight forward matching of query graph to sub graph from the text.

Table 7.27: Reading Comprehension Sample Results

7.4 Conclusions

The initial intrinsic evaluation shows that the graphs from which the auto-frames are built are consistent with the original intent. Furthermore, the representation and the genus detection algorithm are consistent with the simpler LF (Rus, 2004). This consistency gives confidence that the initial graph-based representations and annotations are correct. Furthermore, it seems that connection search can provide good contextualization, creating attachments between neighbouring auto-frames and allowing nouns to modulate verbs. The connection search algorithm is clearly unrefined but shows potential, both in the construction of auto-frames and in their application and contextualization.

While the extrinsic results are not conclusive or state of the art, they do demonstrate that auto-frames and connection search can be applied to the task of reading comprehension. Furthermore the results show precision at a level comparable to other published results. A fully developed and refined system should be able to achieve higher recall. The results also show the benefits of using multiple interpretations, which maximized recall without major loss of precision. Using multiple interpretations, with pruning during the reading comprehension task, maximized precision while providing better recall than a single interpretation. Lastly the use of auto-frames, instead of lemmatization and dependencies, doubles the system's recall, suggesting that it better represents the data than a simple representation.

The next chapter reviews the contributions of this work and describes future work

that could build on this research.

Chapter 8

Conclusions and Future Work

8.1 Summary

This thesis has proposed a solution to a previously unaddressed problem in Natural Language Processing (NLP): "Can one construct automatically a knowledge base using a representation that supports looser meaning of words, including connotation, likely outcomes, relaxation of selectional restrictions, and common sense?" Loose definitions have two key aspects: evocation and contextualization. A word *evokes* more than its definition. It brings up context, as well as likely and expected information that need not be true in every situation. This work encodes such additional information in auto-frames. *Contextualization* is the process of adapting information from the auto-frame to fit a specific context (sentence or sentences). Primarily, this involves removing from the auto-frame some of the additional information that does not fit the context.

One significant challenge in this thesis was to implement such a solution in a tractable manner. The construction of the knowledge base required a few days of computer time, and a task-based evaluation runs in about an hour.[1] This thesis demonstrated a proof-of-concept representation, lexical knowledge structure, and algorithm – Multiple Interpretation Graph (MIG), auto-frames, and connection search – which used knowledge extracted from *Longman Dictionary of Contemporary English* (*LDOCE*) and applied it to the task of reading comprehension.

A new representation developed in this thesis, MIG, supports multiple interpretations and two layers of information: dependency parse trees labelled with auto-frames (senses) and attachment points (semantic roles). Maintaining multiple valid interpretations in the

[1] The original unoptimized attempts took several weeks and days respectively.

task-based evaluation showed an increased recall – without significant loss of precision – over keeping the single highest-ranked interpretation. A new lexical knowledge structure called auto-frames was developed, to represent the evoked meaning of a word, so that a symbolic representation of meaning can be modulated and contextualized. The task-based evaluation showed that using this lexical knowledge structure doubled recall over using dependencies and lemmatized words.

A technique for constructing auto-frames called connection search was developed. It can also be used to assign senses and make attachments. *Attachments* are similar to semantic roles in that they define relations between words, *attachment points* are like slots in auto-frames that are filled by attachments to other auto-frames. Auto-frames were extracted from *LDOCE*. A demonstration of the application of auto-frames to reading comprehension was provided.

This research was broad and exploratory, requiring the study of several fields and extensive software development. It tested and used a wide variety of software and tools, such as Java, MYSQL, Stanford CoreNLP, JUNG, and OpenEphyra. Because of the wide scope of this research, from literature review to theory and then knowledge extraction, followed by a proof of concept, most of the proof of concept was developed in an ad-hoc manner. This work has demonstrated a novel approach to building lexical knowledge bases, a novel representation, and the applicability of such knowledge to the task of reading comprehension. Optimization, refinement and extensions of the method remain as future research.

The problems examined in this research, particularly contextualization and the way in which words evoke meanings, are related to symbolic representations. For the most part, symbolic representations have not supported evoked knowledge and contextualization. Symbolic representations, however, usually have the benefits of supporting composition of knowledge and logical operations. While this work tried to make symbolic processing more flexible, current research (Grefenstette, 2013; Van de Cruys et al., 2013; Cohen et al., 2013; Turney, 2013) seeks statistical approaches, which also support logical operations and composition.

The proposed representation (MIG) and the algorithm (connection search) were designed to be more flexible than traditional symbolic approaches. By design and demonstration, they support evocation of meaning, modulation, and contextualization. While linguistic literature discusses such ideas, there exists little literature in the realm of NLP.

FrameNet is built on the idea of frame semantics, but it is primarily used as an inventory of senses and roles, where most usages simply assign labels to surface concepts.

FrameNet is manually built, so it is of a high quality, but has limited coverage of the English language[2] and many lexical units evoke generic frames. This work has modeled evocation by expanding a lexical unit to its auto-frame, and subsequently contextualizing the auto-frame. This work has more lexical units and senses than *FrameNet*, and is more quickly built, but is of a lower quality.[3] *FrameNet* has been proven to be useful in NLP, while this research has so far only demonstrated some of its potential utility.

8.2 Contributions

This thesis intersects with many research areas: knowledge representation; knowledge extraction, such as string pattern matching, Semantic Role Labeling (SRL), and Word Sense Disambiguation (WSD); knowledge-base population; linguistics and cognitive science, with regards to the meaning of words; and question answering. However, the contributions deal with lexical ambiguity in a symbolic representation that supports contextualization, which is a previously unaddressed problem.

The contributions of this thesis are as follows:

1. A method of detecting genus terms in dictionary definitions was demonstrated as having good precision and moderate recall.

2. A method of classifying definitions into different types was demonstrated, though no evaluation was provided.

3. A knowledge representation that supports multiple interpretations, MIG, was defined and tested. The representation stores the equivalent of multiple graphs into a single structure. It groups sets of common elements (concepts, relations, and links) indicating how they are valid or invalid within an interpretation using restrictions. The representation supports expanding concepts to their auto-frames or definitions instead of labeling them with senses. This work has demonstrated that carrying multiple interpretations into entailment and reading comprehension can improve recall without significantly affecting precision. An intrinsic evaluation also demonstrated a number of circumstances where more than one interpretation may be valid for a single text.

[2] Approximately 10,000 lexical units and more than 1000 frames as of April 2016.

[3] There are approximately 47,000 unique lexical units in this work and only approximately 10,000 in *FrameNet*.

4. Another representation, auto-frames, supports words evoking complex meaning and contextualization. Applying expansion with auto-frames adds implicit concepts and arguments, including some limited common-sense information. The extracted knowledge was shown to be consistent with the simpler Logic Form (LF) representation. The intrinsic evaluation and the in-depth review of reading comprehension contained examples where adding knowledge provided important implied information. The addition of attachment points and auto-frames to reading comprehension improved recall over the simple lemmatization and dependency matching.

5. A method called connection search connects auto-frames and then contextualizes them. In particular, the method supports addition and removal of information depending on the context. The use of a pruned MIG in reading comprehension showed the highest precision. The intrinsic evaluation showed that different sentences and expressions can evoke the same MIG and interpretation.

6. Definitions were used to avoid arbitrary selection of semantic roles. Instead of using an arbitrary predefined set of semantic roles, attachment points were identified within definitions; they are comparable to roles or arguments. The intrinsic evaluation suggested that the majority of core frame elements in *FrameNet* have matching attachment points in auto-frames. Auto-frames will even at times have attachment points that can fill/satisfy the core frame elements of general frames.

7. Auto-frames provide better lexical coverage than *FrameNet* and are more specific to each lexical unit, while still supporting comparisons of similar meanings. The intrinsic evaluation emphasized the greater lexical coverage of the auto-frames. Most *FrameNet* frames are very general and do not describe the specific lexical units in detail. The intrinsic evaluation demonstrates how connection search can produce similar representations for similar meanings. The entailment module was shown, using the unit tests, to be able to compare alike expressions.

8. The source code and data are available at `http://martin.scaiano.com/Auto-frames.html`.

8.3 Future Work

The issues raised by this research are good for years of further study. There are many areas of future work; some are direct improvements of the basic methodology, while

others begin to expand into new areas of research.

8.3.1 Leveraging the definitions of prepositions

This research took a naive approach to connecting dependencies with attachment points. Little consideration was given to the effect of a preposition on the attachment, with the exception of matching or prioritizing matched prepositions. Most dictionaries have definitions for prepositions, which enumerate their semantic relations. Like adjuncts or adjectives, a preposition can select the head it modifies.

The definitions of prepositions could improve auto-frames by creating attachment points for adjunct phrases. Expanding prepositions using connection search could also have a similar effect, attaching adjunct phrases to the auto-frame, in a manner similar to how a *car* has an attachment point for *traveling* or *go*. Could the definition of prepositions be used to define more semantic attachment points?

8.3.2 Connecting adjective, adverbs, and other parts of speech

The connection search technique currently does not connect adjectives and adverbs with their heads. This requires a slightly different method than is used with verbs and nouns, which have attachment points for their dependencies. Instead, the dependent word (adjective or adverb) has restrictions on what types of words it can modify.

In a definition, finding the correct place to connect the head can be difficult. Developing this algorithm would be likely to require extensive testing.

8.3.3 Implicit arguments and discourse

Auto-frames could be applied in processing discourse. How and why are adjacent sentences connected? For example, "We went to Toronto. The car was comfortable." It is implied that the car was the means of travel; this model which expands concepts with connection search would include a statement that the *car* is used for *travel*, and *went* could be connected to *travel* and thus to *car*. However, a method of detecting and attaching concepts across clauses has not been defined.

Such a study is outside the scope of this thesis: it enters into the area of discourse analysis. Discourse analysis and connecting implicit arguments across sentences could produce very practical performance improvements in recognizing entailment, Question-Answering (QA), and reading comprehension. Such research could also include methods

of connecting anaphora with implicit arguments, and connecting clauses in the same sentence.

8.3.4 Corpus-based enhancements

This research was founded on building a lexical knowledge base from a dictionary. Dictionaries have many advantages over corpus-based techniques, such as portability, since they do not require an annotated corpus and since they exist for most languages. Yet much can still be learned from corpora and distributional semantics: common-sense knowledge, distributions of various realizations of an auto-frame, weights and statistics that could be used in connection search.

8.3.5 Other Dictionaries

Ideally, the the knowledge extraction technique would have been applicable to other dictionaries, assuming some normalization or preprocessing. Since most of the extraction and connection search technique was based on dependency parse trees, good portability can be expected. As evidence that some of the techniques here are portable, much of the research in this thesis built parallel knowledge bases using *LDOCE* and *WordNet*.

Definitions in *WordNet* do not have the simplicity and consistency of definitions in *LDOCE*, which was important in finding modulated and contextualized senses, and in identifying attachment points. Since *WordNet* is a semantic network, a lexical knowledge base, and a thesaurus, but not a dictionary, it is perhaps not surprising that such problems appear. Though other dictionaries may have an appropriate structure, a number of patterns would need to be adapted.

8.3.6 Merging Dictionaries

One of the original goals of this thesis was to merge dictionaries, but it was not feasible in the timeframe of my degree program. Once a lexical knowledge base has been produced using one dictionary, it should be possible to interpret a second dictionary using the knowledge from the first. This could lead to simple techniques for merging dictionaries into a single knowledge base.

Glossary

auto-frame Auto-frames are the knowledge structure used by this thesis to store the meaning of word or lexeme. While they are inspired by and similar to semantic frames and predicate-argument structures, there are many differences. Auto-frames are described in detail in Section 4.5.1. ii, iii, v, vi, ix, 4, 5, 31, 36–38, 41–44, 48–50, 52, 55–67, 70–74, 85, 90–96, 100, 104, 106–108, 110–118, 122, 124, 126, 127, 136–143, 152, 156–162, 165, 166, 169, 170, 172, 174–181

bipartite graph A graph with two types of vertices. Any vertex can only be connected with vertices of the opposite type. 27

Conceptual Graph "Conceptual graphs (CGs) (Sowa, 1984) are a system of logic based on the existential graphs of Charles Sanders Peirce (1909) (See Roberts, 1973, on this subject) and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable"[4]. 11, 24, 26–29, 42, 43, 74, 78, 182, 206, 215

contextualization In this work, contextualization refers to a process which modifies an auto-frame (or the meaning of a word) to something more appropriate for a given context. Chapter 4 discusses the topic through-out; Section 4.6.1 gives a practical illustration. 60, 65, 68

interpretation In this work, an interpretation refers to a representation with a specific set of senses (auto-frames) and roles (attachments) assigned. Multiple Interpretation Graphs support multiple interpretations and are not restricted to assigning one sense or role for each concept. 42, 43

[4] 2011, `http://conceptualgraphs.org/`

knowledge spiral The knowledge spiral refers to the problem that the knowledge and language require knowledge and language to understand and learn; thus a system must first know a few (hundred or thousand) central terms then the system can iteratively expand out and learn more terms and knowledge. 21

Machine Readable Dictionary A dictionary in electronic form which may be processed by a program. A minimum requirement would be that a program could index and navigate the entries. 21, 182

modulation Modulation is the subtle adaptation or emphasis of a sense to produce a similar meaning that better fits a context. See page 18 for more details and examples. 18, 20, 65, 66

Multiple Interpretation Graphs Multiple Interpretation Graphs (MIGs) are the representation used in this thesis. They start from dependency graphs, but can include multiple graphs, relations between graphs and nodes, and dependencies or restrictions on interpretation. MIGs are introduced in Section 4.4.2. ii, v, 4, 31, 36, 43–46, 48–53, 55–57, 65–68, 70, 72, 73, 82, 85, 90, 91, 94, 96, 98, 99, 111–115, 117–120, 122, 123, 143–145, 166, 174–177, 182

person-month A unit of measure of work or effort which is about the amount of work performed by an average person in a month. 7

soft definition a soft definition is first defined in Section 3.5, as a definition that allows the meaning to be slightly altered by context. Why auto-frames are considered soft is explained in Section 4.2.1. 24, 38

Web Ontology Language (OWL) An XML based formal logic representation intended for use on the Web and with a focus on sharing and reuse of ontologies. 24, 30, 42, 43, 183

Word Sense Disambiguation The task of assigning the intended sense to word in context. 8, 10, 12, 30, 43, 65, 90, 96, 110, 122, 137, 176, 183

Acronyms

LDOCE *Longman Dictionary of Contemporary English*. ii, ix, x, 11, 15–18, 21, 24, 26, 37–40, 53, 56, 58–61, 63, 65–68, 70, 73, 75, 76, 80, 83, 84, 87, 89, 92, 98, 106, 107, 109, 113, 120–122, 136, 139, 141, 158–161, 171, 174, 175, 179, 232

AI Artificial Intelligence. 2, 13

CG Conceptual Graph. 11, 24, 26–29, 42, 43, 74, 78, 206, 215

CLEF Cross-Language Evaluation Forum. 34

IE Information Extraction. 1, 2

IR Information Retrieval. 1, 2, 9, 32, 33

KB Knowledge Base. 30

LF Logic Form. vi, xi, 5, 11, 26, 37, 79, 126–132, 168, 172, 177, 200, 203, 206, 208, 210, 212–214

LFi Logic Form Identification. 127, 212, 213

MIG Multiple Interpretation Graph. ii, v, 4, 31, 36, 43–46, 48–53, 55–57, 65–68, 70, 72, 73, 82, 85, 90, 91, 94, 96, 98, 99, 111–115, 117–120, 122, 123, 143–145, 166, 174–177

MRD Machine Readable Dictionary. 21

NE Named Entity. 34

NER Named Entity Recognition. 74, 92

NLP Natural Language Processing. 5, 7, 8, 13, 17, 18, 20, 23, 30, 37, 48, 74, 106, 174–176

OMCS Open Mind Common Sense. 10

OWL Web Ontology Language. 24, 30, 42, 43

POS part of speech. 8, 45, 74, 77, 78, 86, 88, 109, 133, 206

QA Question-Answering. x, 1, 2, 7, 9, 11, 12, 32–34, 120, 164, 169, 178

SRL Semantic Role Labeling. 2, 7–9, 12, 30, 33, 34, 43, 90, 92, 176

TREC Text REtrieval Conference. 33, 169

W3C World Wide Web Consortium. 30

WSD Word Sense Disambiguation. 8, 10, 12, 30, 43, 65, 90, 96, 110, 122, 137, 176

Bibliography

Agirre, E., De Lacalle, O. L., and Soroa, A. (2009). Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1501–1506. Morgan Kaufmann Publishers Inc.

Agirre, E. and Soroa, A. (2009). Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, pages 33–41.

Ahn, D., Fissaha Adafre, S., Jijkoun, V., and Rijke, M. d. (2004). The University of Amsterdam at Senseval-3: Semantic roles and logic forms. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 49–53.

Akbik, A. and Löser, A. (2012). KrakeN: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. Association for Computational Linguistics.

Aliprandi, C., Ronzano, F., Marchetti, A., Tesconi, M., and Minutoli, S. (2011). Extracting events from wikipedia as RDF triples linked to widespread semantic web datasets. In *OCSC'11: Proceedings of the 4th international conference on Online communities and social computing*, pages 90–99. Springer-Verlag.

Amsler, R. (1981). A taxonomy for English nouns and verbs. In *Proceedings of the 19th Annual Meeting of Association for Computational Linguistics*, pages 133–138.

Anthony, S. and Patrick, J. (2004). Dependency based logical form transformations. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 54–57, Barcelona, Spain.

Babko-Malaya, O., Palmer, M., and Xue, N. (2004). Proposition Bank II: Delving Deeper. In *The 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2004) Workshop: Frontiers in Corpus Annotation*, pages 17–23, Boston, Massachusetts, USA.

Baker, C., Ellsworth, M., and Erk, K. (2007). SemEval'07 Task 19: Frame Semantic Structure Extraction. *Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics*, pages 99–104.

Baker, C., Fillmore, C. J., and Lowe, J. (1998). The Berkeley Framenet project. *Proceedings of the 17th International Conference on Computational Linguistics (COLING '98)*, pages 86–90.

Banerjee, S. and Pedersen, T. (2002). An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLing '02)*, pages 126–135.

Barker, K., Chaudhri, V., Chaw, S., Clark, P., Fan, J., Israel, D., Mishra, S., Porter, B., Romero, P., Tecuci, D., and Yeh, P. (2004). A question-answering system for AP Chemistry: Assessing KR&R technologies. In *Proceedings of The Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR2004)*, pages 488–497. University of Texas at Austin.

Barker, K., Porter, B., and Clark, P. (2001). A Library of Generic Concepts for Composing Knowledge Bases. In *Proceedings of the 1st International Conference on Knowledge Capture (K-CAP '01)*, pages 14–21.

Barrière, C. (1997). *From a children's first dictionary to a lexical knowledge base of conceptual graphs*. PhD thesis, Simon Fraser University.

Barrière, C. and Popowich, F. (1999). An Iterative Construction Approach for Lexical Knowledge Bases. *Provisionally accepted (May 2000) in Computational Intelligence Journal but not resubmitted.*

Bayer, S., Burger, J., and Greiff, W. (2004). The Mitre logical form generation system. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 69–72.

Bejan, C. A. and Hathaway, C. (2007). UTD-SRL: a pipeline architecture for extracting frame semantic structures. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 460–463. Association for Computational Linguistics.

Bendaoud, R., Napoli, A., and Toussaint, Y. (2008). Formal Concept Analysis: A Unified Framework for Building and Refining Ontologies. In *Knowledge Engineering: Practice and Patterns*, pages 156–171. Springer Berlin Heidelberg.

Bentivogli, L., Clark, P., Dagan, I., and Dang, H. T. (2011). The seventh pascal recognizing textual entailment challenge. In *Proceedings of the Seventh PASCAL Recognizing Textual Entailment Challenge, 2011, TAC 2011 Workshop*, Gaithersburg, Maryland, USA.

Betteridge, J., Carlson, A., Hong, S. A., Hruschka Jr, E. R., Law, E. L. M., Mitchell, T. M., and Wang, S. H. (2009). Toward Never Ending Language Learning. *AAAI Spring Symposium Learning by Reading and Learning to Read*, pages 1–2.

Borin, L., Forsberg, M., and Lönngren, L. (2013). SALDO: a touch of yin to WordNet's yang. *Language Resources and Evaluation*, 47(4):1191–1211.

Butnariu, C., Kim, S. N., Nakov, P., Séaghdha, D. Ó., Szpakowicz, S., and Veale, T. (2010). SemEval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 100–105. Association for Computational Linguistics.

Chambers, N. and Jurafsky, D. (2008). Unsupervised Learning of Narrative Event Chains. *ACL*, pages 789–797.

Charniak, E., Altun, Y., Braz, R., Garrett, B., Kosmala, M., Moscovich, T., Pang, L., Pyo, C., Sun, Y., Wy, W., Yang, Z., Zeller, S., and Zorn, L. (2000). Reading comprehension programs in a statistical-language-processing class. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems*, pages 1–5.

Cheung, J. C. K., Poon, H., and Vanderwende, L. (2013). Probabilistic Frame Induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Associ-*

ation for Computational Linguistics: Human Language Technologies, pages 837–846. Association for Computational Linguistics.

Clark, P. and Harrison, P. (2010). Machine reading as a process of partial question-answering. In *FAM-LbR '10: Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 1–9. Association for Computational Linguistics.

Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., and Wojcik, R. (2006). From WordNet to a Knowlege Base. *AAAI Spring Symposium Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 10–15.

Cobuild, C. (2006). *Collins COBUILD advanced learner's English dictionary*. Collins Cobuild.

Cohen, S. B., Satta, G., and Collins, M. (2013). Approximate PCFG Parsing Using Tensor Decomposition. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 487–496, Atlanta, Georgia. Association for Computational Linguistics.

Compton, P. and Jansen, B. (1990). A Philosophical Basis for Knowledge Acquisition. *Knowledge acquisition*, 2(3):241–258.

Cruse, D. (1986). Lexical semantics. *Cambridge University Press*.

Cruse, D. (1995). Polysemy and related phenomena from a cognitive linguistic viewpoint. *Saint-Dizier and Viegas*, pages 33–49.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. (2009). Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15:459–476.

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Proceedings of the First international conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment (MLCW '05)*, pages 177–190. Springer-Verlag.

Dang, H. T., Lin, J., and Kelly, D. (2007). Overview of the TREC 2007 question answering track. In *The Fifteenth Text REtrieval Conference (TREC 2006)*, page 63.

Das, D., Chen, D., Martins, A. F. T., Schneider, N., and Smith, N. A. (2014). Frame-Semantic Parsing. *Computational Linguistics*, 40(1):9–56.

de Marneffe, M., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454.

de Marneffe, M.-C. and Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Delmonte, R. (2004). Text understanding with GETARUNS for Q/A and summarization. In *TextMean '04: Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, pages 97–104. Association for Computational Linguistics.

Delmonte, R. (2013). Coping With Implicit Arguments And Events Coreference. In *Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 1–10, Atlanta, Georgia. Association for Computational Linguistics.

Escolano, F., Bonev, B., and Lozano, M. (2011). Information-Geometric Graph Indexing From Bags of Partial Node Coverages. In *Proceedings of Graph-Based Representations in Pattern Recognition: 8th IAPR-TC-15 International Workshop, (GbRPR 2011)*, pages 52–61. Springer.

Etzioni, O., Banko, M., and Cafarella, M. J. (2006). Machine Reading. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence, AAAI 2006*, pages 1517–1519.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Ferrucci, D. (2010). Build Watson: an overview of DeepQA for the Jeopardy! challenge. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques (PACT 2010)*, pages 1–2, Vienna, Austria.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79.

Fillmore, C. (1976). Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Foxvog, D. (2010). Cyc. In *Theory and Applications of Ontology: Computer Applications*, pages 259–278. Springer Netherlands.

Gangemi, A., Presutti, V., and Reforgiato Recupero, D. (2014). Frame-based detection of opinion holders and topics: a model and a tool. *Computational Intelligence Magazine, IEEE*, 9(1):20–30.

Giampiccolo, D., Dang, H. T., Magnini, B., Dagan, I., Cabrio, E., and Dolan, B. (2009). The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Fourth PASCAL Recognizing Textual Entailment Challenge (TAC 2008)*.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (RTE-3)*, pages 1–9. Association for Computational Linguistics.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D. (2007). SemEval-2007 task 04: classification of semantic relations between nominals. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 13–18. Association for Computational Linguistics.

Grefenstette, E. (2013). Towards a Formal Distributional Semantics: Simulating Logical Calculi with Tensors. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 1–10, Atlanta, Georgia, USA. Association for Computational Linguistics.

Grois, E. and Wilkins, D. C. (2005). Learning strategies for story comprehension: a reinforcement learning approach. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 257–264. ACM.

Guan-yu, L., Shu-peng, L., and Yan, Z. (2010). Formal concept analysis based ontology merging method. In *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), 2010*, pages 279–282. IEEE.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. (2006). The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment (RTE-2)*.

Havasi, C., Speer, R., and Alonso, J. (2007). ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent advances in natural language processing*, pages 27–29.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Séaghdha, D. Ó., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2010). SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Hirsch Jr, E. D. (2003). Reading comprehension requires knowledge—of words and the world. *American Educator*, 27(1):10–13.

Hirschman, L., Light, M., Breck, E., and Burger, J. D. (1999). Deep Read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*, pages 325–332.

Hobbs, J. R. and Pan, F. (2006). Time-Ontology in OWL - W3C Working Draft 27. Available online at http://www.w3.org/TR/owl-time/.

Ide, N. and Véronis, J. (1995). Knowledge extraction from machine-readable dictionaries: An evaluation. In *Proceedings of the Third International EAMT Workshop on Machine Translation and the Lexicon*, pages 17–34.

Johansson, R. and Nugues, P. (2007). LTH: semantic structure extraction using nonprojective dependency trees. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 227–230. Association for Computational Linguistics.

Johnson, C. R., Fillmore, C. J., Petruck, M. R., Baker, C. F., Ellsworth, M., Ruppenhofer, J., and Wood, E. J. (2002). Framenet: Theory and practice.

Kabbaj, A. (2006). Development of Intelligent Systems and Multi-Agents Systems with Amine Platform. In *Conceptual Structures: Inspiration and Application*, pages 286–299. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kilgarriff, A. (1992). Dictionary word sense distinctions: An enquiry into their nature. *Computers and the Humanities*, 26:365–387.

Kilgarriff, A. (1997). " I Don't Believe in Word Senses". *Computers and the Humanities*, 31:91–113.

Kim, D., Barker, K., and Porter, B. (2010). Building an end-to-end text reading system based on a packed representation. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 10–14.

Kim, S.-M. and Hovy, E. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8. Association for Computational Linguistics.

Klinov, P. and Parsia, B. (2010). Pronto: A practical probabilistic description logic reasoner. In *First International Workshop on Uncertainty in Description Logics (UniDL'1.0)*, pages 59–79.

Lenat, D. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38:33–38.

Lenat, D. B., Miller, G. A., and Yokoi, T. (1995). CYC, WordNet, and EDR: Critiques and Responses. *Commun. ACM*, 38(11):45–48.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

Lin, D. and Pantel, P. (2001). DIRT - Discovery of Inference Rules from Text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328.

Litkowski, K. (2007). CLR: Integration of FrameNet in a Text Representation System. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 113–116. Association for Computational Linguistics.

Liu, Q., Xu, K., Zhang, L., Wang, H., Yu, Y., and Pan, Y. (2008). Catriple: Extracting Triples from Wikipedia Categories. In *ASWC '08: Proceedings of the 3rd Asian*

Semantic Web Conference on The Semantic Web, pages 330–344, Berlin, Heidelberg. Springer-Verlag.

Loni, B. (2011). A Survey of State-of-the-Art Methods on Question Classification. Technical report, Delft University of Technology.

Mihalcea, R. (2001). Extended Wordnet: Progress report. In *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, pages 95–100.

Mihalcea, R. (2007). Using Wikipedia for automatic word sense disambiguation. In *Proceedings of The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies workshop (NAACL-HLT '07)*, pages 196–203.

Mihalcea, R. and Moldovan, D. I. (2001). EZ.WordNet: Principles for Automatic Generation of a Coarse Grained WordNet. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press.

Milne, D. N., Witten, I. H., and Nichols, D. M. (2007). *A knowledge-based search engine powered by Wikipedia*. ACM, New York, New York, USA.

Milo, T. and Suciu, D. (1999). Index Structures for Path Expressions. In *Proceedings of the 7th International Conference on Database Theory*, pages 277–295.

Minsky, M. (1975a). A Framework for Representing Knowledge. *The Psychology of Computer Vision*.

Minsky, M. (1975b). Minskys frame system theory. In *TINLAP*, volume 75, pages 104–116.

Mitchell, T. (2010). Never-Ending Learning. Technical report, CARNEGIE-MELLON UNIV, PITTSBURGH PA.

Mohammad, S., Zhu, X., and Martin, J. (2014a). Semantic role labeling of emotions in tweets. In *Proceedings of WASSA*, pages 32–41.

Mohammad, S. M., Zhu, X., and Martin, J. (2014b). Semantic role labeling of emotions in tweets. In *Proc of WASSA*, pages 32–41.

Mohammed, A., Moldovan, D., and Parker, P. (2004). Senseval-3 logic forms: A system and possible improvements. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 163–166.

Moldovan, D. and Rus, V. (2001). Explaining answers with extended WordNet. *Proceedings of the 39th Annual Meeting of Association of Computation Linguistics*.

Navigli, R. (2006). 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 '06)*. Association for Computational Linguistics.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Navigli, R., Litkowski, K. C., and Hargraves, O. (2007). SemEval-2007 Task 07: Coarse-grained English All-words Task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35.

Neely, J. H. (1976). Semantic priming and retrieval from lexical memory: Evidence for facilitatory and inhibitory processes. *Memory & cognition*, 4(5):648–654.

Nivre, J., Hall, J., and Nilsson, J. (2006). MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 2216–2219.

Osman, A. H., Salim, N., Binwahlan, M. S., Alteeb, R., and Abuobieda, A. (2012). An improved plagiarism detection scheme based on semantic role labeling. *Appl. Soft Comput. ()*, 12(5):1493–1502.

Palmer, M., Dang, H. T., and Fellbaum, C. (2006). Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(02):137–163.

Palmer, M., Gildea, D., and Xue, N. (2010). *Semantic Role Labeling*, volume 3 of *Synthesis Lectures on Human Language Technologies*. Mogan & Claypoole.

Peirce, C. S. and Sowa, J. (2010). Existential Graphs: MS 514 by Charles Sanders Peirce with commentary by John Sowa, 1908, 2000. Available online at http://www.jfsowa.com/peirce/ms514.htm.

Peñas, A. and Hovy, E. (2010). Semantic enrichment of text with background knowledge. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 15–23. Association for Computational Linguistics.

Peñas, A., Hovy, E., Forner, P., Rodrigo, Á., Sutcliffe, R. F. E., Forascu, C., and Sporleder, C. (2011). Overview of QA4MRE at CLEF 2011: Question Answering for Machine Reading Evaluation. In *Proceeding of CLEF 2011 Labs and Workshop*, pages 1–20.

Peñas, A., Hovy, E. H., Forner, P., Rodrigo, Á., Sutcliffe, R. F. E., and Morante, R. (2013). QA4MRE 2011-2013: Overview of Question Answering for Machine Reading Evaluation. *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 303–320.

Pennacchiotti, M., De Cao, D., Basili, R., Croce, D., and Roth, M. (2008). Automatic induction of FrameNet lexical units. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing EMNLP '08*, pages 457–465. Association for Computational Linguistics.

Petermann, H., Euler, L., and Bontcheva, K. (1997). CGPro - a Prolog Implementation of Conceptual Graphs. *Universitaet Hamburg Hamburg, Germany*.

Pozo, P. M., Sanz, M. P., and Cueto, A. S. (2004). Assignment of Semantic Roles Based on Word Sense Disambiguation. *Advances in Artificial Intelligence–IBERAMIA 2004*, pages 256–265.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow Semantic Parsing using Support Vector Machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL 2004)*, page 8.

Procter, P. (1978). *Longman Dictionary of Contemporary English*. Longman, Harlow, England.

Rais-Ghasem, M. and Corriveau, J. P. (1998). Exemplar-Based Sense Modulation. In *Proceedings of COLING-ACL '98: Workshop on the Computational Treatment of Nominals*.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, pages 133–142.

Richardson, S. D., Dolan, W. B., and Vanderwende, L. (1998). MindNet: acquiring and structuring semantic information from text. In *Proceedings of the 17th international conference on Computational linguistics (ACL '98)*, pages 1098–1102.

Richens, T. (2008). Anomalies in the WordNet verb hierarchy. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 729–736. Association for Computational Linguistics.

Riloff, E. (2000). A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems*, pages 13–19.

Riloff, E. and Thelen, M. (2000). A rule-based question answering system for reading comprehension tests. In *ANLP/NAACL 2000 Workshop*, pages 13–19, Morristown, NJ, USA. Association for Computational Linguistics.

Roberts, D. D. (1973). *The Existential Graphs of Charles S. Peirce*. Mouton de Gruyter.

Rodrigo, Á., Peñas, A., Hovy, E., and Pianta, E. (2010). Question answering for machine reading evaluation. In *Proceedings of CLEF 2010*.

Rus, V. (2001). High precision logic form transformation. In *Proceedings of the 13th International Conference with Tools in Artificial Intelligence*, pages 288–295.

Rus, V. (2002). *Logic forms for Wordnet glosses*. PhD thesis, Southern Methodist University, Dallas, TX.

Rus, V. (2004). A first evaluation of logic form identification systems. In *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL-3)*, pages 7–40.

Rus, V. (2005). A Method to Generate Large Common-Sense Knowledge Bases from Online Lexical Resources. In *The Florida AI Research Society (FLAIRS '05)*, pages 635–640.

Schank, R. and Abelson, R. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates.

Schank, R. C. and Abelson, R. P. (1975). Scripts, plans, and knowledge. In *Proceedings of the 4th international joint conference on Artificial intelligence-Volume 1*, pages 151–157. Morgan Kaufmann Publishers Inc.

Schlaefer, N., Gieselmann, P., Schaaf, T., and Waibel, A. (2006). A pattern learning approach to question answering within the ephyra framework. In *Proceedings of the 9th international conference on Text, Speech and Dialogue (TSD'06)*, pages 687–694. Springer-Verlag.

Schlaefer, N., Ko, J., Betteridge, J., and Sautter, G. (2007). Semantic extensions of the Ephyra QA system for TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, pages 332–341.

Schubert, L. (2002). Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research (HLT '02)*, pages 94–97. Morgan Kaufmann Publishers Inc.

Schuler, K. (2005). *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, University of Pennsylvania.

Sharma, A. B. and Forbus, K. D. (2013). Automatic Extraction of Efficient Axiom Sets from Large Knowledge Bases. *AAAI 2013*.

Shi, L. and Mihalcea, R. (2005). Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of the 6th International Conference Computational Linguistics and Intelligent Text Processing (CICLing 2005)*, pages 100–111.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., and Zhu, W. L. (2002). Open Mind Common Sense: Knowledge Acquisition from the General Public. *CoopIS/DOA/ODBASE*, 2519(Chapter 77):1223–1237.

Sirin, E., Parsia, B., Grau, B., and Kalyanpur, A. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

Sowa, J. (1984). *Conceptual structures: information processing in mind and machine*. Addison-Wesley Longman Publishing Co.

Sowa, J. (2000). Knowledge representation: logical, philosophical, and computational foundations. *MIT Press*, 27(2):286–294.

Sun, R., Jiang, J., Fan, Y., Hang, T., Tat-seng, C., and Kan, M. Y. (2005). Using syntactic and semantic relation analysis in question answering. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005)*.

Toba, H., Adriani, M., and Manurung, R. (2011). Expected answer type construction using analogical reasoning in a question answering task. In *Proceedings of International Conference on Advanced Computer Science and Information System (ICACSIS 2011)*, pages 283–290.

Turney, P. D. (2013). Distributional semantics beyond words: Supervised learning of analogy and paraphrase. *Transactions of the Association for Computational Linguistics*, 1:353–366.

Van de Cruys, T., Poibeau, T., and Korhonen, A. (2013). A Tensor-based Factorization Model of Semantic Compositionality. In *Conference of the North American Chapter of the Association of Computational Linguistics HTL-NAACL*, pages 1142–1151, Atlanta, Georgia.

Vanderwende, L., Kacmarcik, G., Suzuki, H., and Menezes, A. (2005). MindNet: an automatically-created lexical resource. In *Proceedings of HLT/EMNLP on Interactive Demonstrations (HLT-Demo '05)*, pages 8–9.

Vanderwende, L. H. (1995). *The Analysis of Noun Sequences Using Semantic Information Extracted from On-line Dictionaries*. PhD thesis, Georgetown University NW, Washington, DC.

Voorhees, E. M. (2004). Overview of the TREC 2004 Question Answering Track. In *Proceedings of The Thirteenth Text Retrieval Conference (TREC 2004)*, pages 1–11. National Institute of Standards and Technology (NIST).

Voorhees, E. M. (2008). Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of ACL-08: HLT*, pages 63–71.

Wang, D. Z., Chen, Y., Goldberg, S., Grant, C., and Li, K. (2012). Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and human feedback. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 106–110. Association for Computational Linguistics.

Wellner, B., Ferro, L., Greiff, W., and Hirschman, L. (2006). Reading comprehension tests for computer-based understanding evaluation. *Natural Language Engineering*, 12(4):305–334.

Welty, C., Fan, J., Gondek, D., and Schlaikjer, A. (2010). Large scale relation detection. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 24–33. Association for Computational Linguistics.

Wenner, C. (2007). Rule-based Logical Forms Extraction. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA-2007)*, pages 402–409.

Williams, D. W., Huan, J., and Wang, W. (2007). Graph Database Indexing Using Structured Graph Decomposition. In *Proceedings of 23rd International Conference on Data Engineering, 2007. (ICDE 2007)*, pages 976–985.

Wren, J. D. (2011). Question answering systems in biology and medicine–the time is now. *Bioinformatics*, 27(14):2025–2026.

Wu, D. and Fung, P. (2009). Can Semantic Role Labeling Improve SMT? In *13th Annual Conference of the European Association for Machine Translation (EAMT 2009)*, pages 218–225.

Yan, X., Yu, P. S., and Han, J. (2004). Graph Indexing: A Frequent Structure-based Approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data (SIGMOD '04)*, pages 335–346, New York, New York, USA. ACM Press.

Yeh, P., Porter, B., and Barker, K. (2006). A Unified Knowledge Based Approach for Sense Disambiguation and Semantic Role Labeling. In *Proceedings of Twenty-First National Conference on Artificial Intelligence (AAAI '06)*, pages 305–310.

Zhang, S., Hu, M., and Yang, J. (2007). TreePi: A Novel Graph Indexing Method. In *Proceedings of 23rd International Conference on Data Engineering, 2007. (ICDE 2007)*, pages 966–975. IEEE.

Appendix A

Building a Syntactic Logic Form

A.1 Logic Form (LF)

LF is described in detail in Rus's dissertation; it is further expanded by Mohammed et al. (2004) for Senseval 3. Here is a quick overview of the Logic Form (LF) representation.

In LF, words are represented as predicates, and arguments connect related words. The predicates are usually the lemmatized word followed by a colon and a simplified part of speech tag (*e.g.*, `dog:NN(x1)`).

Nouns and adjectives take only one argument that represents the instance's identifier. Nouns usually instantiate their own identifier, as they usually represent independent entities. Since adjectives usually modify nouns, that is, the adjective describes a property of the entity that the noun refers to, its identifier is the value of the noun it modifies.

All verbs have at least three arguments:

- The first argument represents the eventuality of the event, action, or state.
- The second argument represents the syntactic subject.
- The third argument represents the syntactic object.
- If a indirect object is present, it is the fourth argument.

In the representation used at Senseval 3, additional arguments are included after the optional indirect object. These arguments represent predicates connected through prepositions, and their value is the identifier of the head modifier. Rus's thesis does not include prepositional phrases in the verb argument list. In all representations, verbs with prepositional phrases are also connected through the predicate associated with the preposition.

Prepositions are mapped to predicates but with no part of speech component and their name is the preposition. Prepositional predicates have two arguments linking two existing predicates by their identities. The first argument represents the head phrase or verb being modified, while the second argument is the prepositional phrase head.

Consider the verb "give" and the preposition "to" in the following examples:

Jane gave the gorilla a kiss.

Thesis → `Jane:NN(x1) give:VB(e5, x1, x3, x2) gorilla:NN(x2) kiss:NN(x3)`

Senseval → `Jane:NN(x1) give:VB(e5, x1, x3, x2) gorilla:NN(x2) kiss:NN(x3)`

The two representations are identical.

Jane gave a kiss to the gorilla.

Thesis → `Jane:NN(x1) give:VB(e5, x1, x2) kiss:NN(x2) to (e5, x3) gorilla:NN(x3)`

Senseval → `Jane:NN(x1) give:VB(e5, x1, x2, x3) kiss:NN(x2) to (e5, x3)`

`gorilla:NN(x3)`

In the Senseval representation, the verb *give* has an extra argument, the head of the prepositional phrase.

An adverb takes one argument, which represents the identity of what they modify; practically speaking, this is either the eventuality (identifier) of a verb being described, or the identifier of an adjective which is the same identifier of the noun modified by the adjective, (*e.g.*, `run:VB(e1, x1, x4) fast:RB(e1)`).

Conjunctions (or disjunctions) are represented by predicates of the same name, with no part of speech. The first argument represents the union or result of the logical operator; all other arguments represent the elements being operated on. For example, the phrase "dogs and cats" would be represented as `dog:NN(x2) cat:NN(x3) and(x1, x2, x3).`

Noun-noun compounds and collocations are handled slightly differently, though on the surface they appear the same. The main distinction is that collocations exist in *WordNet* and complex nominals do not.[1] A collocation is represented as a single predicate with the name being the words concatenated together with underscores. The part of

[1] A multi-word expression which appears in *WordNet* is treated as collocation, otherwise it is treated as complex nominal.

speech of a collocation is the one associated with it in *WordNet*. Noun-noun compounds are represented by a predicate called "nn" that is similar to conjunctions. The first argument represents the joint or resulting meaning of the subsequent arguments. For example:

United States → `United_States(x1)`
telephone directory → `telephone:NN(x1) directory:NN(x2) nn(x3,x1,x2)`

A comparative is represented as a binary predicate, with no part of speech. The first argument is the predicate identity being compared to the second argument. *e.g.*, a horse is bigger than a dog →`horse:NN(x1) bigger(x1,x2) dog(x2)`. No comparatives exist in the training data from Senseval 3.

Post-modifiers (adjectives appearing after a noun instead of before) are combined with their prepositional predicate; this is shown in the example below from Rus (2002). LF does not support an effective representation of post modifiers, thus by rolling the adjective into an associated prepositional predicate the information is stored but processing is delayed.
Example of a post-modifier:

a semiconductor device capable of amplification → `semiconductor_device:NN(x1) capable_of(x1,x2) amplification:NN(x2)`

Possessive nouns are represented by *pos* binary predicate with no part of speech. The first argument represents the possessor, and the second argument represents the possession. In the Senseval data, the *pos* predicate is replaced with a *'s* predicate. *e.g.*, a man's dog → `man:NN(x1) pos(x1,x2) dog:NN(x2)`).

Possessive pronouns are represented as a predicate with no part of speech and the name is the pronoun. The predicate takes a single argument, which is the identity of the possessed noun. *e.g.*, his dog → `his(x1) dog:NN(x1)`).

Relative adverbs (*i.e.*, *where, when, how, and why*) are binary predicates. The first argument refers to the head of main clause and the second to the head of the relative clause. For example,

When Mark went home, his family ate dinner.
`when(e1,e2) Mark:NN(x1) go:VB(e1,x1,x2) home:NN(x2) his(x3) family:NN(x3) eat:VB(e2,x3,x4) dinner:NN(x4).`

A.2 Transformation Rules

The Senseval data were used to evaluate the effectiveness of the transformation from a dependency parse tree into a semantic representation. An accurate syntactic knowledge extraction system was a logical step towards a semantic system. I started with a dependency parse tree from the Stanford parser (de Marneffe et al., 2006), then I created a logical representation by applying a small set of hand-coded transformations. The rules were developed by analyzing the development data, evaluating against the training set, intuition, and review of the reference documents (LF specifications from Rus's thesis and Senseval task, and the Stanford dependency parser manual). In section 7.1.1 these rules are evaluated against other known systems.

There are multiple phases in the transformation process. Figure A.1 lists the high-level steps, while sections A.2.1 to A.2.9 provide detailed descriptions of each step.

1. Create Predicates
2. Process Multi-Word Expressions
3. Process Conjunctions
4. Assign Fixed Identifiers
5. Create Noun-noun compounds
6. Assign Dependent Identifiers
7. Assign Parameters
8. Process Adverbial Clause Modifiers
9. Assign Prepositional Relations

Figure A.1: Dependency to LF Transformation Steps

To help understand the transformation process consider example 1 from the training data of the Senseval task. The variable names may differ between the gold standard LF and a system's LF,[2] but they are considered consistent if they connect the same predicates in the same way (*i.e.*, there exists a mapping between the variable names). Furthermore, the transformation presented contains one error (or inconsistency) the Stanford parser considered the word "for" as preposition acting as the mark for an adverbial clause modifier instead of as a conjunction, as the gold standard does. Thus instead of "for" being processed in step 3, it is processed in step 8. Figure A.2 is my visualization of the

[2]A system being evaluated uses Senseval data and evaluation scripts.

Stanford dependency parse tree.

Example 1 from the Senseval training data:

Alejandro played football, for Maria went shopping.

Gold Standard LF:

```
Alejandro:NN (x1) play:VB (e7, x1, x2) football:NN (x2) for(e2, e7, e8)
Maria:NN (x3) go:VB (e8, x3, x4) shopping:NN (x4) .
```

This system's LF:

```
Alejandro:NN (x1) play:VB (e1, x1, x2) football:NN (x2) for (e1, e2)
Maria:NN (x3) go:VB (e2, x3, x4) shopping:NN (x4)
```

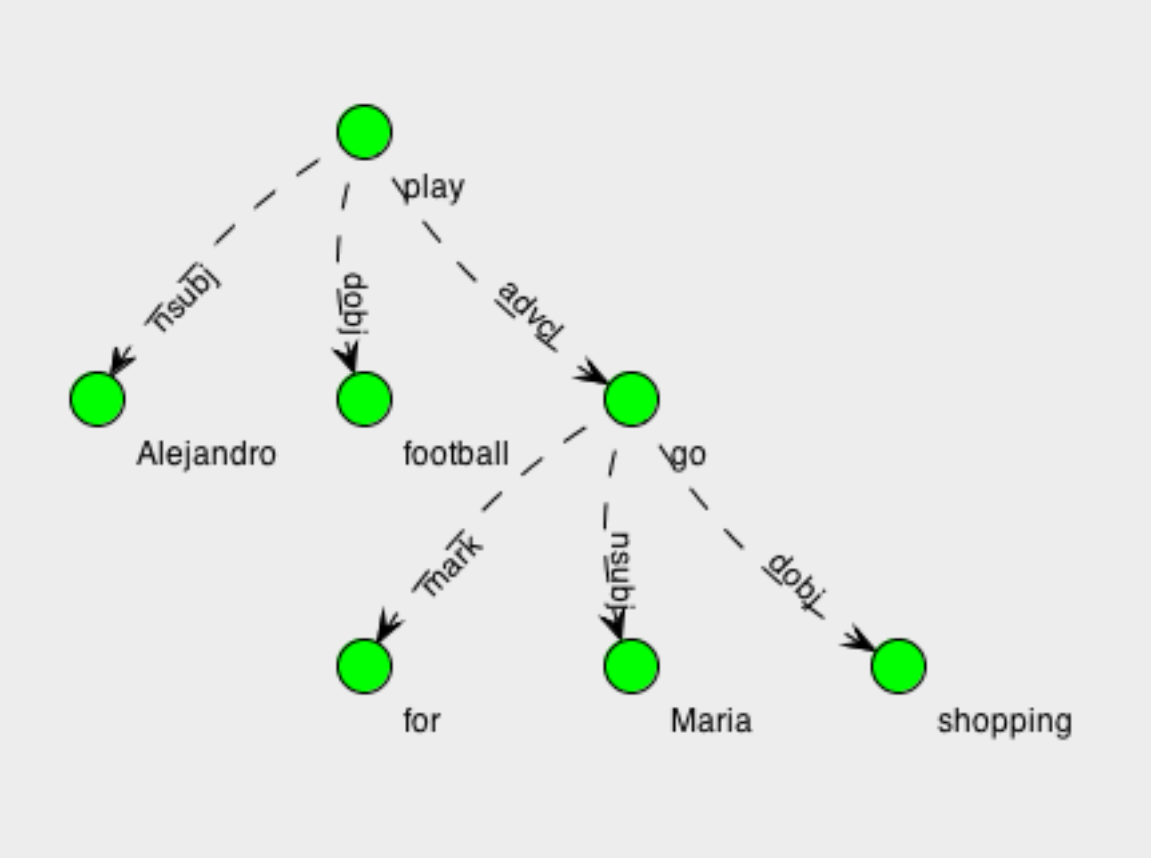


Figure A.2: Visualization of the dependency parse tree from the Stanford parser

A.2.1 Creating Predicates

A predicate is created for each pronoun, noun, verb, adverb, adjective, conjunction, and prepositions, with two exceptions: negations (*e.g.*, not), which are sometime marked as

adverbs by the parser, and auxiliary verbs are ignored. Negations and auxiliary verbs are the target of a "neg" or "aux" relation in the dependency tree. The predicates for verbs and nouns are the lemmatized word.

A.2.2 Processing Multi-Word Expressions

WordNet and most modern dictionaries have entries for some collocations and multi-word expressions. These usually have a meaning that is more than the sum of their parts. In particular they often refer to a concept more specific than the words would suggest. Sometimes these expressions have a meaning not easily inferred from the parts.

I implemented the process identifying multi-word expressions as a left to right process, of trying all n-grams of 5 or less words to see if they are present in *WordNet*. If more than one exists, then the longest is selected. The nodes in the dependency tree that compose the multi-word expression are removed and replaced with a single node. The type of this new node is the complete multi-word expression. When this node is converted into a predicate, all of the spaces between words become underscores. For example, *compound lens* becomes `compound_lens:NN(x1)`.

A.2.3 Processing Conjunctions

Conjunctions come in three forms: coordinating, adverbial, and subordinating. This section handles coordinating and subordinating conjunctions, which can be between clauses, phrases, or words. Each of the conjunctions listed below takes at least two *values* (the identities of other predicates, which are the head of their respective syntactic structures). Each conjunction has its own identity, which represents a new set or value composed using some operation (*i.e.*, logical or mathematical). Clauses may also be connected by adverbial clause modifiers described and processed in section A.2.8. A partial list[3] of conjunctions is: and, or, but, nor, either, neither, plus, minus.

The Stanford dependency parser represents conjunctions with a "conj" dependency. A predicate representing the relation is created with a fresh variable for its identity, which represents the conjoined value, operation, or set. The source and destination of the relation provide the values for subsequent parameters. When more than two values (words, phrases, clauses) are conjoined, the Stanford parser creates a chain of "conj" relations, thus after creating the initial predicate, the rest of the chain should be followed

[3]The complete list is provided by the Stanford parser documentation.

with each new value being added as a parameter to the predicate.

For example consider the phrase "the cat and the dog". Figure A.3 contains a graphical representation of the dependency parse from the Stanford parser. Figure A.4 contains a transformed version of the graph, which can either then be processed into an LF or a Conceptual Graph (CG). And the LF representation of the phrase is: `and(x1,x2,x3) cat:NN(x2) dog:NN(x3)`

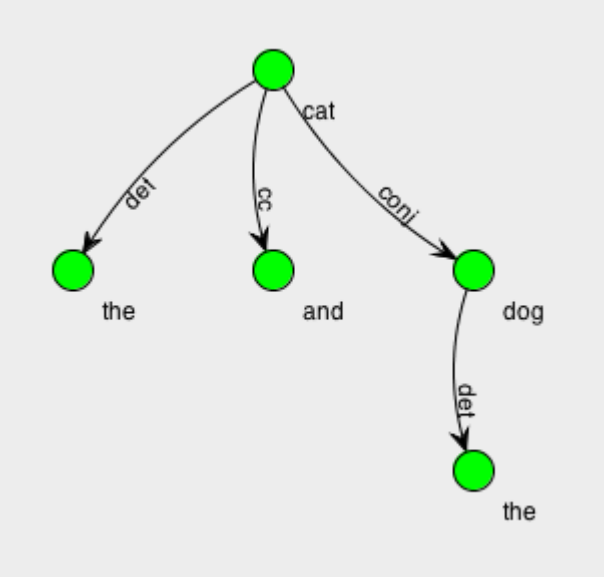


Figure A.3: Dependency parse tree of conjunction

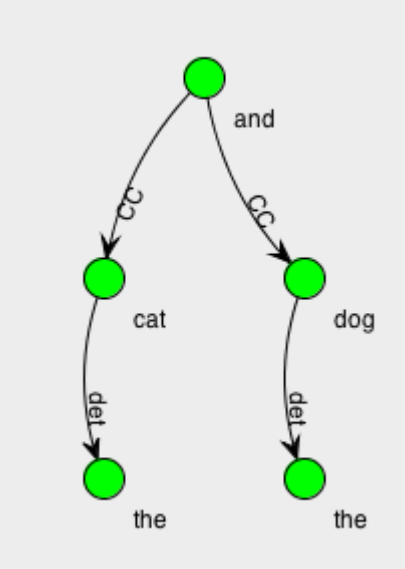


Figure A.4: Post-processed graph of conjunction

A.2.4 Assigning Fixed Identifiers

Most nouns and verbs represent entities or events; each of these is assigned an identity variable (the first argument). The identity variable is used to reference the concept and to uniquely identify it. Since each verb and noun is its own identity, they are assigned fresh variables at this stage. Modal and auxiliary verbs are ignored as they do not define new concepts or events. The part of speech (POS) tags from the parser are effective for identifying nouns and verbs (NN* or VB*). Modal verbs use an MD POS tag, while both modal and auxiliary verbs can be identified by the "`aux`" relation, which connects them to the head verb.

Consider the following LF for the sentence "The big man ran home for dinner." The arguments in bold are fresh (new or unused) identifiers assigned during this step:

```
big:JJ(x1) man:NN(x1) ran:VB(e1, x1, x2) home:NN(x2)
for:IN(e1, x3) dinner:NN(x2)
```

A.2.5 Noun-Noun compounds

Noun-noun compounds which are not in *WordNet* are treated differently than multi-word expressions. If a concept was in the dictionary, it would have been considered a multi-word expression. Noun-noun compounds do not refer to a single concept that satisfies both nouns (*i.e.*, the compound is not a subtype of both nouns), instead both nouns are somehow related by an unmentioned relation, thus a representation such as $noun_1(x_1)$ $noun_2(x_1)$ would not be appropriate. For example, consider "chocolate box". Both words are nouns, and the meaning (sense) does not refer to something that is both a box and chocolate, but refers to a box containing chocolates. Determining the relations between noun-noun compounds is a difficult task (Girju et al., 2007; Hendrickx et al., 2010; Butnariu et al., 2010), thus LF simplifies the concept representation; a new *nn* predicate represents the compound concept, with it's own identity but with arguments associating the original nouns (eg. $nn(x_1, x_2, x_3)$ $noun_1(x_2)$ $noun_2(x_3)$

The Stanford parser will create an "nn" relation between the nouns in such compounds. I created an nn predicate for any "nn" relation where the first argument is a fresh variable and the second and third are the source and destination of the dependency relation.

A.2.6 Assigning Dependent Identifiers

Adjective, adverb, and possessive pronoun predicates do not have their own identity variables; they reuse the variable of the word they modify. We apply an ordered set of rules to identify the modified word; the first rule that matches is applied. If the modified word also has a dependent identifier (*e.g.*, an adverb or an adjective) that has not been resolved, then it is also resolved.

As mentioned earlier, the rules were developed by analyzing the development data, evaluating against the training set, intuition, and review of the reference documents for the competition. The basic intuition for each of adjective rules is based on the situation they are meant to handle and the relation they refer to. Each rule will be followed by a brief situation explanation.

Rules for selecting adjective identifier:

source of "amod" relation - This relation connects an adjective to a noun.

target of "nsubj" relation - See Exception 1 below.

target of "csubj" relation - See Exception 1; this is for clausal subjects.

source of "acomp" relation - This relation connects a noun and an adjective clause.

source of "dep" relation - If no word is found to be modified, find the source of an unlabeled dependency.

Identifying adverb identifier rules:

source of "advmod" relation - This relation connects an adverb to the verb being modified.

source of "dep" relation - If no word is found to be modified, find the source of an unlabeled dependency.

Identifying possessive pronoun identifier rule:

source of "poss" relation - this relation connects the possession to the possessor

Exception 1 - Adjective with subject This example is taken from the development data of the Senseval task. When an adjective is connected to a noun through a verb, such as *gets* (below) or *will be*, the relation assigned is "nsubj" with a reversed direction, instead of "amod". The example and gold LF from the development data:
If I am late for work my boss gets angry.

```
If(e6, e5) I(x3) be:VB(e5, x3, x1) late:JJ(x3) for(x3, x1) work:NN(x1)
my(x2) boss:NN(x2) get:VB(e6, x2) angry:JJ(x2) .
```

Figure A.5 illustrates the Stanford dependency parse tree for this sentence.

A.2.7 Assigning Parameters

To understand the meaning of a word requires an understanding of how other words are related to it. To this end, verb and comparative predicates take extra arguments connecting them to the words/predicates required for understanding. This step assigns the extra parameters to verbs and comparatives. Since there are multiple arguments

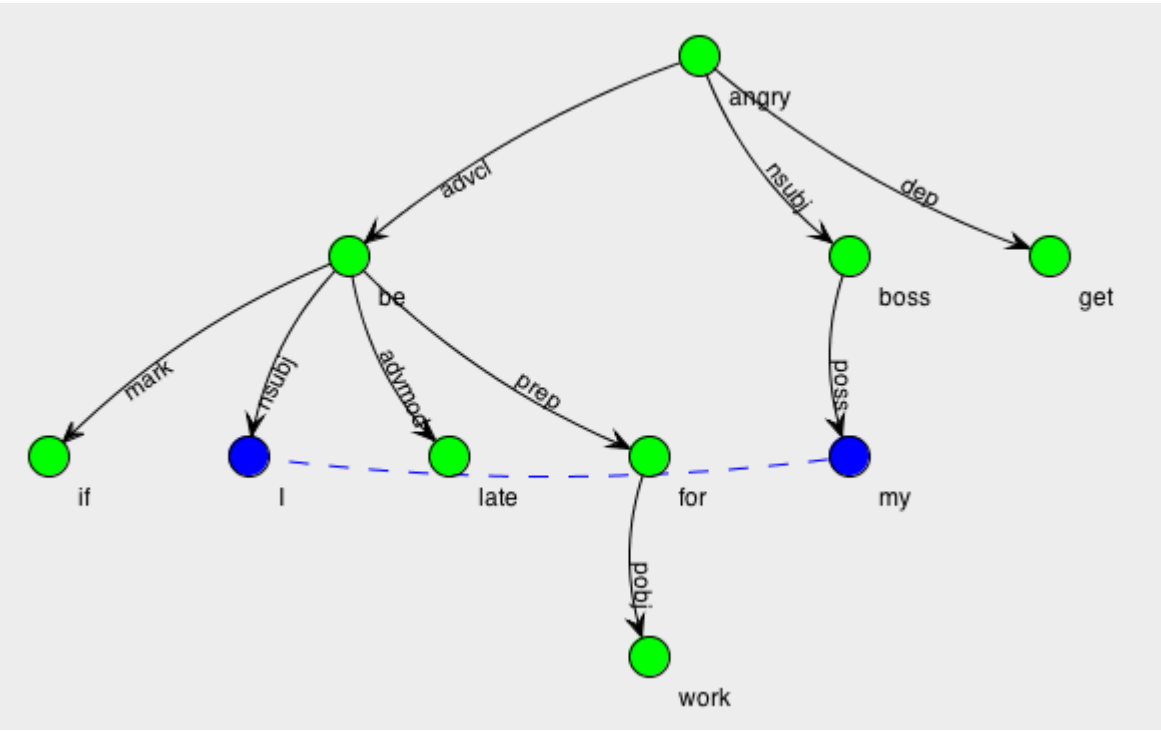


Figure A.5: Example Stanford parse tree of an adjective with a subject

to be identified, there are different lists of rules for each argument. With each list, as before, the result of the first matching rule is used.

Verbs usually have the following arguments in addition to their identity/eventuality: subject, object, and optionally indirect object.

Rules for identifying the subject:

target of "`nsubj`" relation - This relation connects a verb to its subject.

target of "`xsubj`" relation - This relation connects a verb in an open clausal complement to its external (controlling) subject.

source of "`cop`" relation - This relation connects a copula to the subject of the head verb.

target of "`nsubj`" relation from source of "`xcomp`" relation - This explicitly implements the "`xsubj`" rule.

create a fresh variable - This assumes the verb had no subject, but the argument must be filled.

Rules for identifying the direct object:

target of "`dobj`" relation - This relation connects a verb to its direct object.

target of "`nsubjpass`" relation - This relation connects a passive verb to its syntactic subject.

target of "`xcomp`" relation - This relation connects a verb to the head of an open clausal complement, which acts as (or fills the slot of) the direct object in LF.

create a fresh variable - This assumes that no direct object was provided, but the argument must be filled.

Rule for identifying the indirect object:

target of "`iobj`" relation - This relation connects a verb to its indirect object

Comparatives are a type of adjective, thus their identifier is inherited from the word they modify. However, comparatives also have an additional argument the comparison word. To identify the second argument, I have only one rule:

- Target of prepositional relation "`than`"

Some verbs expect verbs as (or to fill the slot of) direct objects; these direct objects tend to be verbs in the infinitive form or as gerunds[4]. The *to* used to denote a verb in the infinitive also acts to connect the two verbs in LF. It takes two parameters; first is the verb expecting a verb as an object and the is second the verb in the infinitive (object). The rules for finding these parameters are as follows:

Rule for identifying first parameter of *to*:

first source of any relation to source of "`aux`" relation

Rule for identifying second parameter of *to*:

source of "`aux`" relation

A.2.8 Processing Adverbial Clause Modifiers

Adverbial clauses modify the verb of another clause. The Stanford parser creates an "`advcl`" relation between the head verbs of both clauses. Thus when a "`advcl`" relation

[4]a derived form of a verb that acts a noun; gerunds are verbs ending with -ing.

is detected, a predicate is created for it. The predicate name is the word that caused the relationship; this word is usually connected to the dependent word of the "`advcl`" relation by either, an "`advmod`" relation and has a `WRB` POS, or a "`mark`" relation.

The identifier of the source of the "`advcl`" relation is assigned as the first parameter of the predicate, while the identity of the target is the second parameter. Both words should be verbs since they are the heads of a clause.

Consider the following example from the Stanford typed dependency manual: *The accident happened as the night was falling.* The second clause *night was falling* modifies the head verb of the first clause *happened*. Figure A.6 illustrates the parse tree of the example sentence.

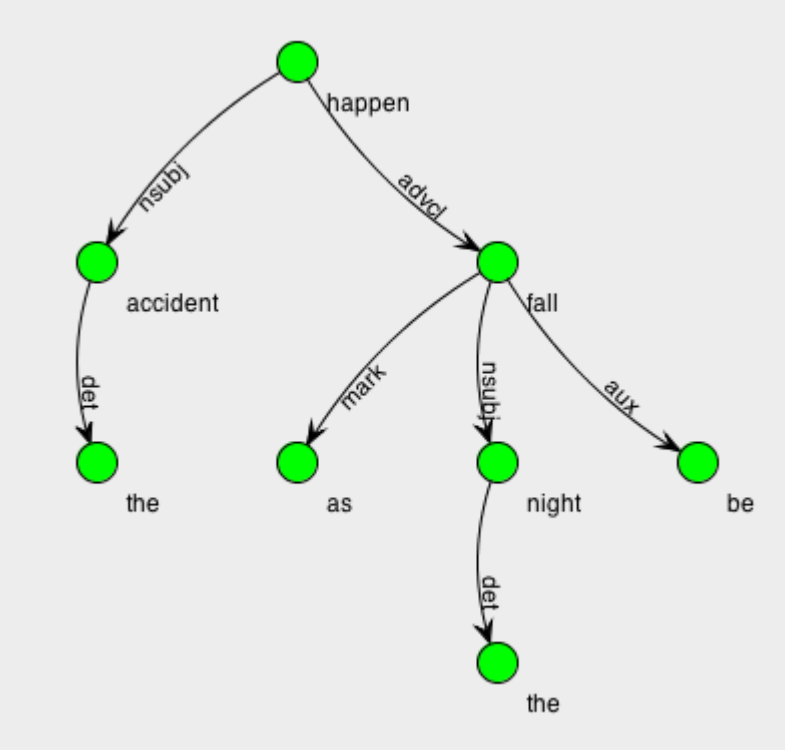


Figure A.6: Example Stanford parse tree with an adverbial modifier clause

A.2.9 Assigning Prepositional Relations

Dependency trees often collapse prepositions giving them no concept instead treating them as only a relation between two words. Uncollapsed representations give prepositions their own concepts with generic relations connecting it to the other concepts. In the

Stanford parser these relations are often "prep" and "pobj", connecting the source and target of the preposition, respectively.

While rare, there are a number of examples of sentences that when collapsed, lose a significant part of their meaning or proper structure. These difficulties tend to involve conjunctions and prepositional attachments. Consider the sentence fragment: *organisms live at or near the bottom of a sea*. The conjunction affects the prepositions, which cannot be represented if they are collapsed into relations. When the Stanford parser collapses the prepositions and their "prep" and "pobj" relations, all other relations attached to the preposition must be moved to either the source or target of the preposition. The conjunction is moved and this leads to difficulties in interpretation. The uncollapsed representation is more accurate, the collapsed representation is more compact, thus I use the uncollapsed representation and automatically collapse those relations where there is no loss of meaning.

Figures A.7 and A.8 illustrate the collapsed and uncollapsed parse trees for the sentence *Organisms live at or near the bottom of a sea*.

To create the predicates for prepositions, all collapsed prepositional relations and all uncollapsed prepositional concepts must be processed. First, a predicate with two arguments is created for the preposition, with the name being that of the preposition. The source and target of each preposition become the first and the second arguments, respectively. If the source is a verb, then the target must be added to the verb argument list.

A.3 Logic Form Criticisms

Implementing a functionality similar to the one used by Rus (2002) and by the Senseval 2004 task logic form identification was done quickly using a modern parser. While Anthony and Patrick (2004), Bayer et al. (2004), Ahn et al. (2004), and Wenner (2007) all completed this task with reasonable success, some using a dependency parser, my results are slightly higher than theirs. Furthermore, the success of the LF evaluation suggests that this is a good starting point for a semantic transformation.

The LF representation does not tag relations with semantic roles (or theta roles). Instead, the relations are syntactic or prepositions. While Rus showed that this representation can be effective, it is still limited in its ability to compare meanings that have different syntactic realizations.

There are a number of subtle representation issues that are questionable in the Logic

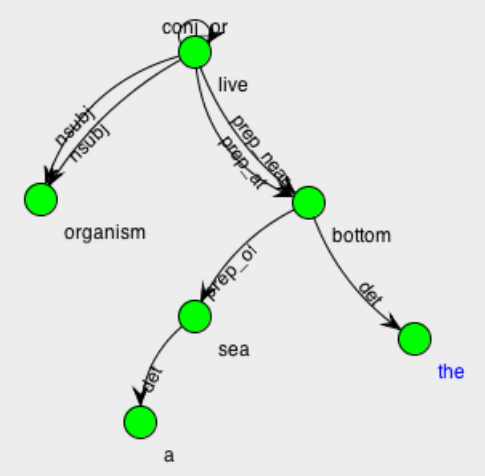


Figure A.7: Collapsed Dependency Parse Tree

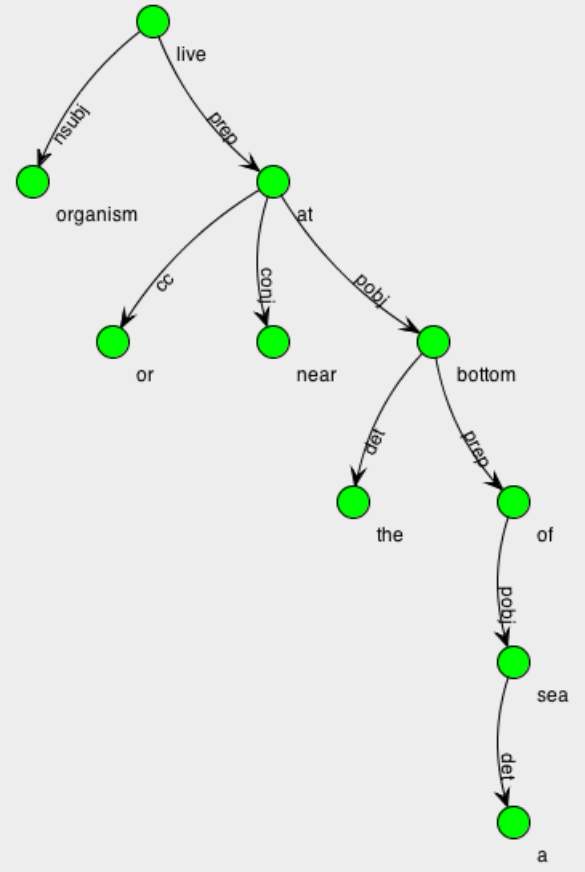


Figure A.8: Uncollapsed Dependency Parse Tree

Form Identification (LFi) data. Obtaining high results on the LFi task requires handling each of these cases correctly, though many of these cases are inconsistent with what we expect from a semantic representation. The following are a number of examples and criticisms of the LF representation used by Rus and SensEval LFi task.

Consider Example 1, the inclusion of "to (e4, e5)" is unnecessary: "to" marks the infinitive of the verb "study", moreover the relation is already defined between the verbs as "study" is the object of "like". It also seems strange to include "morning" as the direct object (third argument) of "study", in addition to connecting "study" through the word "to"; this treatment is consistent with "to" being a preposition, though here it is as an infinitive marker. Normally, we would expect the direct object of "study" to be the topic of study, not the time. Perhaps, this example comes from a slightly different (newer or older) specification of LF: one that does not require verbs to have the direct object, and the prepositional phrase is directly appended after the subject.

Example 1. Some students like to study in the mornings.

```
student:NN (x1) like:VB (e4, x1, e5) to (e4, e5) study:VB (e5, x1, x2)
in (e5, x2) morning:NN (x2) .
```

In Example 2, we see that `funny:JJ (x1)` is associated with Fred, which is debatable but acceptable, while `felt:VB (e2, x1)` has no association with "funny" (it should probably fill the third argument of felt, which is normally reserved for the direct object). Because "funny" does not have a unique identifier, as it is shared with "Fred", it cannot be associated with "felt" without also causing Fred to also share this association. This example leads me to believe that a better representation should allow "funny" to have its own identifier so that it can be associated with "felt".

Example 2. Fred felt funny.

```
Fred:NN (x1) felt:VB (e2, x1) funny:JJ (x1) .
```

If we consider another sentence like "The cake smells good", the LF is `cake:NN (x1) smell:VB (e2, x1) good:JJ (x1).` We would not want good and cake directly associated as it currently is, because the *goodness* of the cake is unknown but the smell of the cake is known to be good.

In Example 3, the function of the verb `be:VB (e8, x1, x2)` is of questionable value in the LF representation; a more concise and simpler representation would be to remove `be:VB (e8, x1, x2)` and simply relabel `on (e8, x2)` to `on (x1, x2)`, directly connecting the student to their location (*i.e.*, `student:NN (x1) on (x1, x2) bus:NN (x2)`).

Example 3. The students who are on the bus to the United States are studying English.

```
student:NN (x1) be:VB (e8, x1, x2) on (e8, x2) bus:NN (x2) to (x2, x4)
United_States:NN (x4) study:VB (e9, x1, x6) English:NN (x6) .
```

While these are only a sample of issues with the LF representation, there are many more that could be called into question. LF still provides a gold standard representation, which more expressive knowledge representations can be mapped onto, and compared against.

Appendix B

Transformations

B.1 Linear Graph Notation

The transformation system uses a special linearized representation of a graph, which was derived from the linear form of Conceptual Graphs (CGs). Concepts are enclosed in square brackets `[concept]`, relations in parentheses `(relation)`. Concepts and relations in this linear representation can specify a *type*, multiple *names*, and *property/value pairs*; each of these is optional and can be used as a pattern requirement or as a value to be set in a transformation.

A type can be a simple alphanumeric string or contained within quotes; the use of quotes allows for non-alphanumeric characters like spaces or colons.

Transformations are composed of two graphs: a pattern graph and a transformation graph. The pattern graph must be found as a subgraph in the input graph. A subgraph matching a pattern graph functions as a check to enable the transformation and aligns the transformation in another graph. The transformation graph defines what the sub graph which matches the pattern, should be like after transformation. The transformation system automatically determines how to transform the matching subgraph into the transformation graph.

Names start with a colon followed by an alphanumeric string (*e.g.*, `:name`). Names enable nodes to be identified and related between a pattern graph and a transformation graph. Names function as a reference. Labeling nodes with names allows a node to be identified so that during transformation it may be kept, moved, or annotated.

The position of labelled concepts may be the destination for a moved concept. Destinations are only used in transformation graphs and are declared with an `@` followed by

a name. If the destination concept is not also moved, then it will be overwritten.

Nodes (concepts and relations) can be converted between forms (concept or relation), though a restriction exists. When converting a concept to a relation, there must be at most one incoming relation and one outgoing relation. This type of conversion is usually used to compress preposition concepts into relations.

Concepts and relations may have annotations (properties) that are required or set. These take the form `?PropertyName=PropertyValue`, where both property name and value must be alphanumeric strings. The value section of the property (`=PropertyValue`) is optional. If the value is not specified then the system only tests for the existence of the property or sets the property with a default value.

B.2 Transformations

Transformations are applied once a match with the pattern has been found. The system converts the sub-graph matching the pattern to a sub-graph matching the result (transformation) graph. Concepts that are in the pattern but not in the result are removed. Concepts only in the result are added. Concepts may be moved and properties added. The labels are used to align concepts between the pattern and the result. Two concepts or relations with the same label should be the same; all the properties are copied or moved, or the same node is used. Concepts may be converted into relations and vice versa (when they have the same labels). Concepts may also be repositioned as in the example below.

Most transformations are used to normalize graphs or annotate specific patterns. The transformation system has a library of transformations indexed by keyword. For example the keyword *adjPattern* will load all the transformations, which are used to normalize and annotate definition of adjectives with are context patterns. Below is one example of a transformation.

Example
template: VGN Amod Pattern
keywords: adjPattern
pattern: `[:mod]<-(nsubj)<-[:head:adj?POS=VGN]->(cop)->[be]`
transform:
`[:mod@head?ModifiedByAdj?PatternHead]->(amod)->[:head?Modifier]`

The transformation consists of two parts: the pattern graph and transformation (or resulting) graph. The pattern is a graph, which must be found and matched; it is expressed in the linear form described above.

In the example above, the pattern requires a simple structure with a verb, a subject, and a copula. The verb is required to have the `POS` property of being a `VGN`. The transformation moves the node labelled *:mod* to the position of the *:head* node, while the *:head* node is moved to the destination of an *amod* relation. Furthermore, a number of properties are set for each of the nodes. Lastly, the copula node and relation from the pattern will be removed because they do not exist in the transformation graph.

Figure B.1 illustrates the pattern graph and transform graph above. An example of input graph is provided as well as the transformed output.

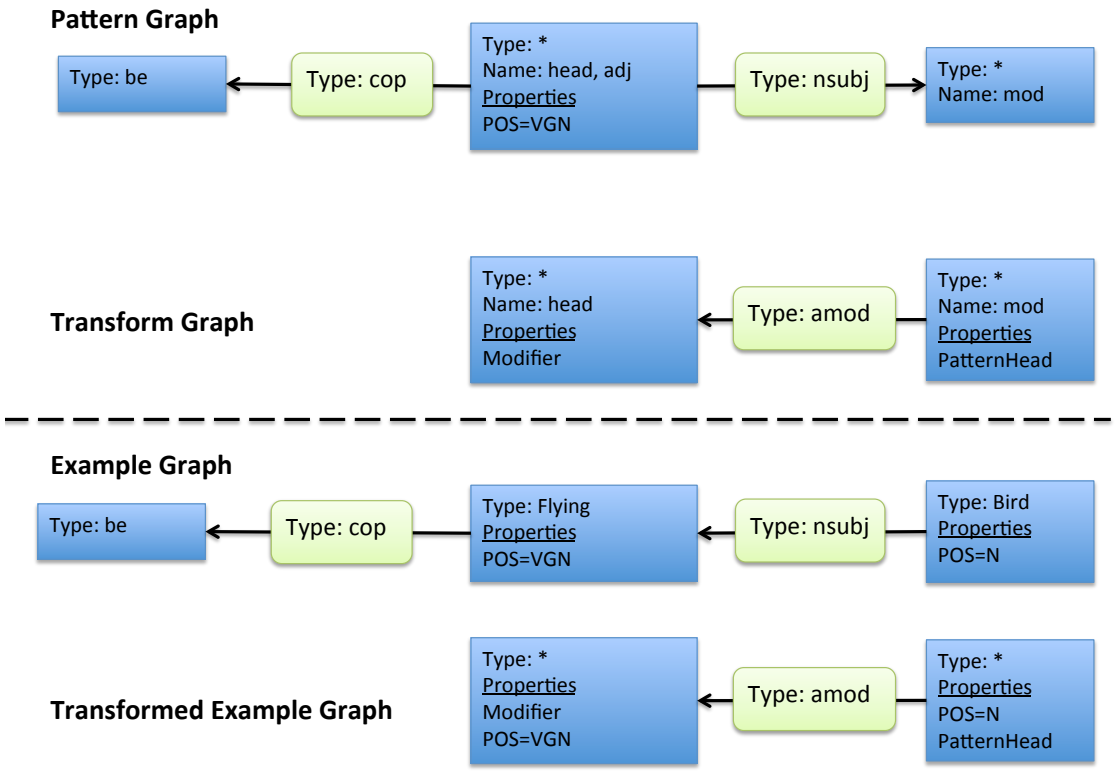


Figure B.1: Example of a transformation

Context Pattern Transforms

We use a number of transformations to label and identify content patterns. Below I list all the patterns used for adjective context patterns; many of the patterns are slight variations of one another to account for slightly different structures that result from the parsing process. *If-then* context patterns are detected programmatically by identifying two clauses, with an *if* mark on clause head. This same clause should also contains the word being defined (it appears to be a recursive definition).

PatternHead property is added to the head of pattern subtree of the definition, while the *DefinitionHead* property is added programmatically by finding the second clause or verb phrase of the current clause. The properties *ModifiedByAdj* and *Modifier* are added to indicate headword being modified and modifier word, respectively. The node in the pattern with the label *:adj* is recursive adjective being defined. Before matching is done, comparison of pattern to a particular subtree, the pattern node is aligned with the node in the parsed tree.

```
template: Basic Amod Pattern
keywords: adjPattern
pattern: [:adj]<-(amod:amod)<-[:head]
transform: [:adj?Modifier]<-(amod:amod)<-
    [:head?ModifiedByAdj?PatternHead]

template: Is Amod Pattern
keywords: adjPatternAlt
pattern: [:mod]<-(nsubj:n)<-[be:head]->(advmod:a)->[:adj]
transform: [:mod?ModifiedByAdj]<-(nsubj:n)<-[be:head?PatternHead]
    ->(advmod:a)->[:adj?Modifier]

template: Is Amod Pattern Mapping
keywords: adjPattern
pattern: [:mod]<-(nsubj:n)<-[be:head]->(advmod:a)->[:adj]
transform: [:mod@head?ModifiedByAdj?PatternHead]
    ->(amod)->[:adj?Modifier]

template: VGN Amod Pattern
keywords: adjPattern
pattern: [:mod]<-(nsubj)<-[:head:adj?POS=VGN]->(cop)->[be]
```

```
transform: [:mod@head?ModifiedByAdj?PatternHead]
    ->(amod)->[:head?Modifier]

template: VGN Amod Pattern
keywords: adjPattern
pattern: [:mod]<-(nsubj)<-[:head:adj?POS=VGN]->(aux)->[be]
transform: [:mod@head?ModifiedByAdj?PatternHead]
    ->(amod)->[:head?Modifier]

template: VGN Amod Pattern
keywords: adjPattern
pattern: [:mod]<-(nsubjpass)<-[:head:adj?POS=VGN]->(auxpass)->[be]
transform: [:mod@head?ModifiedByAdj?PatternHead]
    ->(amod)->[:head?Modifier]

template: JJ as head of Amod Pattern
keywords: adjPattern
pattern: [:mod]<-(nsubj)<-[:head:adj?POS=JJ]->()->[be]
transform: [:mod@head?ModifiedByAdj?PatternHead]
    ->(amod)->[:head?Modifier]
```

Property Transforms

The following patterns are used to identify definitions that define properties of concepts, such as price or weight. This is a complete list of all Property transformations used in this research. The concept, which has the property being defined, is annotated with *PropertyObject*. For example *object* would be PropertyObject of *weight*, or *purchase* would be PropertyObject of *price*. The type of property, that is the type of the value of the property, is assign *PropertyType*.

```
template: attribute How on Amod
keywords: properties
pattern: [how:how]<-(advmod)<-[:propertyType?SPOS=adjective]
    <-(amod)<-[:object]<-(nsubj)<-[be:root?Root]
transform: [:propertyType?PropertyType]<-(amod)
    <-[:object?PropertyObject]<-(nsubj)<-[be:root?Root]

template: attribute How on Amod with Conjunction
```

```
keywords: properties
pattern: [how:how]<-(advmod)<-[:propertyType?SPOS=adjective]
    <-(amod)<-[]<-(CC)<-[:object]<-(nsubj)<-[be:root?Root]
transform: [:propertyType?PropertyType]<-(amod)<-
    [:object?PropertyObject]<-(nsubj)<-[be:root?Root]

template: attribute How on be
keywords: properties
pattern: [:propertyType?SPOS=adjective]<-(amod)<-
    [:object?PropertyObject]<-(nsubj)<-[be:root?Root]->
    (advmod)->[how:how]
transform: [:propertyType?PropertyType]<-(amod)<-
    [:object?PropertyObject]<-(nsubj)<-[be:root?Root]

template: attribute How on be with conjunction
keywords: properties
pattern: [:propertyType?SPOS=adjective]<-(amod)<-[]
    <-(CC)<-[:object]
    <-(nsubj)<-[be:root?Root]->(advmod)->[how:how]
transform: [:propertyType?PropertyType]<-(amod)
    <-[:object?PropertyObject]
    <-(nsubj)<-[be:root?Root]

template: attribute How detached JJ
keywords: properties
pattern: [how:how]<-(advmod)
    <-[:propertyType?SPOS=adjective]<-(dep)
    <-[be:root?Root]->(nsubj)->[:object]
transform: [:propertyType?PropertyType]<-(amod)
    <-[:object?PropertyObject]
    <-(nsubj)<-[be:root?Root]

template: amount of
keywords: properties
pattern: [:propertyType?SPOS=noun]<-(of)
    <-[amount:root?Root]
transform:[:propertyType?SPOS=noun?PropertyType]
```

```
    <-(of)<-[amount:root?Root]
```

Query Transforms

The following patterns are used to transform queries into a for that can be searched for in text or knowledge base.

```
template: Remove is - target NER
keywords: query
pattern: [who:query]<-(:attr)<-[be]->(nsubj)->[:connected?NER]
transform: [:connected]<-(nsubj)
    <-[person?SPOS=noun?QA.QueryNode=WhoDescription]

template: Remove is - target NER
keywords: query
pattern: [who:query]<-(:attr)<-[be]->(nsubj)->[:connected?POS=NNP]
transform: [:connected]<-(nsubj)
    <-[person?SPOS=noun?QA.QueryNode=WhoDescription]

template: Remove is - target NER
keywords: query
pattern: [who:query]<-(:attr)<-[be]->(nsubj)
    ->[:connected?POS=NNPS]
transform: [:connected]<-(nsubj)
    <-[person?SPOS=noun?QA.QueryNode=WhoDescription]

template: Remove is - Target NOUN
keywords: query
pattern: [who:query]<-(:attr)<-[be]->(nsubj)
    ->[:connected?SPOS=noun]
transform: [:connected?QA.QueryNode=WhoNER]

template: Remove is - Target NOUN related to verb
keywords: query
pattern: [who:query]<-(:rel)<-[:verb?SPOS=verb]
transform: [who:query]<-(:rel)
    <-[:verb?SPOS=verb?com.scaiano.QA.QueryNode=WhoNER]
```

```
template: Remove do node and replace with query.
keywords: queryZZZ
pattern: [what]<-(dobj)<-[do:query?ClauseHead=true]
    -{->(aux)->[do];->(advcl)->[:clause?ClauseHead=true]}
transform: [:clause?com.scaiano.QA.QueryNode=WhatDo]

template: Remove do node and replace with query.
keywords: query
pattern: [what]<-(dobj)<-[do:query?ClauseHead=true]->(aux)->[do]
transform: [WhatDo:query?com.scaiano.QA.QueryNode=WhatDo]

template: Process Where did
keywords: query
pattern: [where:query]<-(:rel)<-[:verb]->()->[do]
transform: [:verb?com.scaiano.QA.QueryNode=Where]

template: Process Where is
keywords: query
pattern: [where:query]<-(attr)<-[be]->(nsubj)->[:connected]
transform: [:connected?com.scaiano.QA.QueryNode=Where]

template: Process Where generic
keywords: query
pattern: [where:query]<-(:rel)<-[:verb]
transform: [:verb?com.scaiano.QA.QueryNode=Where]

template: Process Where did
keywords: queryZZZ
pattern: [where:query]<-(:rel)<-[:verb]->()->[do]
transform: [:query?com.scaiano.QA.QueryNode=Where]
   <-(dep:rel)<-[:verb]

template: Process Where is
keywords: queryZZZ
pattern: [where:query]<-(attr)<-[be]->(nsubj)->[:connected]
transform: [:query?com.scaiano.QA.QueryNode=Where]
    ->(dep)->[:connected]
```

```
template: Process Where generic
keywords: queryZZZ
pattern: [where:query]<-(:rel)<-[:verb]
transform: [:query?com.scaiano.QA.QueryNode=Where]<-
    (dep:rel)<-[:verb]

template: Process When did
keywords: queryZZZ
pattern: [when:query]<-(:rel)<-[:verb]->()->[do]
transform: [:query?com.scaiano.QA.QueryNode=When]
    <-(dep:rel)<-[:verb]

template: Process When was
keywords: queryZZZ
pattern: [when:query]<-(:rel)<-[:verb]->()->[be]
transform: [:query?com.scaiano.QA.QueryNode=When]
   <-(dep:rel)<-[:verb]

template: When generic
keywords: queryZZZ
pattern: [when:query]<-(:rel)<-[:verb]
transform: [:verb?com.scaiano.QA.QueryNode=When]

template: Process When did
keywords: query
pattern: [when:query]<-(:rel)<-[:verb]->()->[do]
transform: [:verb?com.scaiano.QA.QueryNode=When]

template: Process When was
keywords: query
pattern: [when:query]<-(:rel)<-[:verb]->()->[be]
transform: [:verb?com.scaiano.QA.QueryNode=When]

template: When generic
keywords: query
pattern: [when:query]<-(:rel)<-[:verb]
```

```
transform: [:verb?com.scaiano.QA.QueryNode=When]

template: Which generic
keywords: queryZZZ
pattern: [:type]<-(nsubj)<-[:verb]
transform: [:type?com.scaiano.QA.QueryNode=WhichInstances]
    <-(nsubj)<-[:verb]

template: Process What is Attribute
keywords: query
pattern: [what:query]<-(attr)<-[be]->(nsubj)->[:connected]
transform: [:query?com.scaiano.QA.QueryNode=WhatIs]
   ->(dep)->[:connected]

template: Process What verb
keywords: query
pattern: [what:query]<-(:dep)<-[:connected]
transform: [:query?com.scaiano.QA.QueryNode=WhatIs]
   <-(:dep)<-[:connected]

template: Process What is name
keywords: query
pattern: [what:query]<-(:dep)<-[name:connected]
transform: [:query?com.scaiano.QA.QueryNode=WhatName]
   <-(:dep)<-[:connected]

template: Process What is called
keywords: query
pattern: [what:query]<-(:dep)<-[call:connected]
transform: [:query?com.scaiano.QA.QueryNode=WhatName]
   <-(:dep)<-[:connected]

template: Process Why did
keywords: query
pattern: [why:query]<-(:rel)<-[:verb]->()->[do]
transform: [:query?com.scaiano.QA.QueryNode=Other]
   <-(:rel)<-[:verb]
```

```
template: Process Why is
keywords: query
pattern: [why:query]<-(attr)<-[be]->(nsubj)->[:connected]
transform: [:query?com.scaiano.QA.QueryNode=Other]
   ->(dep)->[:connected]
```

Appendix C

Knowledge Base Implementation

The knowledge, first described in Section 6.5, is implemented in MYSQL and designed for rapid search and lookup. The underlying details that make the implementation fast for searching are described here.

Any sub-graph can be used as a search query and all matching graphs are quickly returned. The search key may include concept or relation types (or they may be wildcards, matching any type) and may require specific properties, as specified by the requester. Two different attempts were made at implementing database searches: the use of `INNER JOIN` was found to be several times faster, in realistic circumstances, than the use of `SUBSELECT`. It can be difficult to quantify the search speed as it depends highly on the query and data, though during testing `SUBSELECT` searches range in time from 90 to 600 seconds, while `INNER JOIN` searches generally required 20 milliseconds to 20 seconds. Two data sets were searched, one containing 50 000 graphs and another containing 1 000 000 graphs.

Figure C.1 depicts the structure of the database, with regard to graphs. Four tables are used: Concept, Relation, Property, and Graph. The relationships between tables are shown as arcs in the figure. The relationships are made through shared ids. Each table has simple indexes for each "informative" field: graph, id, source, target, type, value.

The SQL queries take the form presented below. The queries are quite long but simple to generate programmatically. The (...)* indicates that zero or more of this type of sub-expressions. The blue terms (*e.g.*, INNER_JOIN_ON_CONCEPTS) represent sub-expressions for a particular type of restriction or part of a search.

```
SELECT Graph.id FROM Graph g
    (INNER_JOIN_ON_CONCEPTS)*
```

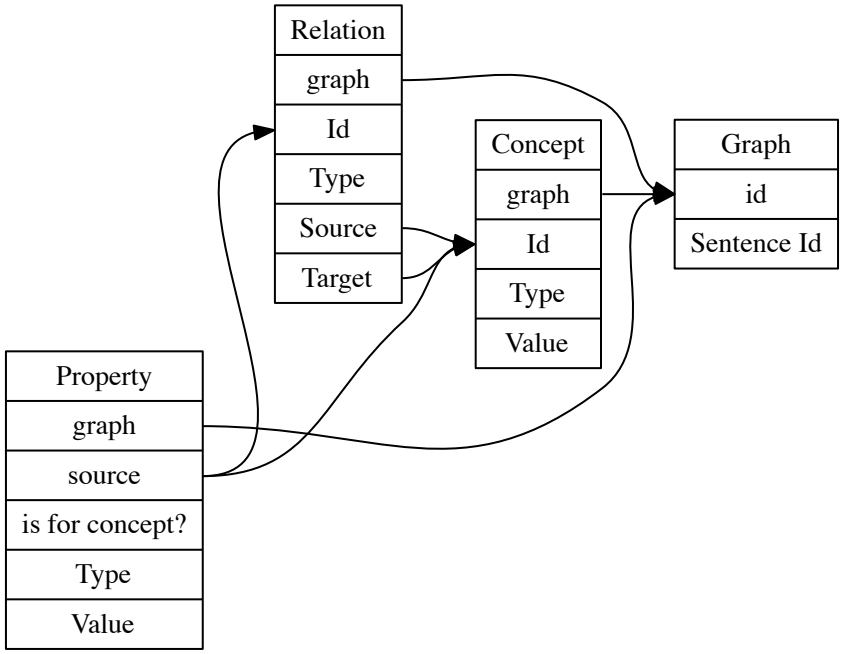


Figure C.1: Structure of the graphs in the database

```
    (INNER_JOIN_ON_RELATIONS)*
WHERE
    (CONCEPT_TYPE_RESTRICTION)*
    (RELATION_TYPE_RESTRICTION)*
    (RELATION_SOURCE_RESTRICTION)*
    (RELATION_TARGET_RESTRICTION)*
```

Each concept or relation (or property, though not represented here) in the query graph produces an `INNER JOIN` expression and possibly one or more `RESTRICTION` expressions. Each concept and relation is given an identifier within the query so that concepts and relations may be correctly connected. Below, I define the subexpressions; in them, concept_id and relation_id represent the unique ids generated for this concept or relation within the query. TYPE represents the type restriction of the concept or relation; if the concepts or relations are anonymous (no type) then no type restriction expression is required.

INNER_JOIN_ON_CONCEPTS →
INNER JOIN Concept concept_id ON g.graph = concept_id.graph

INNER_JOIN_ON_RELATIONS →
INNER JOIN Relation relation_id ON g.graph = relation_id.graph

CONCEPT_TYPE_RESTRICTION →
concept_id.type = TYPE

RELATION_TYPE_RESTRICTION →
relation_id.type = TYPE

RELATION_SOURCE_RESTRICTION →
relation_id.source = concept_id

RELATION_TARGET_RESTRICTION →
relation_id.target = concept_id

This query method returns results reasonably fast (as shown in figure 6.1) because

each component of the query is already indexed. The database makes a number of fast indexed lookups, then combines each of these results. The query speed is thus defined by the time required to join (combine) each of the lookups. Databases are optimized for this task and thus should select the fastest (fewest computations) method of combining the results. The worst-case scenarios for this type of search will be slow, but realistic searches are fast. A few key rules seem to govern search speed when using the `INNER JOIN` searches.

1. searches containing more elements (properties, concepts, or relations) take longer (*i.e.*, more simple lookups to combine)
2. smaller result sets are faster (*i.e.*, at least one of the simple lookups has a small result set)
3. wildcard concepts or relations tend to be slower

Appendix D

Graphical Search User Interface: Search UI

There were a number of graphical tools used during the development of this thesis. The "Search UI" was a valuable tool, because it allows both search access to a database and inspection of parse trees. Searching could be done for all the definitions of a specific term, simple text searches in the sentences of the database. Usually the sentences were the definitions, and search was for all graphs/sentences in the database matching specific sub graphs. The sub-graph search functionality uses a form of the linearly conceptual graph representation.

The parse inspection functionality included four levels of parses: the raw parse from the Stanford dependency parser, a post-processed and restructured graph, a labeled graph (processed with genus detection and has been classified by definition type), Sense Linked (connection search was run on it). The user can toggle on and off the automatic detection, inclusion, and subsequent removal of prefix as needed (for more information see Chapter 5). During parsing, the genus term is detected and annotated with a property. If the sentence was not a definition, the genus property is irrelevant.

Figure D.1 depicts the user interface. The text box in the top center is the sentence to parse (or just parsed in this case). Next to this is a button that re-parses the text. Beside this button are a few options. The Post Process drop-down menu contains a number different options for parsing the sentence:

Raw Parse The output from the Stanford parser put into a graph without post-processing.

Post Processed The output from the Stanford parser with post-parsing transformations, such as removal or collapse certain relations, collapse of names into a single

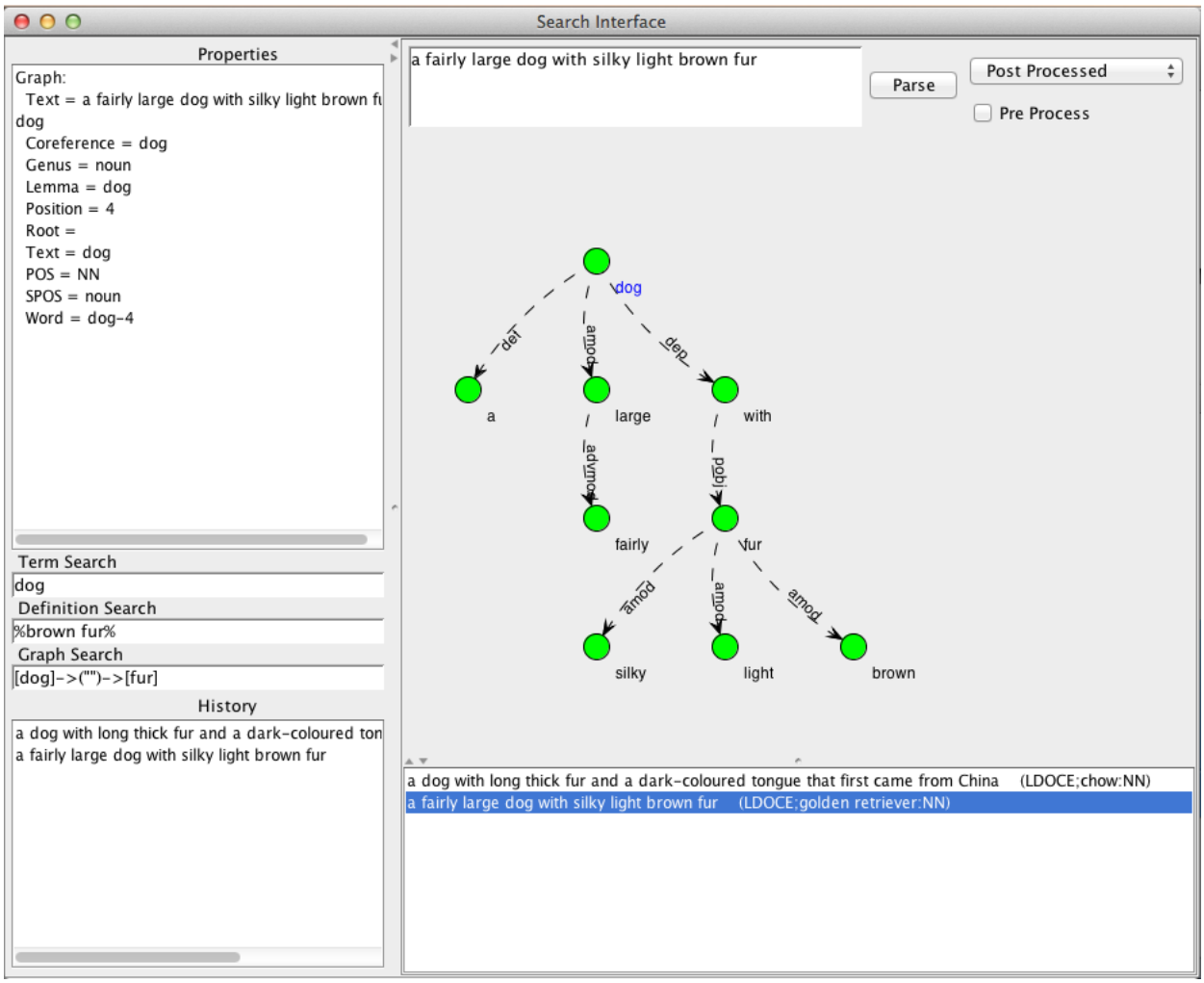


Figure D.1: Search UI

node, and restructuring conjunctions.

Labeled Post-processed graph that has been processed with genus detection and definition classification.

Sense Linked A label graph which has had connection search run on it, expanded some senses, creating attachments, and prune senses.

Database One can access different graphs for this sentence that are stored in the knowledge base.

The Pre Processed option toggles on/off the automatic detection and addition of a prefix to the sentence for optimal parsing.

Below the parse text box, the graph is visually represented, as a tree. The nodes represent concepts and are labelled. Edges are labeled with relations, though if this were a strict conceptual graph, it should be bipartite with nodes between the concepts representing the relations. The "dog" label is in blue, because it is currently selected.

In the top left corner there is a list of properties for the graph and selected nodes. In this example, since the dog node is selected, the properties of the graph and the dog node are presented.

Below the property list are all the text boxes for searching and below that a history of parsed sentences. The results of a search are presented in the list below the graph. Two search results are presently displayed, both from *Longman Dictionary of Contemporary English* (*LDOCE*).

The *term search* text box finds definitions for the term entered into it. Once the text has been entered into this box, press enter.

The *definition* text box finds sentences (or definitions) matching the text entered into it. "%" is treated as a wildcard of unlimited length for searching, thus this search is for sentences containing "brown fur". Once the text is entered into this box, press enter.

The *Graph* text box finds sentences, which contain the linear subgraph entered into it. "[dog]" and "[fur]" are concepts, while "("")" is an anonymous relation. The relation could be specified as "(with)" or "("with")", but all anonymous elements must have the empty double quotes ("").