Semantic Inference at the Lexical-Syntactic Level

Roy Bar-Haim

Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar Ilan University

Ramat Gan, Israel January 2010

This work was carried out under the supervision of Prof. Ido Dagan (Department of Computer Science), Bar-Ilan University.

Abstract

Semantic inference is concerned with deriving target meanings from texts. Within the *textual entailment* framework, this is reduced to inferring a textual statement from a source text, which captures the semantic inferences needed by many text understanding applications. Classical approaches to semantic inference rely on logical representations for meaning, which may be viewed as being "external" to the natural language itself. However, practical applications usually adopt shallower lexical or lexical-syntactic representations, which correspond closely to language structure. In many cases, such approaches lack a principled meaning representation and inference framework.

This thesis first presents an in-depth empirical analysis of the entailment task. We compare different levels for modeling entailment, and identify the prominent types of semantic knowledge required for entailment inference. Our analysis showed that lexical-syntactic representations are more powerful than lexical representations, and can model entailment well in many cases.

We then introduce a generic semantic inference framework that operates directly on language-based structures, particularly syntactic trees. New trees are inferred by applying entailment rules, which specify tree transformations and provide a unified representation for varying types of inference knowledge. Based on this formalism, we describe the development of a novel comprehensive resource of entailment rules for generic linguistic structures. Additional rules for specific lexical-based inferences,

which were derived automatically from a variety of semantic resources, were incorporated as well.

To make our inference approach practical, we also present a novel packed data-structure and a corresponding algorithm for a scalable implementation of our formalism. We proved the validity of the new algorithm and established its efficiency analytically and empirically.

Our implemented inference engine was evaluated on two types of tasks. It was first applied to the task of relation extraction from a large corpus. This novel setting allows evaluation of knowledge-based inferences over a reliable real-world distribution of texts. Second, it was applied to the standard recognizing textual entailment (RTE) benchmarks. In order to cope with the more complex RTE examples, we complemented our knowledge-based inference engine with a machine-learning-based entailment classifier, which provides necessary approximate matching capabilities. The inference engine was shown to have substantial contribution on both tasks, illustrating the utility of our approach.

Preface

Portions of this thesis are joint work and have appeared elsewhere.

Chapter 3 is based on "Definition and Analysis of Intermediate Entailment Levels", which appeared in the proceedings of the ACL-2005 Workshop on Empirical Modeling of Semantic Equivalence and Entailment (Bar-Haim et al., 2005). Chapter 4 and section 8.1 extend the paper "Semantic Inference at the Lexical-Syntactic Level", which appeared in the proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07) (Bar-Haim et al., 2007). Chapter 5 and portions of chapter 8 are an extension of "A Compact Forest for Scalable Inference over Entailment and Paraphrase Rules", which appeared in the proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009) (Bar-Haim et al., 2009a). Chapter 7 and portions of chapter 8 are based on "Efficient Semantic Deduction and Approximate Matching over Compact Parse Forests", which appeared in the proceedings of the First Text Analysis Conference (TAC 2008) (Bar-Haim et al., 2009b).

This work was partially funded by the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886, the Israel Science Foundation grant 1112/08, and the ITCH collaboration project of FBK/irst, University of Haifa and Bar-Ilan University.

Acknowledgements

First and foremost, I would like to thank Ido Dagan for being the best advisor I could hope for. I was fortunate to work on textual entailment during its formative years, when it evolved from an abstract idea to a fast-growing research field. I was even more fortunate to have as my advisor the person whose vision started it all, and who has been the main promoter of textual entailment ever since. Ido encouraged me to consider the fundamental problems, and strive to find clean, principled solutions for them. I thank Ido for his guidance, encouragement, support and friendship, and for all that I have learned from him.

I would also like to thank my Master's advisor, Yoad Winter, for introducing me to Natural Language Processing. I found myself going back to his courses notes long after I took them.

I am deeply grateful to the members of Bar-Ilan Natural Language Processing Lab. Their friendship made these long years so much fun, and the synergetic combination of our individual research efforts into one big group effort allowed me to get much further with my research than I could have achieved alone. In particular, I wish to thank Oren Glickman, Iddo Greental, Eyal Shnarch, Jonathan Berant and Shachar Mirkin for their collaboration. I would like to give special thanks to Idan Szpektor, who shared with me this long and winding road from the beginning, for the fruitful discussions we had, his close collaboration and his friendship.

I am grateful to Satoshi Sekine, Ralph Grishman, and the members of the Proteus

Project at New York University, for their hospitality during my summer internship, which was such an instructive and fun experience. I also thank Alfio Massimiliano Gliozzo and Claudio Giuliano for their research collaboration.

I would like to thank my parents, Amnon and Sara, for their endless love and faith in me, and their constant encouragement and support all these years. It's been a long way since they bought me my first computer, when I was ten years old, and it all started there... I am also grateful to my parents-in-law, Arie and Ella, for their devoted help and support. I thank my children, Shiri and Omer, for bringing joy to my life and giving me precious moments where I could forget all about this thesis...

Finally, I would like to thank my wife, Haya. Her endless love, support and understanding made it possible for me to embark on this endeavor, and complete it successfully. Having the opportunity to pursue my Ph.D was Haya's precious gift to me, which I will always treasure.

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning
IE	Information Extraction
IR	Information Retrieval
LHS	Left-Hand Side
NLP	Natural Language Processing
POS	Part of Speech
QA	Question Answering
RHS	Right-Hand Side
RTE	Recognising Textual Entailment
TE	Textual Entailment

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Textual Entailment

A fundamental phenomenon of natural language is the variability of semantic expression, where the same meaning can be expressed by or inferred from different texts. For example, the sentences *"I bought a Chihuahua"*, *"My dog is clever"* and *"I own a dog"* all imply the proposition *"I have a dog"*. Many natural language processing (NLP) applications, such as Question Answering (QA), Information Extraction (IE), and text summarization need to model this variability in order to recognize that a particular target meaning can be inferred from different text variants. Dagan and Glickman (2004) suggested that major semantic inferences needed by such applications can be cast in terms of *textual entailment (TE)*, the task of judging whether the truth of one text fragment follows from another text. In the TE framework, the entailing and entailed texts are termed *text* and *hypothesis*, respectively.

In the last few years, textual entailment became an emerging research field in

NLP. The four rounds of the *Recognising Textual Entailment (RTE) Challenges* (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007, 2008) attracted submissions from dozens of leading research groups world wide, and yielded many publications in major conferences and journals. The task in these challenges is to classify text-hypothesis (t,h) pairs as either entailing or non-entailing. For example, the following (t,h) pair is a positive (entailing) instance, taken from the RTE-2 development set.

t	Also, in a landmark deal with Disney, iTunes is now offering current and past episodes from two of the most popular shows on television.
h	iTunes does business with Disney.

Recently, several researchers have successfully integrated their RTE system into various text-understanding applications such as question answering (Hickl and Harabagiu, 2006; Iftene and Balahur-Dobrescu, 2008; Negri et al., 2008), Summarization (Harabagiu et al., 2007) and intelligent tutoring (Nielsen et al., 2009). Hickl and Harabagiu, for instance, improved the accuracy of their question answering (QA) system by 20% using their entailment system. Thus, the goal of TE research may be viewed as developing generic *entailment engines* that could be plugged into various text-understanding applications. These engines would encapsulate all needed semantic inferences, analogously to the current use of morphological analyzers and parsers for handling morphology and syntax. Using generic, off-the-shelf entailment modules would have a profound impact on text understanding applications. They would become much easier to develop, and to keep up-to-date with recent advancements in inference technology.

1.1.2 Entailment Systems

Entailment engines usually combine two different types of inferences. Some inferences can be based on available knowledge, such as information about synonyms, paraphrases, world knowledge relationships etc. In the general case, however, some knowledge gaps arise and it is not possible to derive a complete "proof" based on available inference knowledge. Such situations are typically handled through approximate matching methods. Clearly, the key for advancing entailment technology is incorporation of high quality, wide-coverage knowledge bases, which would reduce our dependence on heuristic, and less accurate inference mechanisms.

A major decision in designing entailment engines is the choice of internal representation for the input texts, over which the inference process is applied. According to the traditional formal semantics approach, inference is conducted at the logical level. Texts are first translated into propositions in some logical form and then new propositions are inferred from interpreted texts by a logical theorem prover. However, practical text understanding systems usually employ shallower lexical and lexical-syntactic representations, sometimes augmented with partial semantic annotations like word senses, named-entity classes and semantic roles.

While practical semantic inference is mostly performed over linguistic rather than logical representations, such practices are typically partial and quite ad-hoc, and lack a clear formalism that specifies how inference knowledge should be represented and applied. Well-formalized models seem important for applied semantic inference research, analogously to their role in parsing and machine translation.

1.2 Goals

The goal of this thesis is to develop an entailment engine operating at the lexical-syntactic level, following the common practice for text understanding applications.

Within this setting, we define the following requirements from our engine:

1. *Expressiveness:* The engine should allow incorporation of diverse types of inference knowledge, and allow their combination and composition.

2. *Succinct, well formalized approach:* Our goal is to develop a principled, well-formalized approach to semantic inference at the lexical-syntactic level. We aim at a simple, compact formalism that allows unified representation and inference mechanisms for diverse types of inference knowledge, while providing the required level of expressiveness. In particular, we try to avoid unnecessary semantic annotations, and make only minimal extensions to the common lexical-syntactic representation.

3. *Efficiency and scalability:* Combining and chaining inferences from several large-scale knowledge sources may yield a large space of possible entailed propositions. The engine should accommodate such large search spaces, so that complex inferences could be attempted.

An entailment engine meeting the above desiderata would provide a much needed platform for investigating knowledge-based semantic inference. As shown in the next chapter, current entailment systems only partially meet these requirements.

1.3 Contributions

The main novel contributions of this thesis are summarized in the following:

Empirical analysis of the entailment task: Entailment recognition is a complex task that involves diverse inference types at various levels (lexical, syntactic and semantic). In order to gain better understanding of the problem, we propose a novel methodology to decompose the entailment task into subtasks, and analyze

the contribution of individual NLP components for these subtasks. Applying our methodology to the RTE-1 dataset confirmed the appropriateness of lexical-syntactic representation for modeling entailment. The results also stress the importance of paraphrases and syntactic transformations for entailment inference. These findings motivated and guided the rest of this research.

An inference formalism over parse trees: We define and implement a semantic proof system that operates directly over syntactic parse trees, thus avoiding the complexities of logical interpretation. New parse trees are derived using entailment rules, which provide a principled and uniform mechanism for incorporating a wide variety of inference knowledge types. Based on this formalism, we integrated into the system diverse types of semantic knowledge from various sources, including WordNet, Wikipedia, automatically learned lexical-syntactic inference rules, generic syntactic transformations and so on.

A rule base for generic linguistic structures: Based on our inference formalism, a first comprehensive rule base for generic linguistic structures has been developed. These rules capture inferences associated with common syntactic structures (passive/active, appositions, determiners etc.), and detect contexts which affect the polarity of predicates. For example, *"I convinced him to dance"* entails *"He danced"*, while *"I asked him to dance"* does not.

A packed data structure for scalable inference: In our formalism, each rule application generates a new parse tree (a consequent). However, explicit generation of each consequent may lead to exponential explosion. We propose a solution for this problem, based on a novel packed data-structure for efficient representation of entailed consequents, and a corresponding inference algorithm. We proved that the new algorithm is a valid implementation of our formalism, and established its efficiency analytically and empirically.

The inference engine was evaluated on two types of tasks:

1. Relation extraction from a large corpus. This novel setting allows evaluation of knowledge-based inferences over a reliable real-world distribution of texts.

2. Recognizing textual entailment (RTE). In order to cope with the more complex RTE examples, we complemented our knowledge-based inference engine with a machine-learning-based entailment classifier, which provides necessary approximate matching capabilities.

The inference engine was shown to have substantial contribution on both tasks, illustrating the utility of our approach.

1.4 Thesis Outline

The rest of the thesis is organized as follows: chapter 2 provides the relevant background on the textual entailment task, available semantic resources, previous approaches to the task and their limitations. Chapter 3 presents an empirical analysis of the entailment task. We compared different levels for modeling entailment (lexical vs. lexical-syntactic) and mapped prominent inference types at each level. This analysis led to better understanding of the problem, and guided the rest of the research.

The core of our approach, a semantic inference formalism that operates directly over syntactic parse trees, is defined in chapter 4. Inference knowledge is represented uniformly as *entailment rules*, encoding tree transformations. In chapter 5 we present an efficient implementation of this formalism, using a novel packed data structure termed *compact forest*. Based on our formalism, we describe in chapter 6 the development of a novel rule base for generic linguistic phenomena. Chapter 7 describes a full-blown entailment system built around our implemented proof system.

In particular, we describe a module for approximate entailment classifcation, and how it operates over our compact forest data structure.

Chapter 8 reports the evaluation of our system on a relation extraction task and on the RTE benchmarks. In addition, it evaluates the efficiency of the compact forest data structure, and compares it to a naïve implementation of the formalism. Chapter 9 compares the approach developed in this thesis to related work. Finally, chapter 10 concludes the thesis contributions and suggests directions for future work.

Chapter 2

Background

2.1 Textual Entailment

2.1.1 Motivation

Identifying that the same meaning is expressed by, or can be inferred from, various language expressions is one of the main challenges in natural language understanding. The need to recognize such semantic variability is common to various applications such as Information Extraction (IE), Question Answering (QA), multi-document summarization, and Information Retrieval (IR). However, despite the common need, resolving semantic variability has been studied largely in an application-specific context, by disjoint research communities. This situation led to redundant research efforts, and hampered the sharing of new advancements across these communities.

This observation led Dagan and Glickman to propose a unifying framework for modeling language variability, which they termed *Textual Entailment* (TE) (Dagan and Glickman, 2004). Textual entailment recognition is the task of deciding, given two text fragments, whether the meaning of one text is entailed (can be inferred) from another text. They showed that this task captures generically a broad range of

inferences that are relevant for multiple applications. For example, QA systems need to verify that the retrieved passage text in which the answer was found indeed entails the selected answer. Given the question *"Who is John Lennon's widow?"*, the text *"Yoko Ono unveiled a bronze statue of her late husband, John Lennon, to complete the official renaming of England's Liverpool Airport as Liverpool John Lennon Airport."* entails the expected answer *"Yoko Ono is John Lennon's widow"*. Modeling answer validation as entailment involves the following steps: first, the question is transformed to affirmative form, representing a template for a candidate answer e.g. *"X is John Lennon's widow"*. We then plug the candidate answer term (*"Yoko Ono"*) into this template, obtaining a candidate answer, and finally we check if the candidate answer is entailed from the passage in which the answer term was found.

Similarly, IE systems need to validate that the given text indeed entails the semantic relation that is expected to hold between the extracted slot fillers (e.g. *'X works for Y'*). IR queries such as "Alzheimer's drug treatment"[1] can be rephrased as propsitions (e.g. "Alzheimer's disease is treated using drugs"), which are expected to be entailed from relevant documents. Multi-document summarization systems, when selecting sentences to be included in the summary, should verify that the meaning of the candidate sentence is not entailed by sentences already in the summary, to avoid redundancy.

As illustrated by these examples, judging entailment between given texts is useful for many different text understanding applications. Thus, the textual entailment framework may be the basis for defining an *entailment engine*, a generic semantic inference module to be used within these applications, analogously to the current use of syntactic parsers and morphological analyzers. Notice that textual entailment is defined as a relation between surface texts, and is not bound to a particular semantic representation. This allows a "black-box" view of the entailment engine, where the

[1]This was one of the topics in the TREC-6 IR benchmark (Voorhees and Harman, 1997).

input/output interface is independent from the internal implementation, which may employ various types of semantic interpretation and representation.

We next give a definition of textual entailment, and describe the benchmarks developed for the evaluation of entailment engines.

2.1.2 Definition and Evaluation

We consider an applied notion of textual entailment, defined as a directional relation between two text fragments, termed t - the entailing text, and h - the hypothesized entailed text.

> *Definition:* t *entails* h ($t \Rightarrow h$) if, typically, a human reading t would infer that h is most likely true.

This (somewhat informal) definition aims to capture user expectations from text understanding applications, and can be tested by reference to human judgments, as common in most NLP tasks. It is based on (and assumes) common human understanding of language as well as common background knowledge. *Textual entailment recognition* is the task of deciding, given t and h, whether t entails h.

A necessary step in transforming textual entailment from a theoretical idea into an active empirical research field was the introduction of benchmarks and an evaluation forum for entailment systems. Dagan, Glickman and Magnini (Dagan et al., 2006) initiated in 2004 a series of contests under the PASCAL Network of Excellence, known as *The PASCAL Recognising Textual Entailment Challenges* (*RTE* in short) (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007, 2008). The RTE challenges attracted an increasing number of researchers from leading groups worldwide, from both academia and industry. The impressive success of these challenges established textual entailment as a rapidly-growing research field. It became the subject of a multitude of papers in recent NLP conferences, and is consequently

ID	Text	Hypothesis	Task	Judgment
77	Google and NASA announced a working agreement, Wednesday, that could result in the Internet giant building a complex of up to 1 million square feet on NASA-owned property, adjacent to Moffett Field, near Mountain View.	Google may build a campus on NASA property.	SUM	YES
110	Drew Walker, NHS Tayside's public health director, said: "It is important to stress that this is not a confirmed case of rabies."	A case of rabies was confirmed.	IR	NO
294	Meanwhile, in an exclusive interview with a TIME journalist, the first one-on-one session given to a Western print publication since his election as president of Iran earlier this year, Ahmadinejad attacked the "threat" to bring the issue of Iran's nuclear activity to the UN Security Council by the US, France, Britain and Germany.	Ahmadinejad is a citizen of Iran.	IE	YES
387	About two weeks before the trial started, I was in Shapiro's office in Century City.	Shapiro works in Century City.	QA	YES
415	The drugs that slow down or halt Alzheimer's disease work best the earlier you administer them.	Alzheimer's disease is treated using drugs.	IR	YES
691	Arabic, for example, is used densely across North Africa and from the Eastern Mediterranean to the Philippines, as the key language of the Arab world and the primary vehicle of Islam.	Arabic is the primary language of the Philippines.	QA	NO

Table 2.1: Examples of text-hypothesis pairs, taken from the RTE-2 development set

being listed regularly as one of the solicited topics in calls for papers.

The RTE datasets consist of manually collected text fragment pairs, termed *text* (t), a sentence or a few sentences, and *hypothesis* h, typically a single sentence, which should be judged for entailment. The pairs represent correct and incorrect inferences in various application types, and were collected mostly from actual system outputs. Since RTE-2, these applications include IE, IR, QA and MDS (denoted as "SUM" in the RTE datasets). Each dataset was split into a development set and a test set, each containing a few hundreds of pairs[1].

Table 2.1 shows some examples from the RTE-2 development set (Bar-Haim et al., 2006). These examples illustrate the two primary differences between the TE definition and classical definitions of semantic entailment:

"Most likely" entailment the TE definition allows inference which are very probable, but not completely certain. For instance, in pair #387 one could claim that although Shapiro's office is in Century City, he actually never arrives to his office, and works elsewhere. However, this interpretation of t is very unlikely, and so the entailment holds with high likelihood.

Background knowledge The TE definition allows presupposition of common knowledge, such as: a company has a CEO, a CEO is an employee of the company, an employee is a person, etc. For instance, in pair #294, the entailment depends on knowing that the president of a country is also a citizen of that country.

2.2 Determining Entailment

Consider the following *(t,h)* pair:

[1]Except for RTE-4, for which only a test set was released

t	The oddest thing about the UAE is that only 500,000 of the 2 million people living in the country are UAE citizens.
h	The population of the United Arab Emirates is 2 million.

Understanding that $t \Rightarrow h$ involves several inference steps. First, we infer from the reduced relative clause in *"2 million people living in the country"* the proposition:

(1) *2 million people live in the country.*

Next, we observe that *"the country"* refers to *"the UAE"*, so we can rewrite (1) as

(2) *2 million people live in the UAE.*

Knowing that *"UAE"* is an acronym for *"United Arab Emirates"*, we further obtain:

(3) *2 million people live in the United Arab Emirates.*

which we finally paraphrase to obtain h:

(4) *The population of the United Arab Emirates is 2 million.*

In general, entailment inference involves diverse types of linguistic and world knowledge, including knowledge about relevant syntactic phenomena (e.g. relative clause), paraphrasing ('*X people live in Y*$\rightarrow$ *the population of Y is X*', lexical knowledge ('*UAE*$\rightarrow$ *United Arab Emirates*') and so on. It may also require co-reference resolution, e.g. for substituting *"the country"* with *"UAE"*. We may think of all these types of knowledge as representing *entailment rules*, which define derivation of new entailed propositions, or *consequents*. In this thesis we develop a formal inference framework based on entailment rule application. For the current discussion, however, an informal notion of entailment rules would suffice.

The above example illustrates the derivation of h from t through a sequence of entailment rule applications, a procedure generally known as *forward chaining*. Finding

the sequence of rule applications that would get us from t to h (or as close as possible) is thus a search problem, defined over the space of all possible rule application chains.

Ideally, we would like to base our entailment engine solely on trusted knowledge-based inferences. In practice, however, available knowledge is incomplete, and full derivation of h from t is often not feasible. Therefore, requiring strict knowledge-based "proofs" is likely to yield limited recall. Alternatively, we may back off to a more heuristic approximate entailment classification. Approximate classification typically involves two types of considerations: first, how close to h did we get with available knowledge? If the remaining gap is sufficiently small, the pair may still be judged as entailing. Second, can we find cues for non-entailment? For instance, can we identify crucial parts of h that are missing from t? In our example, if t did not mention any quantities, we could infer with high probability that it does not entail h. Other possible cues are mismatches between t and h. For example, the same verb appears with different polarity in t and h (e.g. *didn't buy* vs. *bought*).

The next two sections survey these two complementary inference types: knowledge-based inference, which is our focus in this research, and approximate entailment matching and classification.

2.3 Knowledge-Based Inference

In this section we describe some of the common resources for entailment rules (2.3.1), and their use in textual entailment systems (2.3.2).

2.3.1 Semantic Knowledge Resources

Lexical Knowledge Lexical-semantic relations between words or phrases play an important role in semantic inference. The most prominent lexical resource is WordNet (Fellbaum, 1998), a manually composed lexical database. In WordNet, nouns, verbs,

adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of semantic and lexical relations. The following relations are typically utilized for inference:

- *Synonyms* such as '*buy* ↔ *purchase*'
- *Antonyms* such as '*win* ↔ *lose*'
- *Hypernyms/Hyponyms* ("is-a" relations) '*violin*→ *musical instrument*'
- *Meronyms* ("part-of" relations) such as '*Provence*→ *France*'
- *Derivations* such as '*meeting*→ *meet*'

Despite its wide coverage, the information in WordNet is incomplete in several respects. First, its coverage, in particular of proper nouns, is limited. Second, the above relations do not cover many types of relevant entailments. For example, the lexical rule '*Abbey Road*→ *Beatles*' allows the inference of *"I listened to The Beatles"* from *"I listened to Abbey Road"* (Shnarch et al., 2009). Another common problem is rules for rare word senses, e.g. '*have*→ *give birth*'. Without sense disambiguation, most applications of such rules would be incorrect.

Several other lexical resources have been derived automatically from various sources, using diverse methods. Much of their extracted knowledge is complementary to WordNet, however their accuracy is typically lower. Snow et al. (2006a) presented a method for automatically expanding WordNet with new synsets, achieving high precision. Lin's thesaurus (Lin, 1998) is based on distributional similarity: words appearing in similar contexts in a given corpus are considered similar. Recently, several works aimed to extract lexical-semantic knowledge from Wikipedia, the online free encyclopedia, utilizing its metadata (e.g. info boxes, links and redirects), as well as textual definitions, using patterns such as *'X is a Y'* (e.g. *"Ramat-Gan is a city in the Tel*

Aviv district of Israel") (Kazama and Torisawa, 2007; Ponzetto and Strube, 2007; Shnarch et al., 2009; Lehmann et al., 2009, and others). For a recent empirical study on the inferential utility of common lexical resources, see (Mirkin et al., 2009).

Paraphrases and Lexical-Syntactic Inference Rules These rules typically represent entailment or equivalence relations between predicates, including the correct mapping between their arguments, e.g. '*acquisition of Y by X* $\rightarrow$ *X purchase Y*'. Much work has been dedicated to unsupervised learning of such entailment rules or paraphrases (bi-directional entailments) from comparable corpora (Barzilay and McKeown, 2001; Barzilay and Lee, 2003; Pang et al., 2003), by querying the Web (Ravichandran and Hovy, 2002; Szpektor et al., 2004), or from a local corpus (Lin and Pantel, 2001; Glickman and Dagan, 2003; Bhagat and Ravichandran, 2008; Szpektor and Dagan, 2008). In particular, the DIRT resource of Lin and Pantel has been widely used by textual entailment systems. The common idea underlying these algorithms is that predicates sharing the same argument instantiations are likely to be semantically related.

A special case of lexical-syntactic rules is nominalization rules, which map between predicates in their verbal and nominal form, e.g. '*acquisition of Y by X* $\rightarrow$ *X acquire Y*'. NOMLEX-PLUS (Meyers et al., 2004) is a lexicon containing mostly nominalizations of verbs, with allowed argument structures (e.g. *'X's acquisition of Y'/'Y's acquisition by X'* etc.). It is an extension of the manually-created NOMLEX lexicon, obtained by semi-automatic integration of several dictionary resources. Recently, Szpektor and Dagan (Szpektor and Dagan, 2009) introduced *Argument-mapped WordNet (AmWN)*, a resource for entailment rules between verbal and nominal predicates , including their argument mapping, based on WordNet and NomLex-plus, verified statistically through intersection with the unary-DIRT algorithm (Szpektor and Dagan, 2008).

Syntactic transformations Textual entailment often involves inference over generic syntactic phenomena such as passive/active transformations, appositions, conjunctions etc., as illustrated in the following examples:

- John smiled and laughed ⇒ John laughed (conjunction)
- My neighbor, John, came in ⇒ John is my neighbor (apposition)
- The paper I'm reading is interesting ⇒ I'm reading a paper (relative clause).

While syntactic transformations have been addressed in previous work to some extent (de Salvo Braz et al., 2005; Romano et al., 2006), no comprehensive rule base has been available so far. In this thesis we develop a syntactic rule base for entailment, based on a survey of relevant linguistic literature, as well as on extensive data analysis (Chapter 6).

2.3.2 The use of Semantic Knowledge in Entailment Systems

Entailment systems usually represent t and h as trees or graphs, based on their syntactic parse, predicate-argument structure and various semantic relations. Entailment is then determined by measuring how well h is *matched* (or *embedded*) in t, or by estimating the *distance* between t and h, commonly defined as the cost of transforming t into h. Various methods for approximate matching and heuristic transformations of graphs and trees have been proposed, which we briefly cover in the next section. The role of semantic knowledge in this general scheme is to bridge the gaps between t and h that result from language variability. For example, applying the lexical-semantic rule '*purchase*→ *buy*' to t allows the matching of the word *buy* appearing in h with the word *purchase* appearing in t.

Most RTE systems restricted both the type of allowed entailment rules and the search space. Systems based on lexical (word-based or phrase-based) matching of h

in t (Haghighi et al., 2005; MacCartney et al., 2008) or on heuristic transformation of t into h (Kouylekov and Magnini, 2005; Harmeling, 2009) typically applied only lexical rules (without variables), where both sides of the rule are matched directly in t and h.

Hickl (2008) derived from a given (t, h) pair a small set of consequents that he terms *discourse commitments*. These consequents are based on syntax (conjunctions, appositions, relative clauses etc), co-reference, predicate-argument structure, the extraction of certain relations, and several paraphrases acquired from the Web. The commitments were generated by several different tools and techniques. Pairs of commitments derived from t and h were fed into the next stages of the RTE system – lexical alignment and entailment classification.

(de Salvo Braz et al., 2005) were the first to incorporate syntactic and semantic entailment rules in a comprehensive entailment system. In their system, entailment rules are applied over hybrid syntactic-semantic structures called *concept graphs*. When the left hand side (LHS) of a rule is matched in the concept graph, the graph is augmented with an instantiation of the right hand side (RHS) of the rule. After several iterations of rule application, their system attempts to embed the hypothesis in the augmented graph. Other types of semantic knowledge, such as verb normalization and lexical substitutions, are applied either before rule application (at preprocessing time) or after rule application, as part of hypothesis subsumption (embedding).

Finally, several entailment systems (Bos and Markert, 2005; Tatu and Moldovan, 2005) represented both (t,h) pairs and the entailment knowledge as logic formulae, and applied a theorem prover for inference. Bos and Markert (2006) found that without much inference knowledge, a simple weighted lexical overlap baseline outperforms their logical inference system. Tatu et al.'s commercial system was one of the best performing systems in both RTE2 and RTE3 (Tatu et al., 2006; Tatu and Moldovan, 2007). It is based on proprietary tools for deriving rich semantic representations, and

on extensive knowledge bases for inference. These tools and knowledge have been developed over several years, and required heavy investment in their development.

2.4 Approximate Entailment Classification

Semantic knowledge is always incomplete, and therefore in most cases knowledge-based inference must be complemented with approximate, heuristic methods for determining entailment. As previously mentioned, most RTE systems employed only limited amount of semantic knowledge, and focused on methods for approximate entailment classification. In this section we look more closely at these methods.

A common architecture for RTE systems (Hickl et al., 2006; Snow et al., 2006b; MacCartney et al., 2006) comprises the following stages:

1. *Linguistic processing*: including syntactic (and possibly semantic) parsing, named-entity recognition, co-reference resolution etc. Often, t and h are represented as trees or graphs, where nodes correspond to words and edges represent relations between words.

2. *Alignment*: find the best mapping from h nodes to t nodes, taking into account both node and edge matching. Several optimization and machine-learning-based approaches have been proposed for finding this alignment, some of which also utilized hand-aligned (t,h) pairs for training.

3. *Entailment classification*: Based on the alignment found, a set of features is extracted and passed to a classifier for determining entailment. These features measure the alignment quality, and also try to detect cues for false entailment. For example, if a node in h is negated but its aligned node in t is not negated, it may indicate false entailment. Other relevant contexts for mismatches are modal verbs, quantifiers, conditionals and so on. Some additional examples for

non-entailment cues include mismatching relations between aligned endpoints, and unaligned entities in h.

An alternative approach aims to transform the text into the hypothesis, rather than aligning them. Kouylekov and Magnini (2005) applied a tree edit distance algorithm for textual entailment. Three types of transformations were defined: node insertion, node deletion and node substitution. Each operation is assigned a cost and the algorithm aims to find the minimum-cost sequence of operations that transform t into h. Harmeling (2009) developed a probabilistic transformation-based approach. He defined a fixed set of operations, including syntactic transformations, WordNet-based substitutions and more heuristic transformations such as adding/removing a verb or a noun. The probability of each transformation was estimated from the development set.

Zanzotto et al. (2009) aimed to classify a given (t, h) pair by analogy to similar pairs in the training set. Their method is based on finding intra-pair alignment (i.e. between t and h) for capturing the transformation from t to h, and inter-pair alignment, capturing the analogy between the new pair (t, h) and a previously seen pair (t', h'). A cross-pair similarity kernel is then computed, based on tree kernel similarity applied to the aligned texts and the aligned hypotheses. Another cross-pair similarity kernel was proposed by Wang and Neumann (2007). They extracted *tree skeletons* from t and h, consisting of left and right *spines*, defined as unlexicalized paths starting at the root. They then found sections where t and h spines differ and compared these sections across pairs using a subsequence kernel.

2.5 Summary and Takeouts

The textual entailment framework provides unified modeling for semantic inferences underlying various text understanding tasks. Thus, the goal of textual entailment research may be viewed as developing entailment engines to be used as generic inference components within these applications.

Most text understanding applications, including textual entailment systems, operate over lexical-syntactic representations, possibly supplemented with some partial semantic annotation. Comparing current lexical-syntactic RTE systems surveyed in this chapter, to the desiderata defined in section 1.2, we observe the following gaps:

1. *Limited integration of semantic knowledge:* Most entailment systems do not incorporate much semantic knowledge, and instead focus on approximate methods for entailment classification. Often, these systems allow only application of lexical rules.

2. *Limited search space:* Most entailment systems do not allow composition (chaining) of semantic knowledge, and consider only entailment rules that match directly terms or phrases in t and h. Thus, these systems would miss inferences such as *David Bowie* $\Rightarrow$ *musician* $\Rightarrow$ *performer*, where the first step is based on knowledge exracted from Wikipedia, and the second step is provided by WordNet. In other cases (Harmeling, 2009), chaining of transformations is allowed but only in a rather fixed, heuristic order.

3. *Lack of a unified formalism:* Current systems employ multiple representations and inference mechanisms for different types of semantic knowledge, and lack a clear, unified formalsim for knowledge representation and inference.

Logic-based entailment systems provide a more formalized and expressive framework. However, this comes at the cost of increased complexity, making these systems much

harder to implement. Available tools for logic interpretation and inference are less mature than current syntactic parsers, in terms of robustness, efficiency and accuracy. Furthermore, interpretation into logic forms is often unnecessary, as many of the common inferences can be modeled with shallower representations. The successful logic-based entailment systems of Tatu et al. is based on proprietary tools and resources that are not publicly available, in which many person-years have been invested. Consequently, this approach has not been widely adopted.

The goal of this thesis is to make a step towards closing the above gaps. In chapters 4–7 we develop a well-formalized, expressive and efficient entailment approach for the lexical-syntactic level. The data analysis presented in the next chapter provides further justification for choosing lexical-syntactic representations for modeling entailment.

Chapter 3

Definition and Analysis of Intermediate Entailment Levels[1]

3.1 Introduction

As observed earlier, identifying entailment is a complex task that incorporates many levels of linguistic knowledge and inference. The complexity of modeling entailment was demonstrated in the first PASCAL Challenge Workshop on Recognizing Textual Entailment (RTE) (Dagan et al., 2006). Systems that participated in the challenge used various combinations of NLP components in order to perform entailment inferences. These components can largely be classified as operating at the lexical, syntactic and semantic levels (see Table 1 in Dagan et al., 2006). However, only little research was done to analyze the contribution of each inference level, and the contribution of individual inference mechanisms within each level.

In this chapter we suggest that decomposing the complex task of entailment into subtasks, and analyzing the contribution of individual NLP components for these subtasks would make a step towards better understanding of the problem, and for

[1] Joint work with Idan Szpektor.

pursuing better entailment engines. We set three goals for our study. First, we consider various levels of modeling entailment, and for each level we investigate an idealized setting where the relevant inference knowledge is complete. We explore how well these models approximate the notion of entailment, and analyze the differences between the outcome of the different levels. Second, for each of the presented levels, we evaluate the distribution (and contribution) of each of the inference mechanisms typically associated with that level. Finally, we suggest that the definitions of entailment at different levels of inference, as proposed in this chapter, can serve as guidelines for manual annotation of a "gold standard" for evaluating systems that operate at a particular level. Altogether, we set forth a possible methodology for annotation and analysis of entailment datasets.

We introduce two levels of entailment: *Lexical* and *Lexical-Syntactic*. We propose these levels as intermediate stages towards a complete entailment model. We define an entailment model for each level and manually evaluate its performance over a sample from the RTE test-set. We focus on these two levels as they correspond to well-studied NLP tasks, for which robust tools and resources exist, e.g. parsers, part of speech taggers and lexicons. At each level we included inference types that represent common practice in the field. More advanced processing levels which involve logical/semantic inference are less mature and were left beyond the scope of this study.

We found that the main difference between the lexical and lexical-syntactic levels is that the lexical-syntactic level corrects many false-positive inferences obtained by using only the lexical level, while introducing only a few false-positives of its own. As for identifying positive cases (recall), both systems exhibit similar performance, and were found to be complementary. Neither of the levels was able to identify more than half of the positive cases, which emphasizes the need for deeper levels of analysis. Among the different inference components, *paraphrases* stand out as a dominant contributor to the entailment task, while synonyms and derivational transformations

were found to be the most frequent at the lexical level.

Using our definitions of entailment models as guidelines for manual annotation resulted in a high level of agreement between two annotators, suggesting that the proposed models are reasonably well-defined.

Our study follows on previous work (Vanderwende and Dolan, 2006), which analyzed the RTE Challenge test-set to find the percentage of cases in which syntactic analysis alone (with optional use of thesaurus for the lexical level) suffices to decide whether or not entailment holds. Our study extends this work by considering a broader range of inference levels and inference mechanisms and providing a more detailed view. A fundamental difference between the two works is that while Vanderwende et al. did not make judgements on cases where additional knowledge was required beyond syntax, our entailment models were evaluated over all of the cases, including those that require higher levels of inference. This allows us to view the entailment model at each level as an idealized *system*, and to evaluate its overall success.

The rest of the chapter is organized as follows: section 3.2 provides definitions for the two entailment levels; section 3.3 describes the annotation experiment we performed, its results and analysis; section 3.4 concludes and presents planned future work.

3.2 Definition of Entailment Levels

In this section we present definitions for two entailment models that correspond to the *Lexical* and *Lexical-Syntactic* levels. For each level we describe the available inference mechanisms. Table 3.1 presents several examples from the RTE test-set together with annotation of entailment at the different levels.

No.	Text	Hypothesis	Task	Ent.	Lex. Ent.	Syn. Ent.
322	Turnout for the historic vote for the first time since the EU took in 10 new members in May has hit a record low of 45.3%.	New members joined the EU.	IR	true	false	true
1361	A Filipino hostage in Iraq was released.	A Filipino hostage was freed in Iraq.	CD	true	true	true
1584	Although a Roscommon man by birth, born in Rooskey in 1932, Albert "The Slasher" Reynolds will forever be a Longford man by association.	Albert Reynolds was born in Co. Roscommon.	QA	true	true	true
1911	The SPD got just 21.5% of the vote in the European Parliament elections, while the conservative opposition parties polled 44.5%.	The SPD is defeated by the opposition parties.	IE	true	false	false
2127	Coyote shot after biting girl in Vanier Park.	Girl shot in park.	IR	false	true	false

Table 3.1: Examples of text-hypothesis pairs, taken from the PASCAL RTE test-set. Each line includes the example number at the RTE test-set, the text and hypothesis, the task within the test-set, whether entailment holds between the text and hypothesis (Ent.), whether Lexical entailment holds (Lex. Ent.) and whether Lexical-Syntactic entailment holds (Syn. Ent.).

3.2.1 The Lexical entailment level

At the lexical level we assume that the text T and hypothesis H are represented by a bag of (possibly multi-word) terms, ignoring function words. At this level we define that entailment holds between T and H if every term h in H can be matched by a corresponding entailing term t in T. t is considered as entailing h if either h and t share the same lemma and part of speech, or t can be matched with h through a

sequence of lexical transformations of the types described below.

Morphological derivations This inference mechanism considers two terms as equivalent if one can be obtained from the other by some morphological derivation. Examples include nominalizations (e.g. '*acquisition* ↔ *acquire*'), pertainyms (e.g. '*Afghanistan* ↔ *Afghan*'), or nominal derivations like '*terrorist* ↔ *terror*'.

Ontological relations This inference mechanism refers to ontological relations between terms. A term is inferred from another term if a chain of valid ontological relations between the two terms exists (Andreevskaia et al., 2006). In our experiment we regarded the following three ontological relations as providing entailment inferences: (1) *synonyms* (e.g. '*free* ↔ *release*' in example 1361, Table 3.1); (2) *hypernyms* (e.g. '*produce*→ *make*') and (3) *meronyms/holonyms* (e.g. '*executive*→ *company*').

Lexical World knowledge This inference mechanism refers to world knowledge reflected at the lexical level, by which the meaning of one term can be inferred from the other. It includes both knowledge about named entities, such as '*Taliban*→ *organization*' and '*Roscommon* ↔ *Co. Roscommon*' (example 1584 in Table 3.1), and other lexical relations between words, such as WordNet's relations *cause* (e.g. '*kill*→ *die*') and *entail* (e.g. '*snore*→ *sleep*').

3.2.2 The Lexical-syntactic entailment level

At the lexical-syntactic level we assume that the text and the hypothesis are represented by the set of syntactic dependency relations of their dependency parse. At this level we ignore determiners and auxiliary verbs, but do include relations involving other function words. We define that entailment holds between T and H if the

relations within H can be "covered" by the relations in T. In the trivial case, lexical-syntactic entailment holds if all the relations composing H appear verbatim in T (while additional relations within T are allowed). Otherwise, such coverage can be obtained by a sequence of transformations applied to the relations in T, which should yield all the relations in H.

One type of such transformations are the lexical transformations, which replace corresponding lexical items, as described in sub-section 3.2.1. When applying morphological derivations it is assumed that the syntactic structure is appropriately adjusted. For example, "Mexico produces oil" can be mapped to "oil production by Mexico". The NOMLEX resource (Macleod et al., 1998) provides a good example for systematic specification of such transformations.

Additional types of transformations at this level are specified below.

Syntactic transformations This inference mechanism refers to transformations between syntactic structures that involve the same lexical elements and preserve the meaning of the relationships between them (as analyzed in Vanderwende and Dolan, 2006). Typical transformations include passive-active and apposition (e.g. 'An Wang, a native of Shanghai $\Leftrightarrow$ An Wang is a native of Shanghai').

Entailment paraphrases This inference mechanism refers to transformations that modify the syntactic structure of a text fragment as well as some of its lexical elements, while holding an entailment relationship between the original text and the transformed one. Such transformations are typically denoted as 'paraphrases' in the literature, where a wealth of methods for their automatic acquisition were proposed (Lin and Pantel, 2001; Shinyama et al., 2002; Barzilay and Lee, 2003; Szpektor et al., 2004). Following the same spirit, we focus here on transformations that are local in

nature, which, according to the literature, may be amenable for large scale acquisition. Examples include: '*X is Y man by birth→ X was born in Y*' (example 1584 in Table 3.1), '*X take in Y→ Y join X*'[1] and '*X is holy book of Y→ Y follow X*'[2].

Co-reference Co-references provide equivalence relations between different terms in the text and thus induce transformations that replace one term in a text with any of its co-referenced terms. For example, the sentence "Italy and Germany have each played twice, and they haven't beaten anybody yet."[3] entails "Neither Italy nor Germany have won yet", involving the co-reference transformation 'they ⇒ Italy and Germany'.

Example 1584 in Table 3.1 demonstrates the need to combine different inference mechanisms to achieve lexical-syntactic entailment, requiring world-knowledge, paraphrases and syntactic transformations.

3.3 Empirical Analysis

In this section we present the experiment that we conducted in order to analyze the two entailment levels, which are presented in section 3.2, in terms of relative performance and correlation with the notion of textual entailment.

3.3.1 Data and annotation procedure

The RTE test-set[4] contains 800 Text-Hypothesis pairs (usually single sentences), which are typical to various NLP applications. Each pair is annotated with a boolean

[1]Example no 322 in the PASCAL RTE test-set.
[2]Example no 1575 in the PASCAL RTE test-set.
[3]Example no 298 in the PASCAL RTE test-set.
[4]The complete RTE dataset can be obtained at http://www.pascal-network.org/Challenges/RTE/Datasets/

value, indicating whether the hypothesis is entailed by the text or not, and the test-set is balanced in terms of positive and negative cases. We shall henceforth refer to this annotation as the *gold standard*. We constructed a sample of 240 pairs from four different tasks in the test-set, which correspond to the main applications that may benefit from entailment: information extraction (IE), information retrieval (IR), question answering (QA), and comparable documents (CD). We randomly picked 60 pairs from each task, and in total 118 of the cases were positive and 122 were negative.

In our experiment, two of the authors annotated, for each of the two levels, whether or not entailment can be established in each of the 240 pairs. The annotators agreed on 89.6% of the cases at the lexical level, and 88.8% of the cases at the lexical-syntactic level, with Kappa statistics of 0.78 and 0.73, respectively, corresponding to 'substantial agreement' (Landis and Koch, 1997). This relatively high level of agreement suggests that the notion of lexical and lexical-syntactic entailment we propose are indeed well-defined.

Finally, in order to establish statistics from the annotations, the annotators discussed all the examples they disagreed on and produced a final joint decision.

3.3.2 Evaluating the different levels of entailment

	L	LS
True positive (118)	52	59
False positive (122)	36	10
Recall	44%	50%
Precision	59%	86%
F_1	0.5	0.63
Accuracy	58%	71%

Table 3.2: Results per level of entailment.

Table 3.2 summarizes the results obtained from our annotated dataset for both lexical (L) and lexical-syntactic (LS) levels. Taking a "system"-oriented perspective, the annotations at each level can be viewed as the classifications made by an idealized system that includes a perfect implementation of the inference mechanisms in that level. The first two rows show for each level how the cases, which were recognized as positive by this level (i.e. the entailment holds), are distributed between "true positive" (i.e. positive according to the gold standard) and "false positive" (negative according to the gold standard). The total number of positive and negative pairs in the dataset is reported in parentheses. The rest of the table details recall, precision, F_1 and accuracy.

The distribution of the examples in the RTE test-set cannot be considered representative of a real-world distribution (especially because of the controlled balance between positive and negative examples). Thus, our statistics are not appropriate for accurate prediction of application performance. Instead, we analyze how well these simplified models of entailment succeed in approximating "real" entailment, and how they compare with each other.

The proportion between true and false positive cases at the lexical level indicates that the correlation between lexical match and entailment is quite low, reflected in the low precision achieved by this level (only 59%). This result can be partly attributed to the idiosyncracies of the RTE test-set: as reported in (Dagan et al., 2006), samples with high lexical match were found to be biased towards the negative side. Interestingly, our measured accuracy correlates well with the performance of systems at the PASCAL RTE Workshop, where the highest reported accuracy of a lexical system is 0.586 (Dagan et al., 2006).

As one can expect, adding syntax considerably reduces the number of false positives - from 36 to only 10. Surprisingly, at the same time the number of true positive cases grows from 52 to 59, and correspondingly, precision rise to 86%. Interestingly,

		Lexical-Syntactic	
		T ⇒ H	T ⇏ H
Lexical	T ⇒ H	38	14
	T ⇏ H	21	45

(a) positive examples

		Lexical-Syntactic	
		T ⇒ H	T ⇏ H
Lexical	T ⇒ H	7	29
	T ⇏ H	3	83

(b) negative examples

Table 3.3: Correlation between the entailment levels. (a) includes only the positive examples from the RTE dataset sample, and (b) includes only the negative examples.

neither the lexical nor the lexical-syntactic level are able to cover more than half of the positive cases (e.g. example 1911 in Table 3.1).

In order to better understand the differences between the two levels, we next analyze the overlap between them, presented in Table 3.3. Looking at Table 3.3(a), which contains only the positive cases, we see that many examples were recognized only by one of the levels. This interesting phenomenon can be explained on the one hand by lexical matches that could not be validated in the syntactic level, and on the other hand by the use of paraphrases, which are introduced only in the lexical-syntactic level. (e.g. example 322 in Table 3.1).

This relatively symmetric situation changes as we move to the negative cases, as shown in Table 3.3(b). By adding syntactic constraints, the lexical-syntactic level was able to fix 29 false positive errors, misclassified at the lexical level (as demonstrated in example 2127, Table 3.1), while introducing only 3 new false-positive errors. This exemplifies the importance of syntactic matching for precision.

3.3.3 The contribution of various inference mechanisms

Inference Mechanism	f	$\triangle$R	%
Synonym	19	14.4%	16.1%
Morphological	16	10.1%	13.5%
Lexical World knowledge	12	8.4%	10.1%
Hypernym	7	4.2%	5.9%
Mernoym	1	0.8%	0.8%
Entailment Paraphrases	37	26.2%	31.3%
Syntactic transformations	22	16.9%	18.6%
Coreference	10	5.0%	8.4%

Table 3.4: The frequency (f), contribution to recall ($\triangle$R) and percentage (%), within the gold standard positive examples, of the various inference mechanisms at each level, ordered by their significance.

In order to get a sense of the contribution of the various components at each level, statistics on the inference mechanisms that contributed to the coverage of the hypothesis by the text (either full or partial) were recorded by one annotator. Only the positive cases in the gold standard were considered.

For each inference mechanism we measured its frequency, its contribution to the recall of the related level and the percentage of cases in which it is required for establishing entailment. The latter also takes into account cases where only partial coverage could be achieved, and thus indicates the significance of each inference mechanism for any entailment system, regardless of the models presented in this paper. The results are summarized in Table 3.4.

From Table 3.4 it stands that paraphrases are the most notable contributors to recall. This result indicates the importance of paraphrases to the entailment task and the need for large-scale paraphrase collections. Syntactic transformations are also shown to contribute considerably, indicating the need for collections of syntactic transformations as well. In that perspective, we propose our annotation framework as

means for evaluating collections of paraphrases or syntactic transformations in terms of recall.

Finally, we note that the co-reference moderate contribution can be partly attributed to the idiosyncracies of the RTE test-set: the annotators were guided to replace anaphors with the appropriate reference, as reported in (Dagan et al., 2006).

3.4 Conclusions

We presented the definition of two entailment models, Lexical and Lexical-Syntactic, and analyzed their performance manually. Our experiment shows that the lexical-syntactic level outperforms the lexical level in all measured aspects. Furthermore, paraphrases and syntactic transformations emerged as the main contributors to recall. These results suggest that a lexical-syntactic framework is a promising step towards a complete entailment model.

Beyond these empirical findings we suggest that the presented methodology can be used generically to annotate and analyze entailment datasets. In future work, it would be interesting to analyze higher levels of entailment, such as logical inference and deep semantic understanding of the text.

Chapter 4

An Inference Formalism Over Parse Trees

4.1 Introduction

The previous chapters highlighted the need for a more principled, well-formalized approach to semantic inference at the lexical-syntactic level. In this chapter we propose a step towards filling this gap, by defining a formalism for semantic inference over parse-based representations. All semantic knowledge required for inference is represented as *entailment rules*, which encode parse tree transformations, and each rule application generates a new consequent sentence (represented as a parse tree) from a source tree. Figure 4.1(b) shows a sample entailment rule, representing a passive-to-active transformation.

From a knowledge representation and usage perspective, entailment rules provide a simple unifying formalism for representing and applying a very broad range of inference knowledge. Some examples of this breadth are illustrated in Table 4.1. From a knowledge acquisition perspective, representing entailment rules at the lexical-syntactic level allows easy incorporation of rules learned by unsupervised methods,

Rule Type	Sources	Examples
Syntactic	Manually-composed	Passive/active, apposition, relative clause, conjunctions
Lexical -Syntactic	Learned with unsupervised algorithms (DIRT, TEASE), and derived automatically by integrating information from WordNet and Nomlex, verified using corpus statistics (AmWN)	*X's wife, Y → X is married to Y* *X bought Y → Y was sold to X* *X is a maker of Y → X produces Y*
Lexical	WordNet, Wikipedia	*steal→take*, *Albanian→Albania* *Janis Joplin→singer*, *Amazon→South America*
Polarity	Manually-composed, utilzing VerbNet and PARC's polarity lexicon	Verbal negation, modal verbs, conditionals, verb polarity

Table 4.1: Representing diverse knowledge types as entailment rules.

which seems essential for scaling inference systems. Interpretation into stipulated semantic representations, which is often difficult and is inherently a supervised semantic task for learning, is circumvented altogether. From a historical machine translation perspective, our approach is analogous to Transfer-based translation, as confronted with semantic interpretation into Interlingua. Our overall research goal is to explore how far we can get with such an inference approach, and identify the scope in which semantic interpretation may not be needed.

Given a source text, syntactically parsed, and a set of entailment rules, our formalism defines the set of consequents derivable from the text using the rules. Each consequent is obtained through a sequence of rule applications, each generating an intermediate parse tree, similar to a proof process in logic. In addition, new consequents may be inferred based on co-reference relations and identified traces. Our formalism also includes *annotation rules* that add features to existing trees, which may affect (e.g. block) subsequent entailment rule application. According to the formalism, a text t *entails* a hypothesis h if h is a consequent of t.

The rest of this chapter defines and illustrates each of the formalism components: sentence representation (section 4.2), entailment rules and their application (sections 4.3–4.4), inference based on co-reference relations and traces (section 4.5), and annotation rules (section 4.6). These components are composed into an inference process that specifies the set of inferrable consequents for a given text and a set of rules (section 4.7). Finally, section 4.8 extends the hypothesis definition, allowing h to be a template rather than a proposition.

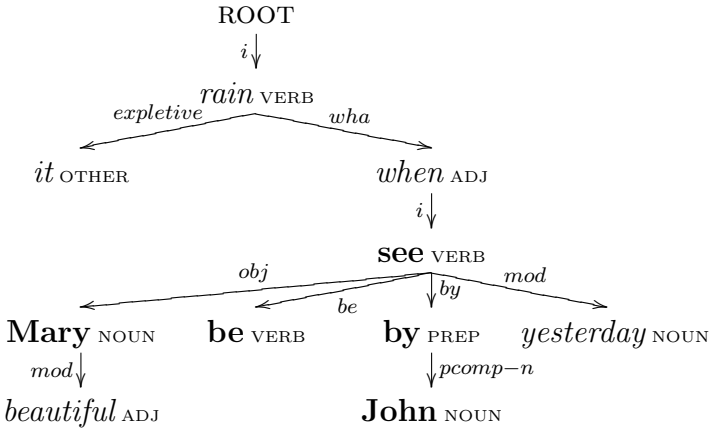
4.2 Sentence Representation

Our general approach assumes that sentences are represented by some form of parse trees. In this thesis we focus on dependency tree representation, which is often preferred to capture directly predicate-argument relations. Two dependency trees are shown in figure 4.1(a). Nodes represent words and hold a set of features and their values. These features include the word lemma and part-of-speech, and additional features that may be added during the inference process. Edges are annotated with dependency relations.

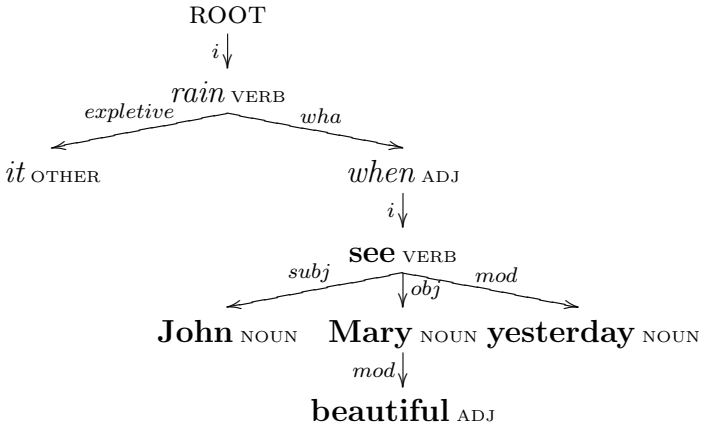
4.3 Entailment Rules

A rule '$L \rightarrow R$' is primarily composed of two *templates*, *left-hand-side (LHS)* L and *right-hand-side (RHS)* R. Templates are dependency subtrees which may contain POS-tagged *variables*, matching any lemma. Figure 4.1(b) shows passive-to-active transformation rule, and (a) illustrates its application.

The rule application procedure is given in algorithm 1. Rule application generates a set D of derived trees (*consequents*) from a source tree s through the following steps:

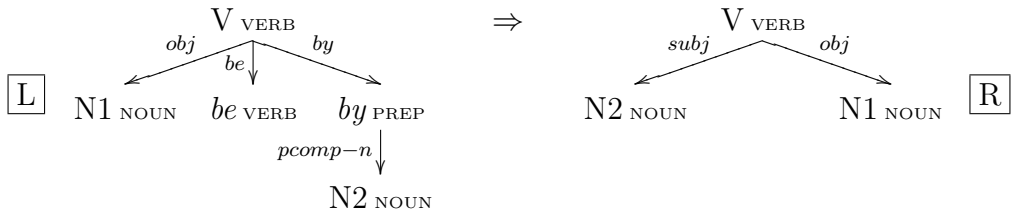


Source: it rained when beautiful Mary was seen by John yesterday



Derived: it rained when John saw beautiful Mary yesterday

(a) Passive-to-active tree transformation



(b) Passive to active substitution rule.

Figure 4.1: Application of an inference rule. POS and relation labels are based on Minipar (Lin, 1998). $N1$, $N2$ and V are variables, whose instances in L and R are implicitly aligned.

L matching: First, matches of L in the source tree s are sought. L is *matched* in s if there exists a one-to-one node mapping function f from L to s, such that:

1. For each node u in L, $f(u)$ has the same features and feature values as u. Variables match any lemma value in $f(u)$.

2. For each edge $u \rightarrow v$ in L, there is an edge $f(u) \rightarrow f(v)$ in s, with the same dependency relation.

If matching fails, the rule is not applicable to s. In our example, the variable V is matched in the verb *see*, $N1$ is matched in *Mary* and $N2$ is matched in *John*. If matching succeeds, then the following is performed for each match found.

R instantiation: a copy of R is generated and its variables are instantiated according to their matching node in L. In addition, a rule may specify *alignments*, defined as a partial function from L nodes to R nodes. An alignment indicates that for each modifier m of the source node that is not part of the rule structure, the subtree rooted at m should also be copied as a modifier of the target node. In addition to defining alignments explicitly, each variable in L is implicitly aligned to its counterpart in R. In our example, the alignment between the V nodes implies that *yesterday* (modifying *see*) should be copied to the generated sentence, and similarly *beautiful* (modifying *Mary*) is copied for $N1$.

Derived tree generation: Let r be the instantiated R, along with its descendants copied from L through alignment, and l be the subtree matched by L. The formalism has two methods for generating the derived tree d: *substitution* and *introduction*, as specified by the rule type. *Substitution* rules specify modification of a subtree of s, leaving the rest of s unchanged. Thus, d is formed by copying s while replacing l (and the descendants of l's nodes) with r. This is the case for the passive rule, as well as

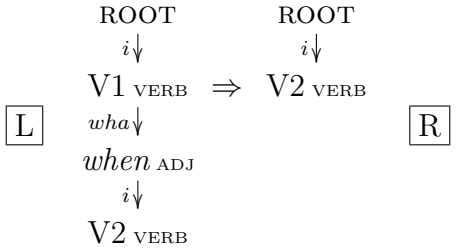


Figure 4.2: Temporal clausal modifier extraction (introduction rule)

for lexical rules such as '*buy* → *purchase*'. By contrast, *introduction* rules are used to make inferences from a subtree of s, while the other parts of s are ignored and do not affect d. A typical example is inferring a proposition embedded as a relative clause in s. In this case, the derived tree d is simply taken to be r. Figure 4.2 presents such a rule which enables to derive propositions that are embedded within temporal modifiers. Note that the derived tree does not depend on the main clause. Applying this rule to the right part of Figure 4.1(a) yields the proposition *John saw beautiful Mary yesterday.*

4.4 Further Examples for Rule Application

In this section we further illustrate rule representation and application through some additional examples.

Lexical substitution rule with explicit alignment Figure 4.3 shows the derivation of the consequent *"John purchased books."* from the sentence *"John bought books."* using the lexical substitution rule '*buy* → *purchase*'. This example illustrates the role of explicit alignment: since *buy* and *purchase* are not variables, they are not implicitly aligned. However, they need to be aligned explicitly, otherwise the daughters of *buy* would not be copied under *purchase*.

Input: a source tree s ; a rule $E : L \rightarrow R$
Output: a set D of derived trees

$M \leftarrow$ the set of all matches of L in s
$D \leftarrow \emptyset$
for each $f \in M$ **do**
  $l \leftarrow$ the subtree matched by L in s according to match f

  // R instantiation
  $r \leftarrow$ a copy of R
  for each variable $v \in r$ **do**
    Instantiate v with $f(v)$
  for each aligned pair of nodes $u_L \in l$ and $u_R \in r$ **do**
    for each daughter m of u_L such that $m \notin l$ **do**
      Copy the subtree of s rooted in m under u_R in r, with the same dependency relation

  // Derived tree generation
  if *substitution* rule **then**
    $d \leftarrow s$ copy with l (and the descendants of its nodes) replaced by r
  else // *introduction* rule
    $d \leftarrow r$

  add d to D

Algorithm 1: Applying a rule to a tree

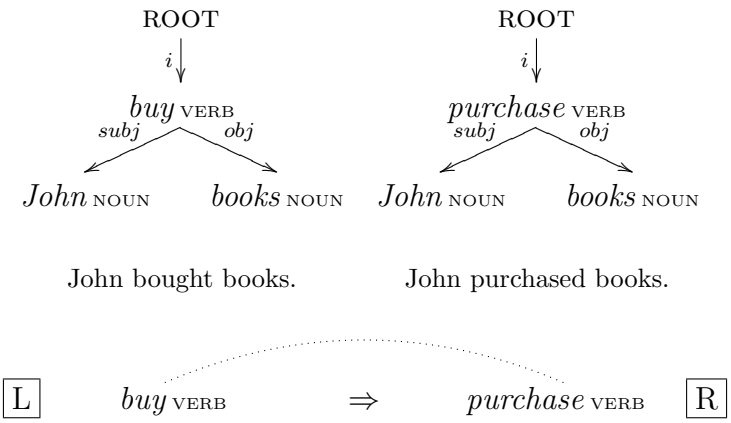
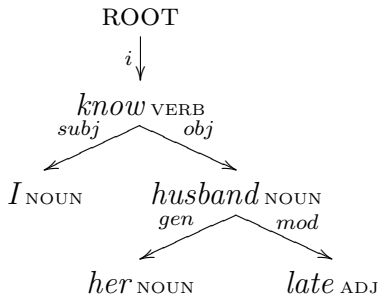
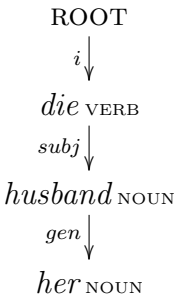


Figure 4.3: Application of a lexical substitution rule. The dotted arc represents explicit alignment.

Lexical-Syntactic introduction rule Figure 4.4 illustrates the application of a lexical-syntactic rule, which derives the sentence *"Her husband died."* from *"I knew her late husband."*. It is defined as introduction rule, since the resulting tree is derived based solely on the phrase *"Her late husband"*, while ignoring the rest of the source tree. This example illustrates that a leaf variable in L (variable at a leaf node) may become a non-leaf in R and vice versa. The alignment between the instances of variable N (matched in *husband*) allows copying of its modifier, *her*. We note here that the correctness of rule application may depend on the context in which it is applied. For instance, N in our example should be animated. Recently, a context modeling framework for entailment rules has been proposed by (Szpektor et al., 2008), which can be easily integrated into our framework, although this was not attempted in the current work.



I knew her late husband.



Her husband died.

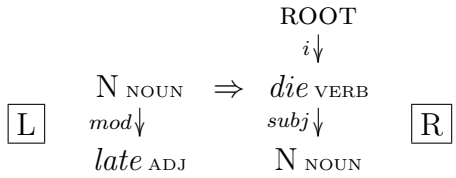


Figure 4.4: Application of a lexical-syntactic introduction rule.

4.5 Co-Reference and Trace-Based Inference

Besides the primary inference mechanism of rule application, our formalism also allows inference based on co-reference relations and long-distance dependencies. We view co-reference as an equivalence relation between complete subtrees, either within the same tree or in different trees, which are linked by a co-reference chain. In practice, such relations are obtained from an external co-reference resolution tool, as part of the text pre-processing. The *co-reference substitution* operation is similar to application of a substitution rule. Given a pair of co-referring subtrees, t_1 and t_2, the derived tree is generated by copying the tree containing t_1, while replacing t_1 with t_2 (the same operation is symmetrically applicable for t_2). If t_1 is a subtree of t_2, we exclude from the substituted copy of t_2 all the nodes dominating t_1 (except for t_2's root) and their descendants. For example, given the sentences *"[My brother] is a musician. [He]*

plays the drums", we can infer that *"My brother plays the drums"*.

Another type of useful relations for inference are long-distance dependencies as illustrated in the following examples:

(1) *Relative clause:* The boy$_i$ whom [I saw t$_i$] went home
(⇒ I saw the boy)

(2) *Control verbs:* John$_i$ managed to [t$_i$ open the door]
(⇒ John opened the door).

(3) *Verbal conjunction:* [John$_i$ sang] and [t$_i$ danced]
(⇒ John danced).

Some parsers, including Minipar which we use in the current work, recognize and annotate such long distance dependencies. For instance, Minipar generates a node representing the trace (t$_i$ in the examples), which holds a pointer to its antecedent (e.g. *John$_i$* in (2)). As shown in these examples, inference from such sentences may involve resolving long distance dependencies, where traces are substituted with their antecedent. Thus, we can generalize co-reference substitution to operate over trace-antecedent pairs as well. This mechanism works together with entailment rule application. For instance, after substituting the trace with its antecedent in (2) we obtain *"John managed to [John opened the door]"*. Then, we apply the introduction rule *'N managed to S → S'* to extract the embedded clause *"John opened the door"*.

4.6 Polarity Annotation Rules

In addition to inference rules, our formalism implementation includes a mechanism for adding semantic features to parse tree nodes. However, in many cases there is no natural way to define semantic features or classes, and hence it is often difficult to agree on the "right" set of semantic annotations (a common example is the definition of

word senses). Hence, with our approach we aim to keep semantic annotation to minimum, while sticking to lexical-syntactic representation, for which widely-agreeable schemes do exist.

Consequently, the only semantic annotation we employ is predicate *polarity*. This feature marks the truth of a predicate, and may take one of the following values: *positive*(+), *negative*(-) or *unknown(?)*. Some examples of polarity annotation are shown below:

(4) John called$^{[+]}$ Mary.

(5) John *hasn't* called$^{[-]}$ Mary yet.

(6) John *forgot* to call$^{[-]}$ Mary.

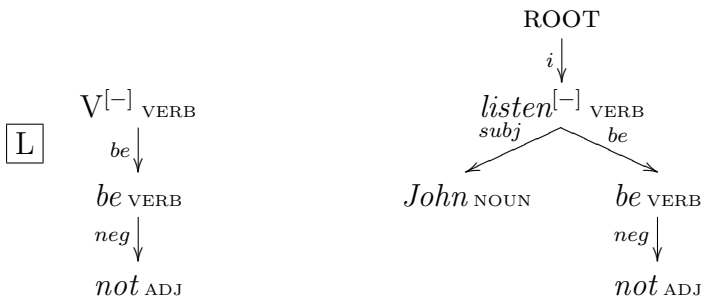
(7) John *might have* called$^{[?]}$ Mary.

(8) John *wanted* to call$^{[?]}$ Mary.

Sentences (5) and (6) both entail *"John didn't call Mary."*, hence the negative annotation of *call*. By contrast, the truth of *"John called Mary."* cannot be determined from (7) and (8), therefore the predicate *call* is marked as *unknown*. In general, the polarity of predicates may be affected by the existence of modals, negation, conditionals, certain verbs etc. Our polarity rule base, which addresses various polarity-affecting contexts, is described in section 6.4.

Technically, annotation rules do not have a right-hand-side R, but rather each node of L may contain annotation features. If L is matched in a tree then the annotations it contains are copied to the matched nodes. Figure 4.5 shows an example of annotation rule application.

Predicates are assumed to have positive polarity by default, and the polarity rules are used to mark negative or unknown polarity. If more than one rule applies to



(a) Annotation rule (b) Annotated sentence

Figure 4.5: Application of the annotation rule (a), marking the predicate *listen* with negative polarity (b) .

the same predicate (as with the sentence *"John forgot not to call Mary"*), they may be applied in any order, and the following simple calculus is employed to combine current polarity with new polarity:

Current polarity	New polarity	Result
+	−	−
−	−	+
?	−	?
+/ − /?	?	?

Notice that this mechanism is local, and does not handle polarity propagation from the main clause into nested embedded clauses, as in (9). A simple polarity propagation algorithm is described in (Nairn et al., 2006).

(9) Mary admitted that she did not fail to avoid meeting John. ⇒ Mary did not meet John.

The annotation rules are used for detecting polarity mismatches between the text and the hypothesis. Incompatible polarity would block the hypothesis from being

matched in the text. In the case of approximate entailment classification, polarity mismatches detected by the annotation rules are used as features for the classifier, as we discuss further in Section 7.4.

4.7 The Inference Process

Let T be a set of dependency trees representing the text, along with co-reference and trace information. Let h be the dependency tree representing the hypothesis, and let $\mathcal{R}$ be a collection of entailment rules (including both inference and polarity rules). Based on the previously-defined components of our inference framework, we next give a procedural definition for the set of trees inferrable from T using $\mathcal{R}$, denoted $\mathcal{I}(T, \mathcal{R})$. The inference process comprises the following steps:

1. Initialize $\mathcal{I}(T, \mathcal{R})$ with T.

2. Apply all matching polarity rules in $\mathcal{R}$ to each of the trees in $\mathcal{I}(T, \mathcal{R})$ (cf. sec. 4.6).

3. Replace all the trace nodes with a copy of their antecedent subtree (cf. sec. 4.5).

4. Add to $\mathcal{I}(T, \mathcal{R})$ all the trees derivable by co-reference substitution (cf. sec. 4.5).

5. Apply all matching inference rules in $\mathcal{R}$ to the trees in $\mathcal{I}(T, \mathcal{R})$ (cf. sec. 4.3), and add the derived trees to $\mathcal{I}(T, \mathcal{R})$. Repeat this step iteratively for the newly added trees, until no new trees are added.

Steps 2 and 3 are performed for h as well. h is inferrable from T using $\mathcal{R}$ if $h \in \mathcal{I}(T, \mathcal{R})$. Since $\mathcal{I}(T, \mathcal{R})$ may be infinite or very large, practical implementation of this process must limit the search space, e.g. by restricting the number of iterations and the applied rules at each iteration.

When an inference rule is applied, polarity annotation is propagated from the source tree s to the derived tree d as follows. First, nodes copied from s to d retain their original polarity. Second, a node in d gets the polarity of its aligned node in s.

4.8 Template Hypotheses

For many applications it is useful to allow the hypothesis h to be a template rather than a proposition, that is, to contain variables. The variables in this case are existentially quantified: t entails h if there exists a proposition h', obtained from h by variable instantiation, so that t entails h'. Each variable X is instantiated (replaced) with a subtree S_X. If X has modifiers in h (i.e. X is not a leaf), they become modifiers of S_X's root. The obtained variable instantiations may stand for sought answers in questions or slots to be filled in relation extraction. For example, applying this framework in a question-answering setting, the question *Who killed Kennedy?* may be transformed into the hypothesis *X killed Kennedy*. A successful proof of h from the sentence *"The assassination of Kennedy by Oswald shook the nation"* would instantiate X with *Oswald*, providing the sought answer. Note that h' is an instantiation of h if and only if the result of applying the introduction rule '$h \rightarrow h$' to h' is exactly h'.

4.9 Summary

This chapter presented a well-formalized approach for semantic inference over parse-based representations, which is the core of this thesis. In our framework, semantic knowledge is represented uniformly as entailment rules specifying tree transformations. We provided detailed definitions for the representation of these rules as well as the inference mechanisms that apply them. Our formalism also models inferences based on co-reference relations and traces. In addition, it includes *annotation* rules

that are used to detect contexts affecting the polarity of predicates. In the next chapter we present an efficient implementation of this formalism. The expressiveness of our formalism is illustrated in chapter 6, where it is used for the development of an entailment rule base for generic linguistic phenomena.

Chapter 5

A Compact Forest for Scalable Inference

5.1 Introduction

Chapter 4 introduced a generic formalism for semantic inference over parse trees. Entailment rules are used as a unifying representation for various types of inference knowledge, allowing unified inference as well. In our formalism, each rule application generates a new sentence parse (a *consequent*), semantically entailed by the source sentence. The inferred consequent may be subject to further rule applications and so on.

A straightforward implementation of this formalism would generate each consequent as a separate tree. Unfortunately, this naïve approach raises severe efficiency issues, since the number of consequents may grow exponentially in the number of rule applications. Consider, for example, the sentence *"Children are fond of candies."*, and the following entailment rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*. The number of derivable sentences (including the source sentence)

would be 2^3 (the power set size), as each rule can either be applied or not, independently. Indeed, we found that this exponential explosion leads to poor scalability of the naïve implementation approach in practice. Intuitively, we would like that each rule application would add just the entailed part of the rule (e.g. *kids*) to a packed sentence representation. Yet, we still want the resulting structure to represent a set of entailed sentences, rather than some mixture of sentence fragments whose semantics is unclear. As discussed in chapter 9, previous work proposed only partial solutions to this problem.

In this chapter we present a novel data structure, termed *compact forest*, and a corresponding inference algorithm, which efficiently generate and represent all consequents while preserving the identity of each individual one. This data structure allows compact representation of a large set of inferred trees. Each rule application generates explicitly only the nodes of the rule right-hand-side while the rest of the consequent tree is shared with the source sentence, which also reduces the number of redundant rule applications. As we shall see, this representation is based primarily on *disjunction edges*, an extension of dependency edges that specify a set of alternative edges of multiple trees.

Our work is inspired by previous work on packed representations in other fields, such as parsing, generation and machine translation, which we survey in chapter 9. As we follow a well-defined inference formalism, we could prove that all inference operations in our formalism are equivalently applied over the compact forest. We compare inference cost over compact forests to explicit consequent generation both theoretically, illustrating an exponential-to-linear complexity ratio, and empirically, showing improvement by orders of magnitude. These results suggest that our data-structure and algorithm are both valid and scalable, opening up the possibility to investigate large-scale entailment rule application within a well-formalized framework.

The rest of this chapter is organized as follows: section 5.2 introduces the compact forest data structure, section 5.3 presents an efficient algorithm for inference over compact forests, and sections 5.4–5.5 discuss its correctness and complexity. Empirical evaluation of the compact forest is described in chapter 8.

5.2 The Compact Forest Data Structure

A compact forest $\mathcal{F}$ represents a set of dependency trees. Figure 5.1 shows an example of a compact forest, containing both the source and derived sentences of Figure 4.1. We first define a more general data structure for directed graphs, and then narrow the definition to the case of trees.

A *Compact Directed Graph (cDG)* is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of **disjunction edges** *(d-edges)*. Let $\mathcal{D}$ be a set of dependency relations. A d-edge d is a triple (S_d, rel_d, T_d), where S_d and T_d are disjoint sets of source nodes and target nodes; $rel_d : S_d \rightarrow \mathcal{D}$ is a function specifying the dependency relation corresponding to each source node. Graphically, d-edges are shown as point nodes, with incoming edges from source nodes and outgoing edges to target nodes. For instance, let d be the bottommost d-edge in Figure 5.2. Then $S_d = \{$*of, like*$\}$, $T_d = \{$*candy, sweet*$\}$, $rel($*of*$)$ = *pcomp-n*, and $rel($*like*$)$ = *obj*.

A d-edge represents, for each $s_i \in S_d$, a set of alternative directed edges $\{(s_i, t_j) : t_j \in T_d\}$, all of which are labeled with the same relation given by $rel_d(s_i)$. Each of these edges, termed *embedded edge (e-edge)*, would correspond to a different graph represented in $\mathcal{G}$. In the previous example, the e-edges are *like* $\xrightarrow{obj}$ *candy*, *like* $\xrightarrow{obj}$ *sweet*, *of* $\xrightarrow{pcomp-n}$ *candy* and *of* $\xrightarrow{pcomp-n}$ *sweet* (notice that the definition implies that all source nodes in S_d have the same set of alternative target nodes T_d). d is called an *outgoing d-edge* of a node v if $v \in S_d$ and an *incoming d-edge* of v if $v \in T_d$. A *Compact Directed Acyclic Graph (cDAG)* is a cDG that contains no cycles of e-edges.

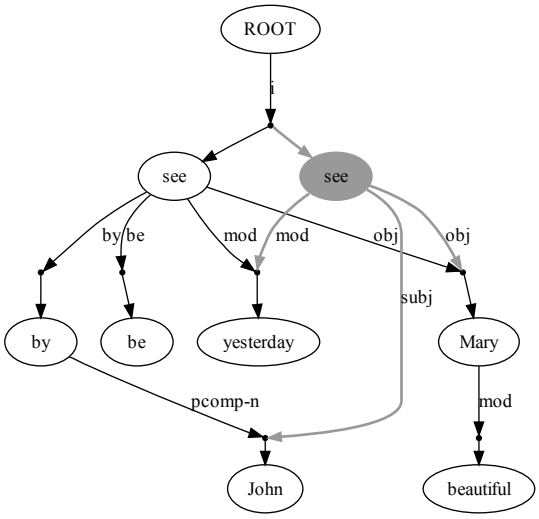


Figure 5.1: A compact forest containing both the source and derived sentences of Figure 4.1. Parts of speech are omitted.

A DAG G rooted in a node $v \in \mathcal{V}$ of a cDAG $\mathcal{G}$ is *embedded* in $\mathcal{G}$ if it can be derived as follows: we initialize G with v alone; then, we *expand* v by choosing exactly one target node $t \in T_d$ from each outgoing d-edge d of v, and adding t and the corresponding e-edge (v, t) to G. This expansion process is repeated recursively for each new node added to G.

Each such set of choices results in a different DAG with v as its only root. In Figure 5.1, we may choose to connect the root either to the left *see*, resulting in the source passive sentence, or to the right *see*, resulting in the derived active sentence.

A **Compact Forest** $\mathcal{F}$ is a cDAG with a single root r (i.e. r has no incoming d-edges) where all the embedded DAGs rooted in r are trees. This set of trees, termed *embedded trees*, and denoted $\mathcal{T}(\mathcal{F})$comprise the set of trees represented by $\mathcal{F}$.

Figure 5.2 shows another example for a compact forest efficiently representing the 2^3 sentences resulting from three independently-applied rules (cf. section 5.1).

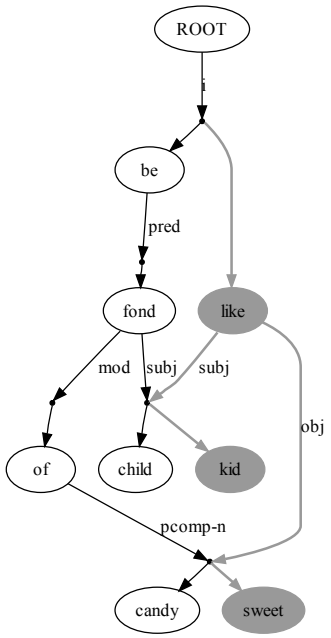


Figure 5.2: A compact forest representing the 2^3 sentences derivable from the sentence *"children are fond of candies"* using the following three rules: *'children→kids'*, *'candies→sweets'*, and *'X is fond of Y→X likes Y'*.

5.3 The Inference Process

Next, we describe the algorithm implementing the inference process described in section 4.7 over the compact forest (henceforth, *compact inference*), illustrating it through Figures 4.1 and 5.1.

Forest initialization $\mathcal{F}$ is initialized with the set of dependency trees representing the *text* sentences, with their roots connected under the forest root as the target nodes of a single d-edge. Dependency edges are transformed trivially to d-edges with a single source and target. Annotation rules are applied at this stage to the initial $\mathcal{F}$. The black part of Figure 5.1 corresponds to the initial forest (containing a single

Input: a compact forest $\mathcal{F}$; an inference rule $E : L \rightarrow R$
Output: A modified $\mathcal{F}$, denoted $\mathcal{F}'$, such that $\mathcal{T}(\mathcal{F}') = \mathcal{T}(\mathcal{F}) \cup D$, where D is the set of trees derived by applying E for any subset of L's matches in each of the trees in $\mathcal{T}(\mathcal{F})$

// L matching
$M \leftarrow$ the set of all matches of L in $\mathcal{F}$
for each match $f \in M$ **do**
  $l \leftarrow$ the subtree of $\mathcal{F}$ in which L is matched according to f
  $S_L \leftarrow l$ excluding *dual leaf variable* nodes
  $r_L \leftarrow root(l)$

  // Right-hand-side generation
  $S_R \leftarrow$ copy of R excluding *dual leaf variable* nodes
  $r_R \leftarrow root(S_R)$
  Add S_R to $\mathcal{F}$
  if E is a *substitutison* rule **then**
    $d \leftarrow$ the incoming d-edge of r_L // will set S_R as an alternative to S_L
  else // *introduction* rule
    $d \leftarrow$ the outgoing d-edge of $root(\mathcal{F})$ // will set S_R as an alternative to other trees in $\mathcal{T}(\mathcal{F})$
  Add r_R to T_d

  // Variable instantiation
  for each variable X held in node $x_R \in S_R$ **do** // R's variables excluding dual leaves
    if X is not a leaf in L **then**
      $x_L \leftarrow f(X)$ // the node in S_L matched by X
      (x_R.lemma, x_R.polarity) $\leftarrow$ (x_L.lemma, x_L.polarity)

    else // X is a leaf in L so it is matched in the whole target node set
      (x_R.lemma, x_R.polarity) $\leftarrow$ (n.lemma, n.polarity) for some node $n \in f(X)$
      for each $n' \in f(X); n' \neq n$ **do**
        generate a substitution rule $n \rightarrow n'$ where n and n' are aligned, and apply it to x_R
        $x'_R \leftarrow$ the instantiation of n'
        for each $u \in S_L$ such that u is aligned to x_R **do**
          add alignment from u to x'_R

  // Alignment sharing
  for each aligned pair of nodes $n_L \in S_L$ and $n_R \in S_R$ **do**
    n_R.polarity $\leftarrow$ n_L.polarity
    for each outgoing d-edge d of n_L whose e-edges are not part of S_L **do**
      Add n_R to S_d
      $rel_d(n_R) \leftarrow rel_d(n_L)$

  // Dual leaf variable sharing
  for each dual-leaf variable X matched in a node $v \in l$ **do**
    $d \leftarrow$ the incoming d-edge of v
    $p \leftarrow$ parent node of X in S_R

    // go over p and alternatives for p generated during variable instantiation
    $P \leftarrow$ set of target nodes of p's incoming d-edge
    for each $p' \in P$ **do**
      Add p' to S_d
      $rel_d(p') \leftarrow$ relation between X and p

Algorithm 2: Applying an inference rule to a compact forest

sentence in our example).

Inference Rule application comprises the following steps, described below and as summarized in Algorithm 2:

L matching: L is matched in $\mathcal{F}$ if there exists an embedded tree t in $\mathcal{F}$ such that L is matched in t, as in section 4.3. We denote by l the subtree of t in which L was matched. This subtree may be shared by multiple trees represented in $\mathcal{F}$, and the rule is applied simultaneously for all these trees. As in section 4.3, the match in our example is $(V, N1, N2)$=*(see, Mary, John)*. Notice that this definition does not allow l to be scattered over multiple embedded trees. Matches are constructed incrementally, aiming to add L's nodes one by one (variable nodes are added last), while verifying for each candidate node in $\mathcal{F}$ that both node content and the corresponding edge labels match. It is also verified that the match does not contain more than one e-edge from each d-edge. The nodes in $\mathcal{F}$ are indexed using a hash table to allow fast lookup.

As the target nodes of a d-edge specify alternatives for the same position in the tree, their parts-of-speech are expected to be of substitutable types. We further assume that all target nodes of the same d-edge have the same part-of-speech[1] and polarity. Consequently, variables that are leaves in L and may match a certain target node of a d-edge d are mapped to the whole set of target nodes T_d rather than to a single node. This yields a compact representation of multiple matches, and prevents redundant rule applications. For instance, given a compact representation of *'{Children/kids} are fond of {candies/sweeets}'* (cf. Figure 5.2), the rule *'X is fond of Y→X likes Y'* will be matched and applied only once, rather than four times (for each combination of matching X and Y).

Right-hand-side generation: A template S_R consisting of R while excluding variables that are leaves of both L and R (termed *dual leaf-variables*), is generated and

[1]This is the case in our current implementation, which is based on the coarse tag-set of Minipar

inserted into $\mathcal{F}$. Variables which are the only node in R (and hence are both the root and a leaf), and variable with additional alignments (other than the implicit alignment between their occurrences in L and R) are not considered dual-leaves. Similarly, we define S_L as l excluding dual-leaf variables.

In the case of a substitution rule (as in our example), S_R is set as an alternative to S_L by adding r's root to T_d, where d is the incoming d-edge of S_L's root. In case of an introduction rule, it is set as an alternative to the other trees in the forest by adding r's root to the target node set of the forest root's outgoing d-edge. In our example, S_R is the gray node (still labeled with the variable V) , and it becomes an additional target node of the d-edge entering the original (left) *see*.

Variable instantiation: Each variable in S_R (i.e. a non-dual leaf) is instantiated according to its match in L (as in Section 4.3), e.g. V is instantiated with *see*. As specified above, if the variable is a leaf in L then it is matched in a set of nodes, and hence each of them should be instantiated in S_R. This is decomposed into a sequence of simpler operations: first, S_R is instantiated with a representative from the set, and then we apply (ad-hoc) lexical substitution rules for creating a new node for each other node in the set. Notice that these nodes, in addition to the usual alignment with their source nodes in S_L, share the same daughters in S_R.

Alignment sharing: Modifiers of aligned nodes are shared (rather than copied) as follows. Given a node n_L in l aligned to a node n_R in r, and an outgoing d-edge d of n_L which is not part of l, we share d between n_L and n_R by adding n_R to S_d and setting $rel_d(n_R) = rel_d(n_L)$. This is illustrated by the sharing of *yesterday* in Figure 5.1. We also copy polarity annotation from n_L to n_R.

We note at this point that the instantiation of variables that are not dual leaves cannot be shared because they typically have different modifiers at the two sides of the rule. Yet, their modifiers which are not part of the rule are shared through the alignment operation (recall that common variables are always considered aligned).

Dual leaf variables, on the other hand, might be shared, as described next, since the rule doesn't specify any modifiers for them.

Dual leaf variable sharing: This final step is performed analogously to alignment sharing. Suppose that a dual leaf variable X is matched in a node v in l whose incoming d-edge is d. Then we simply add the parent p of X in r to S_d and set $rel_d(p)$ to the relation between p and X (in R). Since v itself is shared, its modifiers become shared as well, implicitly implementing the alignment operation. The subtrees *beautiful Mary* and *John* are shared this way for variables $N1$ and $N2$. If ad-hoc substitution rules were applied to p at the variable instantiation phase, the generated nodes serve as alternative parents of X and thus the sharing procedure applied to p should be repeated for each of them.

Applying the rule in our example added only a single node and linked it to four d-edges, compared to duplicating the whole tree in explicit inference.

Co-reference Substitution In section 4.5 we defined *co-reference substitution*, an inference operation that allows replacing a subtree t_1 with a co-referring subtree t_2. This operation is implemented by generating on-the-fly a substitution rule $t_1 \rightarrow t_2$ and applying it to t_1. In our implementation, the initial compact forest is annotated with co-reference relations obtained from an external co-reference resolution tool, and all substitutions are performed prior to rule applications. Substitutions where t_2 is a pronoun are ignored, as they do not seem useful inferences.

5.4 Correctness

In this section we present two theorems that prove that the inference process presented is a valid implementation of the inference formalism. We sketch the proof outlines here and give the full proofs in Appendix A.

We first argue in theorem 1 that performing any sequence of rule applications over the set of initial trees results in a compact forest. Notice that the fact that the embedded DAGs generated during the inference process are indeed trees is not trivial, since nodes generally have many incoming e-edges from many nodes. However, it can be shown that any pair of these parent nodes cannot be part of the same embedded DAG. For example, in figure 5.2, the node *'candies'* has an incoming e-edge from both the node *'like'* and the node *'of'*. However, the nodes *'like'* and *'of'* are not part of the same embedded DAG. This is because of the d-edge emanating from the root that forces us to choose between the node *'like'* and the node *'are'*. Thus, we see that the reason for correctness is not local: the two incoming e-edges into the leaf node *'candies'* cannot be in the same embedded DAG because of a rule applied at the root of the tree. We now turn to the theorem and its proof scheme:

Theorem 1 *The inference process generates a compact forest.*

Proof scheme We prove by induction on the number of rule applications. Initialization generates a single-rooted cDAG, whose embedded DAGs are all trees, as required. We then prove that if applying a rule on a compact forest creates a cycle or an embedded DAG that is not a tree, then such a cycle or a non-tree DAG already existed prior to rule application, in contradiction with the inductive assumption. A crucial observation for this proof is that for any directed path from a node u to a node v that passes through S_R, where u and v are outside S_R, there is also an analogous path from u to v that passes through S_L instead. ∎

The next theorem is the main result. We argue that the inference process over a compact forest is complete and sound, i.e., it generates exactly the set of consequents derivable from a text according to the inference formalism.

Theorem 2 *Given a rule base $\mathcal{R}$ and a set of initial trees T, a tree t is represented by a compact forest derivable from T by the inference process $\Leftrightarrow$ t is a consequent of T according to the inference formalism*

Proof scheme We first show completeness by induction on the number of explicit rule applications. Let t_{n+1} be a tree derived from a tree t_n using the rule r_n according to the inference formalism. The inductive assumption determines that t_n is embedded in some derivable compact forest $\mathcal{F}$. It is easy to verify that applying r_n on $\mathcal{F}$ will yield a compact forest $\mathcal{F}'$ in which t_{n+1} is embedded.

Next, we show soundness by induction on the number of rule applications over the compact forest. Let t_{n+1} be a tree represented in some derived compact forest $\mathcal{F}_{n+1}$ ($t_{n+1} \in \mathcal{T}(\mathcal{F}_{n+1})$). $\mathcal{F}_{n+1}$ was derived from the compact forest $\mathcal{F}_n$, using the rule r_n. The inductive assertion states that all the trees in $\mathcal{T}(\mathcal{F}_n)$ are consequents according to the formalism. Hence, if t_{n+1} is already in $\mathcal{T}(\mathcal{F}_n)$ then it is a consequent. Otherwise, it can be shown that there exists a tree $t_n \in \mathcal{T}(\mathcal{F}_n)$ such that applying r_n to t_n will yield t_{n+1} according to the formalism. t_n is a consequent according to the inductive assertion and therefore t_{n+1} is a consequent as well. ∎

These two theorems guarantee that the compact inference process is valid - i.e., it yields a compact forest that represents exactly the set of consequents derivable from a given text by a given rule set.

5.5 Complexity

In this section we explain why compact inference exponentially reduces the time and space complexity in typical scenarios.

We consider a set of rule matches in a tree T *independent* if their matched left-hand-sides (excluding dual-leaf variables) do not overlap in T, and their application

over T can be chained in any order. For example, the three rule matches presented in figure 5.2 are independent.

Let us consider explicit inference first. Assume we start with a single tree T with k independent rules matched. Applying k rules will yield 2^k trees, since any subset of the rules might be applied to T. Therefore, the time and space complexity of applying k independent rule matches is $\Omega(2^k)$. Applying more rules on the newly derived consequents behaves in a similar manner.

Next, we examine compact inference. Applying a rule using compact inference adds the right-hand-side of the rule and shares with it existing d-edges. Since that the size of the right-hand-side and the number of outgoing d-edges per node are practically bounded by low constants, applying k rules on a tree T yields a linear increase in the size of the forest. Thus, the resulting size is $O(|T| + k)$, as we can see from Figure 5.2.

The time complexity of rule application is composed of matching the rule in the forest and applying the matched rule. Applying a matched rule is linear in its size. Matching a rule of size r in a forest $\mathcal{F}$ takes $O(|\mathcal{F}|^r)$ time even when performing an exhaustive search for matches in the forest. Since r tends to be quite small and can be bounded by a low constant, this already gives polynomial time complexity. In practice, indexing the forest nodes, as well as the typical low connectivity of the forest, result in a very fast matching procedure, as illustrated in the empirical evaluation, described in chapter 8.

5.6 Summary

In this chapter we addressed the efficiency of entailment rule application. We presented a novel compact data structure and a rule application algorithm for it, which

are provably a valid implementation of our inference formalism. We examined inference efficiency analytically, and in chapter 8 it will be assessed empirically as well. Beyond entailment inference, we suggest that the compact forest may also be useful in generation tasks, such as paraphrasing. Our efficient representation of the consequent search space opens the way to future investigation of the benefit of larger-scale rule chaining, and to the development of efficient search strategies required to support such inferences.

Chapter 6

A Generic Entailment Rule Base[1]

6.1 Introduction

Generic linguistic phenomena play a crucial role in entailment inference. Examples include coordination and subordination, relative clauses and appositions, negation and modality, conditionals, control verbs and so on. Their importance emerged from our RTE data analysis (presented in chapter 3), as well as from previous work that analyzed this dataset (Vanderwende and Dolan, 2006).

Although several entailment systems have addressed these phenomena to some extent, no comprehensive rule base for such phenomena has been made available to date. Based on our formalism (introduced in Chapter 4), this chapter describes the development of the first comprehensive, publicly available, entailment rule base addressing generic linguistic structures. While lexical and lexical-syntactic entailments amount to millions of rules, these generic phenomena could be well covered by less than a hundred rules[2]. Thus, it was affordable to compose this rule base manually, which allowed us to fine-tune it for both accuracy and coverage.

[1] Joint work with Iddo Greental.

[2] As we shall see, some of these rules compactly encode multiple lexical and structural variations.

Essentially, the development of such a rule base comprises two stages. First, identifying linguistic phenomena that are relevant for entailment inference. Second, writing concrete entailment rules in our formalism for these phenomena. Since the formalism is parse-based, and its implementation relies on parser-specific output, robust implementation of the rule base must take into account the various ways the parser might represent these linguistic structures.

Accordingly, this chapter makes two contributions: first, we present a taxonomy of linguistic phenomena relevant for entailment (sections 6.2–6.4), and illustrate their implementation for a specific parser (Minipar, in our case). We then describe our methodology for constucting a robust, parser-specific rule base for given linguistic phenomena (section 6.5). This methodology may guide porting our rule base to other parsers, or to other languages.

6.2 Rulebase Overview

6.2.1 Rule Types

Following our formalism, the rule base consists of two types of rules: *inference rules*, whose application generates a new tree, entailed by the source tree, and *polarity annotation rules*, marking the truth of predicates in existing trees (as positive, negative or unknown).

Both inference rules and polarity rules are divided into two subtypes: *generic* rules and *lexicalized* rules. We illustrate the difference between these two classes through the following examples. First, consider sentences (1)-(4):

(1) She wasn't contradicted by Mr. Johnson, the defense attorney, while she gave testimony.

(2) The defense attorney did not contradict her while she gave testimony.

(3) She gave testimony.

(4) Mr. Johnson is a defense attorney.

Sentence (1) entails (2), (3) and (4). These entailments can all be reached by relying on the syntactic structure of (1), without reference to the lexical or semantic properties of individual words. These entailments utilize the following syntactic features of sentence (1):

- Contracted negation (wasn't)
- Active and passive sentence (wasn't contradicted)
- Apposition (Mr. Johnson, the defense attorney)
- Nominative and accusative case marking (contradict her)
- Clausal modifiers (while)

Clearly, all of the above are well-established syntactic phenomena in English. Their behavior is well understood and covered in the literature. Consequently, they are fairly well handled by syntactic parsers. Such general phenomena, which depend on syntactic structure and closed-class words form the basis of our *generic* syntactic rule set. Now, consider sentences (5)-(7):

(5) Thankfully, he has already finished his work.

(6) Hopefully, he has already finished his work.

(7) He has already finished his work.

(5) entails (7) while (6) does not, although (5) and (6) are syntactically equivalent. The key difference between (5) and (6) is the semantic content of the initial adverb. These examples demonstrate a different class of entailment phenomena which combine

syntactic form and open-class lexicons which exhibit predictable semantic behavior. Our rule base contains a number of such rules that we refer to as *lexicalized rules*.

6.2.2 Rule Sources

Several sources were used to develop the rules. Firstly, we turned to the established literature on the syntax of English. Fundamental grammar books such as (Quirk et al., 1985; Baker, 1995) provided us with succinct formulation of the main generalizations in English syntax, as well as a rich source of complex hand-picked examples to be used later for testing. Secondly, we reviewed sentence pairs from the available RTE training datasets in order to identify valid entailments that rely on generalizable syntactic-based transformations. Rule base development was also affected by the information available from the parser. For example, Minipar dependency relations include *apposition* and *abbreviation*, enabling entailment rules that, for instance, replace a noun phrase with its appositive or with its abbreviation. We studied the parser output by parsing a large sample of sentences from a news corpus (Reuters), and searching the parse trees for common structures and dependency relations that are relevant for entailment inference. Finally, many of the lexicalized rules were collected from existing lexical resources, such as VerbNet (Kipper, 2005) and PARC's polarity lexicon (Nairn et al., 2006).

6.2.3 Scope

Our rule base focuses on linguistic phenomena that are common enough to have an impact on our entailment engine, and that can be modeled well within our formalism. Consequently, some of the most extensively investigated phenomena in the linguistic literature, such as quantification and scope ambiguity, are not fully addressed in our rule base. These phenomena often require more refined modeling. However, looking

at real-world data, we found that our rules provide fairly good coverage of common phenomena, while many of the more complex cases which we do not handle are often quite rare in practice. For some linguistic structures, such as coordination and quantification, we assume simple semantics that holds in the common case, and could be modeled within our formalism, and ignore the more complex (but typically less frequent) cases, whose modeling falls out of the scope of the current work.

The scope of the rules is also affected by the level of information available in the given dependency trees. Our rules correspond to typical dependency schemes of parsers such as Minipar (Lin, 1998) and the Stanford parser (de Marneffe et al., 2006), which include dependency relations for relative clauses, appositions, abbreviations etc.

Some of the rules are quite simple, and their implementation is straightforward. Other rules required careful compilation of lexicons and investigation of syntactic variations. Overall, the main contribution of this work is the collection and categorization of generic linguistic phenomena relevant for entailment, which resulted in a comprehensive wide-coverage entailment rule base.

6.2.4 Notes on Rule Implementation and Representation

As we shall see, many of the linguistic phenomena modeled in our rule base required multiple entailment rules for their implementation, due to linguistic variability and parser variations. For each such phenomenon described in the next sections (6.3–6.4) we specify the number of corresponding rules.

Rule format and notation: The sample rules shown in this chapter (e.g. the relative clause introduction rule in figure 6.1) are based on the Minipar scheme, with slight modifications[1]. Table 6.1 lists our part-of-speech (POS) tag set. Table 6.2 lists common Minipar relations, including those appearing in our examples. Minipar

[1]Minipar's VBE and *be* categories are conflated here with the VERB category.

POS Tag	Description
ADJ/ADV	adjective/adverb
AUX	auxiliary verb
CLAUSE	clause
DET	determiner
NOUN	noun
VERB	verb
PREP	preposition
OTHER	other

Table 6.1: POS tag set, adapted from Minipar's scheme.

generates artificial CLAUSE nodes as clause roots, with CLAUSE as the POS and lemma *'fin'* for finite clauses and *'inf'* for infinite clauses. The tree root has OTHER as the POS and an empty lemma. Variables are shown as ALL CAPS words. For example, X and C in figure 6.1 represent variables in the relative clause rule. In order to allow compact rule representation and simplify rule writing, we allowed in L multiple alternatives for the lemma at each node, and for the dependency relation at each edge, and adjusted the L matching procedure defined in section 5.3 accordingly. Alternatives are separated by a slash ('/'). Thus, each such "packed" rule may represent dozens of rule variations. This compact representation is demonstrated in figure 6.1.

The relative clause rule works as follows: L matches a noun phrase with a relative clause, where X matches the head of the noun phrase, and C matches the head of the relative clause. Since it is an introduction rule, the resulting tree is the instantiation of R. The implicit alignment between C in L and C in R would copy the relative clause subtree under the newly-generated tree. However, the relative pronoun (*who/that...*) is part of the rule structure and therefore will not be copied.

Relation	Description
abbrev	abbreviation
amod	adjectival/adverbial modifier
appo	apposition
aux	auxiliary verb
be	"be" modifier
c	clausal complement
det	determiner
gen	genitive noun modifier
i	the relationship between a main clause and a complement clause
lex-mod	lexical modifier
mod	adjunct modifier
neg	negation
nn	noun-noun modifier
obj	object of a verb
pcomp-n	nominal complement of prepositions
poss	possessive marker ('s)
pred	predicate of a clause
punc	punctuation
rel	relative clause
s	surface subject
sc	sentential complement
subj	subject of verbs
wha, whn, whp	wh-elements at C-spec positions

Table 6.2: Common Minipar relations

6.3 Inference Rules

Inference rules capture general syntactic transformations that consistently preserve meaning or produce valid entailment. The rule base exploits both the *substitution* and the *introduction* rule types defined in the formalism. The rule base contains 40 packed inference rules.

Substitution rules are typically used to derive either a canonical form of the source tree, or to derive a simplified entailed consequent, by replacing part of the source tree with an equivalent or entailed subtree, which is often a simplified version of

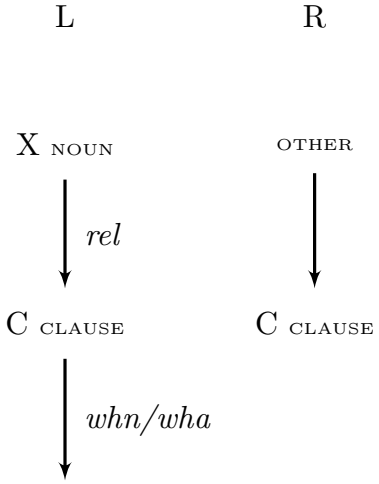


Figure 6.1: Relative clause extraction rule (*introduction*). This rule extracts the relative clause subtree rooted at the node matched by variable C to a new tree. The subtree rooted at C's match in the source tree is copied to the derived tree due to the implicit alignment between C in L and C in R. However, the relative pronoun (who/that...) is part of the rule structure and therefore will not be copied to the derived tree.

the matched subtree. Examples include passive-to-active transformation, possessive alternation (*'Spain's ⇒ of Spain'*), conjunctions such as *'John and Mary left ⇒ Mary left'* and apposition replacement, e.g. *'John, my brother, is a dentist ⇒ My brother is a dentist'*.

Introduction rules are used to extract propositions embedded within larger propositions when it is sound to do so, as in the case of clausal modification (cf. example (3) in the previous section). They are also used to infer propositions from non-propositional subtrees, such as apposition (see example (4) above).

Some rules are designed to interact with each other. For example, applying passive-to-active transformation to (8) would result in the ungrammatical (9).

(8) He was surprised by their visit.

(9) *Their visit surprised *he*.

A subsequent rule application would then detect a *he* in an object position and correct it to *him*, resulting in (10).

(10) Their visit surprised *him*.

We note that the inference process aims to generate a target hypothesis parse, which is assumed to be grammatical. Thus, intermediate ungrammatical consequents such as (9) would not match any valid hypothesis. We shall see some more examples for rule interactions later in this section.

As described in section 4.5, entailment rules may also interact with inferences based on trace marking. Minipar's trace marking for relative clauses, control verbs and verbal conjunction simplified the implementation of relevant inference rules. Consider again the relative clause example in section 4.5:

(11) The boy$_i$ whom [I saw t$_i$] went home.

Substitution of the trace with its antecedent resolves the object of *saw*. In the resulting sentence, the relative clause is expanded into a complete embedded proposition, ready to be extracted by the relative clause introduction rule.

(12) The boy whom [I saw *the boy*] went home.

In the remainder of this section, we list the concrete linguistic phenomena covered by our current set of inference rules, and their classification as either substitution or introduction rules, and illustrate their implementation for the Minipar scheme.

6.3.1 Generic Inference Rules

Passivization (*substitution*)

Passives with normal form are usually assumed to be semantically equivalent to their active counterpart, as long as the subject and object roles are maintained. We use

rules to transform passive sentences with 'by phrase' into their active counterpart. The passive rule was shown in section 4.3.
Example:

(13) The cake was eaten by John. ⇒ John ate the cake.

Apposition and Abbreviation (*1 introduction, 2 substitution*)

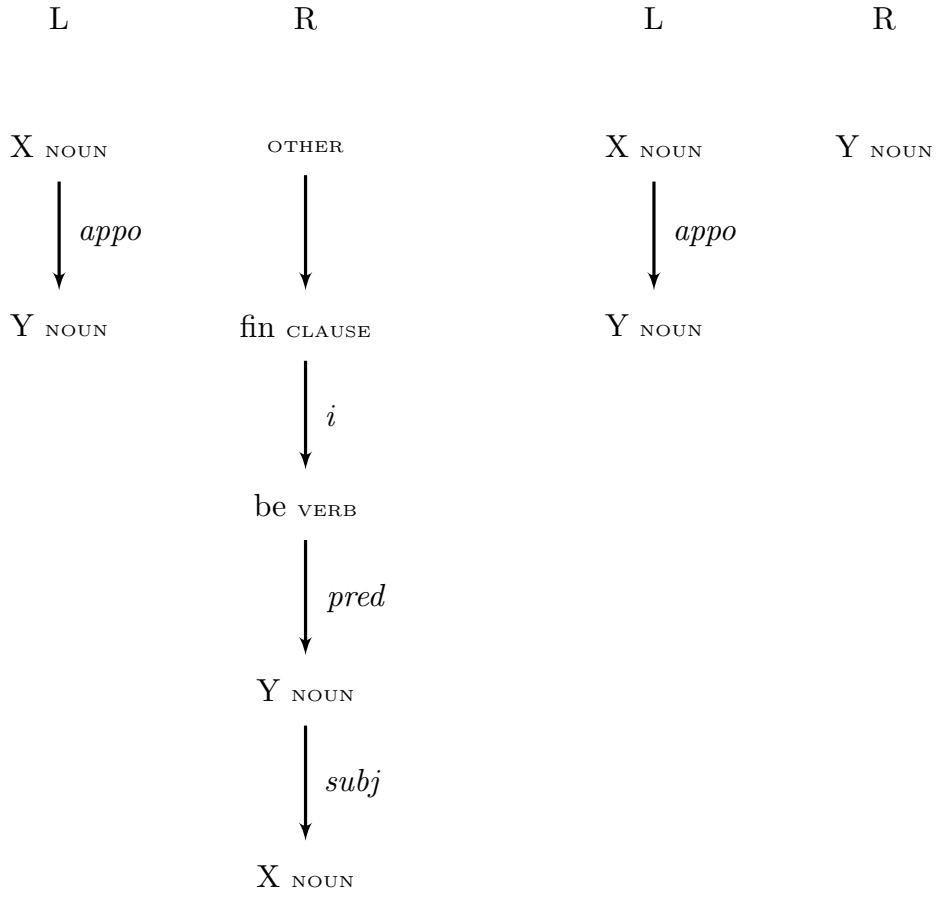
Semantically, a nominal apposition relation asserts an equivalence between a noun and its apposition. We use an introduction rule to extract the noun and apposition to a stand-alone copula (14), and a substitution rule to replace the noun with the apposition (15).
Examples:

(14) Superman, the Man of Steel, saved the world once again. ⇒ Superman is the Man of Steel.

(15) Superman, the Man of Steel, saved the world once again. ⇒ The Man of Steel saved the world once again.

The implementation of these two rules is shown in figure 6.2. It exploits Minipar's dependency structure for the apposition relation. The same relation holds between abbreviations and the expression they stand for. In (16) we can replace *World Health Organization* with *WHO* and *United Nations* with *UN*. Thus, we also define an abbreviation replacement rule, analogously to the apposition replacement rule. Its implementation utilizes Minipar's "abbrev" relation.

(16) The World Health Organization (WHO) is a specialized agency of the United Nations. (UN)



(a) Apposition to copula (*introduction*) Apposition replacement (*substitution*)

Figure 6.2: Inference rules for treating apposition. Given a noun phrase of the type *'X,Y'*, representing an apposition relation, (a) would entail a sentence of the form *'X is a Y'*, while (b) would replace *'X,Y'* with *'Y'*.

"and" conjunction (*substitution*)

While conjunctions may have complex semantics, in its simple use, a conjunction can be replaced with each of its conjuncts. We handle conjunctions at various levels: sentence-level conjunctions as well as subsentential conjunction of noun phrases (NP), verb phrases (VP), and adjectival and adverbial phrases.

Examples:

(17) George is steering the boat and Harris is pulling the oars. ⇒ Harris is pulling the oars.

(18) John is smart and handsome. ⇒ John is handsome.

Note: This type of operation is sometimes called *Conjunction Reduction*, and would produce valid entailments in most cases. However, in some situations the coordination semantics is more complex and this rule would derive incorrect consequents, as illustrated in the next examples:

(19) John and Mary are a nice couple. ⇒ *Mary is a nice couple.

(20) John and Mary like each other. ⇒ *Mary like each other.

Recognizing and addressing these situations falls beyond the scope of our current treatment, and should be the subject of further research.

Clausal Extraction from Connectives(*5 introduction*)

These rules include various types of connectives that allow entailment of the clause following them.

Some coordinative conjunctions (*but, so, for*)[1] allow inference of independent clauses joined in a compound sentence.
Example:

(21) Mary wanted to go out, *but* John wanted to stay home. ⇒ John wanted to stay home.

Similarly, some subordinative conjunctions allow inference of the dependent clause in complex sentences. They express relations such as time (*before, after, until, when*),

[1]This also applies for *and*, which we already covered in the previous rule.

cause and effect (*because, since, as*), comparison and contrast (*although, though, while*), and manner and place (*where, how*).
Examples:

(22) *When* we arrived to the cinema, the movie had already started. ⇒ We arrived to the cinema.

(23) John told me *how* Mary fixed the bug. ⇒ Mary fixed the bug.

We also extract clauses that follow certain conjunctive adverbs (*meanwhile, however, nevertheless*).
Examples:

(24) *Meanwhile*, Sony lost the lead to its Japanese rival Matsushita. ⇒ Sony lost the lead to its Japanese rival Matsushita.

Relative clause (*3 introduction*)

Relative clauses embed propositional content which may serve in the entailment process. We use introduction rules to extract relative clauses as independent propositions. Our rules account for various configurations of relative clauses:

- Human and non-human antecedents (who/whom vs. which).
- Different grammatical case (who/whom/whose).
- Reduced relative clauses where the relative pronoun is omitted , as in example (26).
- Use with prepositions, as in example (27).

Examples:

(25) The assailants fired six bullets at the car, *which carried Vladimir Skobtsov.* ⇒ The car carried Vladimir Skobtsov.

(26) The chaotic situation *unleashed in Bogota last night* began on 28 July in Medellin. ⇒ The chaotic situation was unleashed in Bogota last night.

(27) This generation, *to whom the torch was passed*, gladly and quickly took up the challenge. ⇒ The torch was passed to this generation.

The implementation of one of the relative clause rules, handling cases such as example (25), was shown and explained in section 6.2.4.

Note: In the case of reduced relative clause, the extracted proposition will be a passive sentence. If it has a *by* clause, the passive rule can transform it into an active sentence, e.g. *"The man arrested by the police was involved in a series of bank robberies."*⇒ *"The man was arrested by the police"*⇒ *"The police arrested the man"*.

Determiner Canonization (*2 substitution*)

Most determiners entail existential quantification, expressed by the indefinite article *'a/an'* in most cases. Our rule base includes such canonization rules for the following types of determiners: the definite article (*the*), quantifiers (*each, every, several, some*), and demonstratives (*this, that, these, those*). Notice that possessive determiners are handled by the "genitive to definite" rule, described next.
Examples:

(28) Bilbo found *the* ring. ⇒ Bilbo found *a* ring.

(29) *This* student managed to solve all the exercises. ⇒ *A* student managed to solve all the exercises.

Note: The definite to indefinite transformation fails under generics (*the rich, the poor*) as well as 'natural' uniqueness (*the universe, the management*).

Genitive to definite (*substitution*)

Genitive constructions are definite. We use a rule to make this substitution.
Example:

(30) Bilbo's ring is dangerous. ⇒ The ring is dangerous.

(31) Your dog bit me. ⇒ The dog bit me.

As another example of rule interaction, notice that the outcome of this rule may be chained with the definite-to-indefinite rule, e.g. to obtain *"a dog bit me"* from the right-hand-side of (31).

Genitive to modifier (*substitution*)

Genitive expressions may be paraphrased as post-nominal modifiers.
Example:

(32) Spain's natural resources were depleted. ⇒ The natural resources of Spain were depleted.

The implementation of this rule is shown in Figure 6.3.

Case adjustment rules (*10 substitution*)

As illustrated in sentences (8)-(9), application of the passive-to-active transformation may leave nominative pronouns (*he, she, we*) in an object position and accusative pronouns (*him, her, us*)in a subject position. The case adjustment rules handle such situations by changing the pronoun's case from nominative to accusative or vice versa, as required. One of the implemented rules is shown in figure 6.4

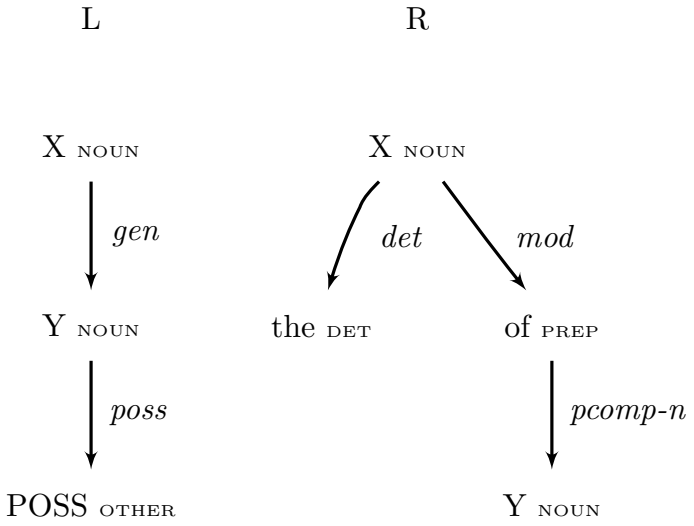


Figure 6.3: Genitive to modifier (*substitution*). This rule transforms expressions of the type *'Y 's X '* to *'the X of Y '*. The variable *POSS* catches the possessive marker (*'s* or *'*) so that it will not be copied to the derived tree as part of *Y*'s subtree.

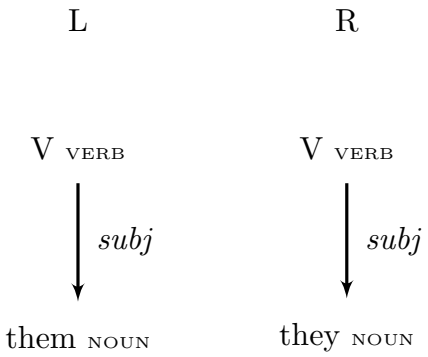


Figure 6.4: Accusative to nominative adjustment (*substitution*)

6.3.2 Lexicalized Inference Rules

Verb complement clause extraction (*5 introduction*)

This class of rules extracts a clausal complement of verbs which permit such entailment. A rich source of such verbs is the polarity lexicon developed at PARC by Nairn

et al. (2006), based on their analysis of implicative and factive verbs[1]. The PARC lexicon consists of verbs and their subcategorization frames, which are semantically characterized by preserving or reversing the truth of their complements, when appearing in positive (non-negated) or negative contexts. For our introduction rules, we considered verbs that, when appearing in positive contexts, preserve the truth of their complement. The corresponding entailment rules extract these complements as separate trees.

The implemented rules correspond to several verb subcategorization frames specified in the lexicon, which we adapted to the Minipar format. These include *subject + finite clause* (sentence 33), *subject + infinite clause* (sentence 34), and *subject + object + infinite clause* (sentence 35). For each frame, all the corresponding verbs are represented as a single packed rule[2]. Figure 6.5 shows one of the implemented rules. *Examples:*

(33) [$_{subj}$ John] *admitted* [$_{comp-fin}$ that Mary was right]. ⇒ Mary was right.

(34) [$_{subj}$ Ed] *remembered* [$_{comp-inf}$ to lock the door]. ⇒ Ed locked the door.

(35) [$_{subj}$ Mary] *forced* [$_{obj}$ John] [$_{comp-inf}$ to pay for the damage.] ⇒ John paid for the damage.

The role of the subcategorization frame in determining polarity is illustrated by sentences (36)–(37). These examples show that different frames for the same verb may induce different polarity.

(36) John *forgot* to buy milk. ⇒ John didn't buy milk.

(37) John *forgot* that he already bought milk. ⇒ John already bought milk.

[1]We are grateful to Cleo Condoravdi for making the PARC lexicon available for this research.

[2]More accurately, in the rule base we have two rules for each frame, to preserve the distinction between strict and plausible entailments defined in the PARC lexicon.

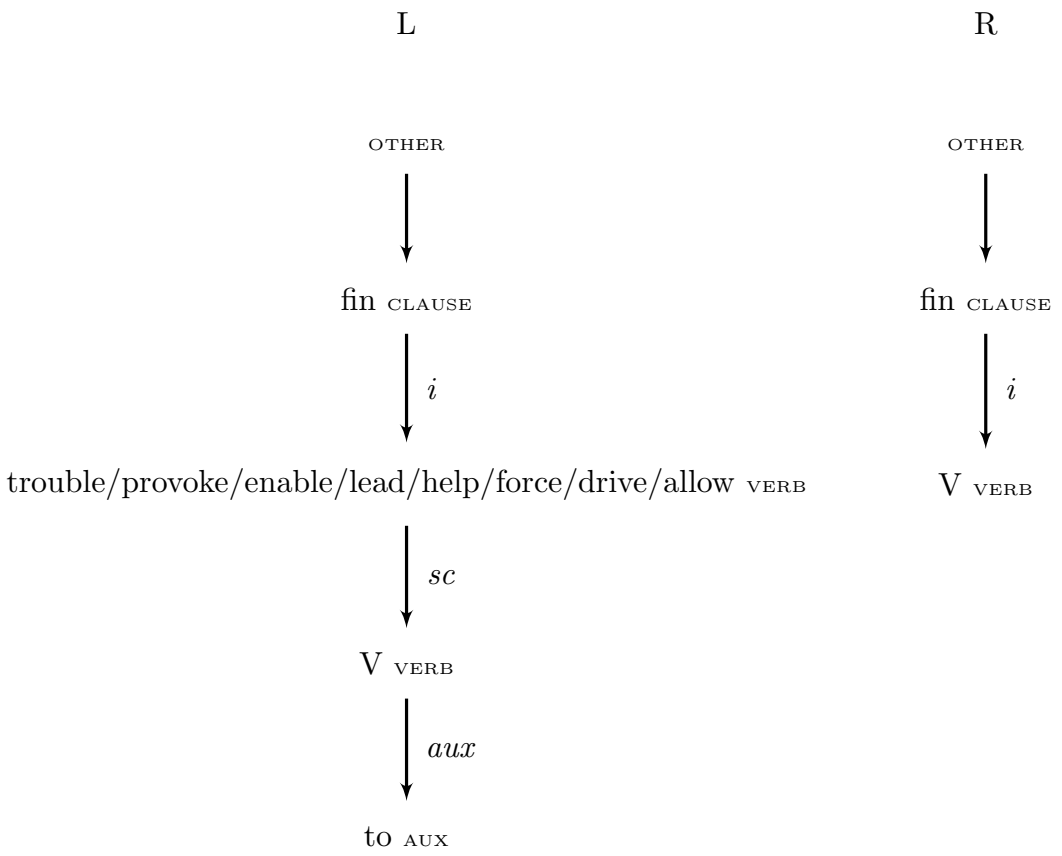


Figure 6.5: Extraction of verbal complement (*introduction*). This rule corresponds to the *subject+object+infinite clause* subcategorization frame. It extracts the clausal complement of the main verb (*trouble/provoke/...*), whose root is matched by V, as a separate tree.

In addition to their use as a source for introduction rules, these lexicons were also utilized for developing lexicalized polarity annotation rules, as described in section 6.4.2.

Reporting verbs finite clause (*introduction*)

These rules extract clauses embedded as complements of "reporting verbs" such as *say, announce*, and *report*. We assume content reported by such verbs to be veridical.

Our list includes the following verbs: *say, report, declare, tell, announce, disclose, add, emphasize, stress, clarify*, which are commonly found in news articles.
Example:

(38) LA police chief William Bratton *told* CNN that the police are collecting evidence to help determine how the 'Heal the World' singer died. ⇒
The police are collecting evidence to help determine how the 'Heal the World' singer died.

6.4 Polarity Annotation Rules

As we described in section 4.6, polarity annotation is used mainly for detecting mismatches between the text (and its consequents) and the hypothesis. Polarity may be affected by various types of contexts such as auxiliary verbs, explicit negation, conditionals, and the existence of certain clause-embedding verbs. Our polarity annotation rules detect such relevant contexts and set the polarity of the corresponding predicate accordingly. As with inference rules, the polarity rules are also divided into *generic* and *lexicalized* rules, which together amount to 30 rules.

6.4.1 Generic Polarity Rules

Explicit negation (12 rules)

Most commonly, negative polarity is expressed by explicit negation, where the negation particle appears either in full form (*not*) or in contracted form (*n't*). In verbal negation, the negation particle follows the auxiliary verb. Our rules address the following types of auxiliary verbs:

(39) *Passive:* John *was not/wasn't* invited$^{[-]}$.

(40) *Modal:* John *can not/can't/cannot* dance$^{[-]}$.

(41) *Present participle:* John *is not/isn't* dancing$^{[-]}$.

(42) *Past participle:* John *has not/hasn't* arrived$^{[-]}$ yet.

(43) *Dummy auxiliary:* John *did not/didn't* know$^{[-]}$ the answer.

Verbal negation rules also handle infinitives (44), and verbs modified by the adverb *never* (45).

(44) Mary convinced John *not to* dance$^{[-]}$.

(45) John *never* dances$^{[-]}$.

Finally, the rules annotate negated predicates in copular sentences, both nominal (46) and adjectival (47).

(46) John *is not/isn't* a student$^{[-]}$.

(47) John *is not/isn't* famous$^{[-]}$.

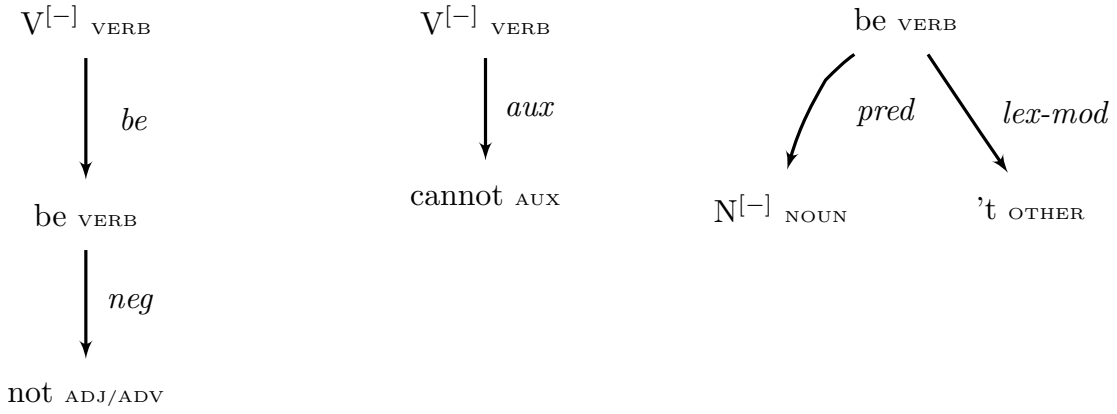
The implementation of some explicit negation rules is shown in Figure 6.6.

Implied NP negation

These rules detect specific pronouns that imply negation: *no one, noone, nobody, none, nothing.* They may appear at various syntactic positions, including subject (48), direct and indirect object (49). These pronouns trigger negative polarity annotation of the corresponding verb.
Examples:

(48) *No one* stayed$^{[-]}$ for the last lecture.

(49) A witty saying proves$^{[-]}$ *nothing* (Voltaire).



John is not listening[−]. John cannot sing[−]. John isn't a student[−].

Figure 6.6: Explicit negation rules, and their application to sample sentences

Modal auxiliaries (2 rules)

These rule set polarity to *unknown* due to the presence of the following modal auxiliaries: *could, should, can, must, may, might.*
Example:

(50) I *could* eat[?] a whale now!

Note the absence of *would* from the rule. It requires finer heuristics to determine if *would* functions as a modality marker or future tense marker in a sentence.

Overt conditionals (3 rules)

Conditional sentences can be identified if they are introduced by an overt complementizer, such as *if, whether, unless.* The polarity of both the condition and the consequence is set to *unknown.*
Example:

(51) *if* Venus wins[?] this game, she will meet[?] Sarena in the finals.

6.4.2 Lexicalized Polarity Rules

Negative and unknown polarity of verb complement clause (7 rules)

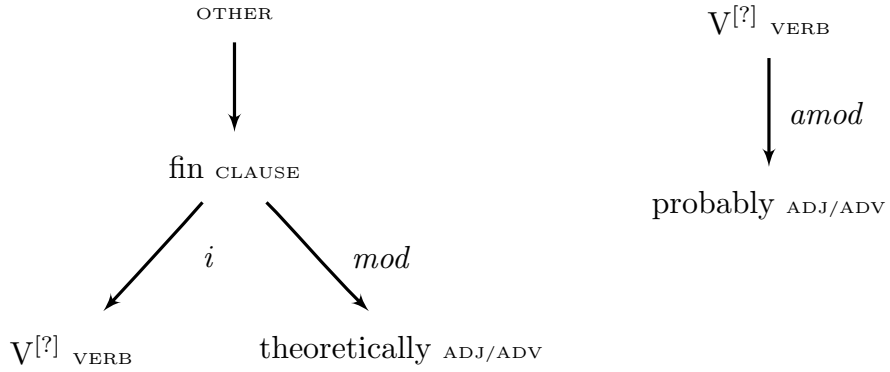
In section 6.3.2 we described lexicalized inference rules that extract clausal complements for specific verbs and subcategorization frames inducing *positive* polarity context. These rules were derived from the PARC polarity lexicon (Nairn et al., 2006). When the polarity is *negative* or *unknown*, we do not extract the embedded clause, but rather use annotation rules to mark its polarity as negative/unknown.

In addition to PARC lexicon, we used VerbNet (Kipper, 2005) as a complementary source for lexicalized polarity rules. VerbNet is a comprehensive verb lexicon for English. It extends Levin's (1993) attempt to classify English verbs into syntactically and semantically coherent classes. For example, consider the verb *want*, as in *"Everyone wanted the war to end"* . Its post verbal complement can be marked with *unknown* polarity. VerbNet allows us to extend this knowledge to other verbs. Searching VerbNet, we find that *want* is the representative member of verb class 32.1 . Other members of this class include *covet*, *crave*, *fancy* and *desire*, hence we may extend the rule to cover these verbs as well. Crucially, VerbNet gives us more information than a thesaurus or WordNet-like database since verbs classification takes into account their syntactic configuration as well as semantic behavior. Negative polarity rules are based on both the PARC lexicon and VerbNet, while unknown polarity rules are based solely on VerbNet, since the PARC lexicon does not address unknown polarity. In the future, we plan to use VerbNet as a source of lexicalized inference rules as well.

Examples:

(52) I *pretend* that I know$^{[-]}$ calculus.

(53) Whenever possible, I *refrain* from eating$^{[-]}$ meat.



Adverb modifying the CLAUSE node Adverb modifying the verb

Figure 6.7: Adverbs marking unknown polarity - two syntactic variations.

(54) They *suspected* that John is cheating[?].

(55) They *requested* John to quit[?] smoking.

Adverbs marking unknown polarity (2 rules)

Unknown polarity may also be induced by certain adverbs. Our rules detect the following adverbs: *probably, possibly, theoretically, presumably, potentially, hopefully, seemingly, apparently, likely.*
Examples:

(56) *Theoretically*, I know[?] how to cook rice.

(57) She *probably* danced[?] all night.

The Minipar representation of (56) places *Theoretically* as a modifier of the artificial CLAUSE node, a sister of the verb *know*. Sentence (57) has different representation, where the adverb *probably* modifies the verb *dance*. Thus, the rule base contains two separate rules that capture these variations, shown in Figure 6.7.

Adjectives marking negative and unknown polarity (2 rules)

In sentences of the structure *"It is ADJ that S"*, certain adjectives (ADJ) set the polarity of the embedded clause S to negative or unknown. Negative polarity adjectives include *impossible, inconceivable, unimaginable* and *unthinkable*. Unknown polarity adjectives include *unlikely, likely, improbable, probable* and *possible*.
Examples:

(58) It is *impossible* that he survived$^{[-]}$ such a fall.

(59) It is *unlikely* that he survived$^{[?]}$ such a fall.

Figure 6.8 shows the implementation of this rule for the unknown polarity adjectives.

6.5 Robust Rule Base Derivation for a Target Parser

The previous sections described the linguistic phenomena modeled in our entailment rule base, and illustrated their implementation. We next describe a methodology for deriving concrete, parser-specific rules from high-level description of linguistic phenomena. Our implementation is based on Minipar, but the same methodology is applicable to other parsers as well.

The implemented rules correspond to linguistic structures that are treated consistently by the parser, that is, structures whose representation in the parser's output is largely predictable. Yet, as illustrated in the previous sections, modeled phenomena often correspond to multiple parser representations, and therefore a robust implementation of the rule base must take these variants into account. Rules should be specified at an appropriate level of detail: underspecified rules may harm precision, while overspecified rules may harm recall.

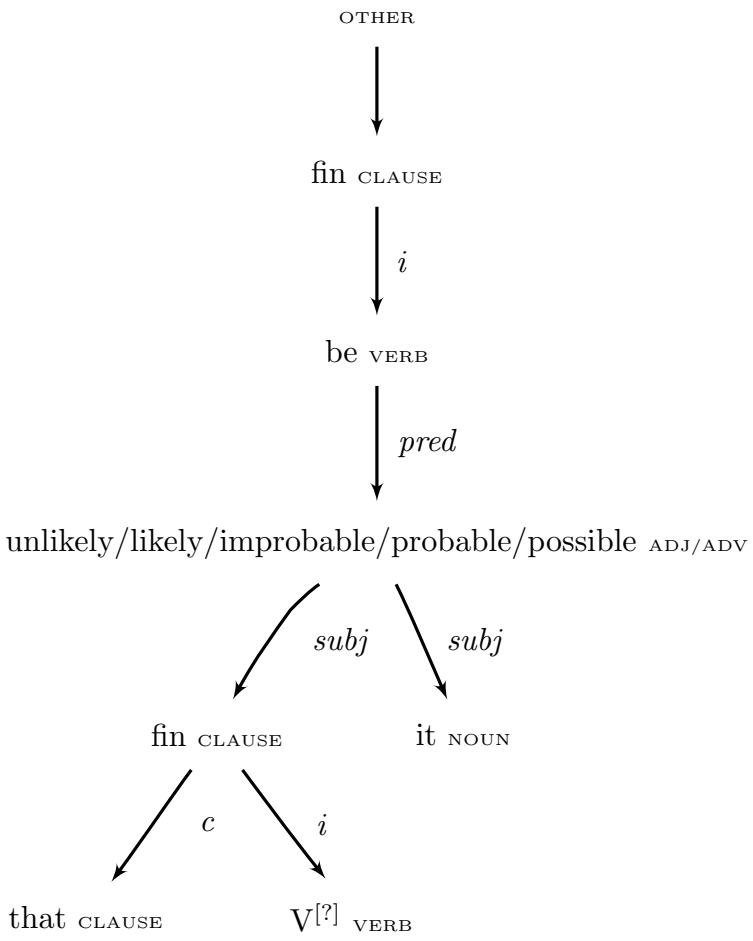


Figure 6.8: Adjectives marking unknown polarity in sentences of the structure *"It is ADJ that S"*.

These considerations illustrate the importance of testing the rules on real-world data. The next two subsections describe how we utilized corpus-based data to characterize the parser's behavior and to compose and validate our entailment rules. Finally, we briefly describe our rule development environment.

Relation	Description	Percentage
mod	adjunct modifier	15.13
punc	punctuation	11.64
pcomp-n	nominal complement of prepositions	9.14
lex-mod	lexical modifier	9.11
det	determiner	7.43
i	the relationship between a main clause and a complement clause	7.03
subj	subject of verbs	6.39
s	surface subject	6.37
obj	object of a verb	4.89
nn	noun-noun modifier	4.24

Table 6.3: 10 most frequent Minipar dependency relations

6.5.1 Preliminary study of the parser's output

In order to obtain some preliminary statistics on Minipar's output, we parsed a sample of texts and looked at the frequency of dependency relations in the resulting parse trees. When parsing the texts of the RTE2 development set, we observed 76 distinct dependency relations. However, we found that the 25 most frequent relations already cover more than 90% of the edges, and the 40 most frequent relations cover more than 99% of the edges. Table 6.3 lists the ten most frequents dependency relations in the RTE2 development set. Focusing on the most salient relations greatly simplified rule writing, with only little decrease in coverage. It was also a practical approach to investigate undocumented behavior of Minipar.

Beyond these statistics, we learned from examining Minipar's output about the representation of various syntactic phenomena, variations and inconsistencies in its parses and so on.

6.5.2 Rule Composition and Validation

We have experimented with two complementary approaches for building individual rules, *synthetic* and *corpus-based*. In the synthetic approach, we write the rule from scratch, specifying each of the required nodes and dependency relations. This straightforward approach allows direct capturing of the desired linguistic structures. However, the resulting rule might require substantial adjustments before it can be successfully applied to real-world data.

Therefore, our alternative approach was to rely heavily on corpus verification early in the construction of individual rules. In the corpus-based approach, we start with an example text taken from a textbook, a research article or a corpus sample, featuring the linguistic phenomenon we are trying to model. The rule $L \rightarrow R$ is then composed as follows.

Constructing an initial L: We obtain an initial left-hand-side (L) by applying the following adjustments to the sample text parse s:

1. Extract the relevant subtree from s.
2. Replace content nodes with variables where required.
3. Generalize function words to their appropriate classes. For instance, we may replace "he" in the original parse with "he/she/it" to represent this entire pronoun class.
4. Remove irrelevant subtrees such as adjunct modifiers (e.g. *"X was found yesterday* $\Rightarrow$ *X was found"*.

As an illustration of this process, consider again the passive rule presented in Chapter 4. The above procedure derives the left-hand-side L of the rule (Figure 4.1(b), left) from the source sentence tree (Figure 4.1(a), left).

L validation and adjustment We then look for matches of the initial L in a large corpus, and manually review a sample of the obtained matches. This provides a quick and efficient way to verify that L reliably represents the linguistic structure we are after. At this point, we may need to repeat the process of adjusting L and testing it, until the expected behavior has been reached. If a rule's recall is too low, L may be too restrictive. In this case, it may be necessary to simplify L further by removing redundant subtrees or replacing some words with variables. If, on the other hand, the precision is too low, we may find that the set of matches obtained contains irrelevant constructions. In this case, L may need to be constrained in some way, e.g. by adding more context to it. We may decide to split L into several distinct rules, each with a refined L, which match slightly different configurations, but all representing the same linguistic phenomenon.

Rule completion Next, we complete the rule according to its type. For polarity annotation rules, we simply add polarity features to L, to be copied to matched nodes. For inference rules, we need to construct the right-hand-side, R. As with L, we obtain an initial R by simplifying and adjusting a parsed sentence. This sentence is typically the desired result of applying (manually) the transformation we wish to model to the source sentence. R is also validated against the corpus in a similar fashion to L. Finally, we may add alignment links between L and R nodes, as required.

Rule testing Finally, the complete rule is validated. The rule is tested on both synthetic examples and corpus samples. To test rule robustness, we have tried to validate the rules under "noisy" synthetic examples. Starting from an example that was already validated on the rule, we manually created more complex examples by attaching subtrees as premodifiers and postmodifiers and validating that the instantiated R remains as expected after application. For example, given that our passive/active

transformation works correctly for the sentence *"John was attacked by villains"*, we may test it on the following examples:

- John was attacked by villains *yesterday.*
- John was attacked *yesterday* by villains.
- *Yesterday,* John was attacked by villains.

Another test is to embed the synthetic example at different levels of a larger text, and test whether the rule performs as expected. Each of these variations may result in slightly different parse structures, which would require rule adaptation.
Examples:

- It was reported that John was attacked by villains.
- That John was attacked by villains was never confirmed.
- My sister said that she heard her friend tell her mother how John was attacked by villains.

An important issue in rule testing was to validate that rules interact well with each other. Namely, to ensure that in an entailment chain involving multiple rules, each rule generates a right-hand-side that is a valid left-hand-side for the next rule. We have learned that this was an important test during development, when we found rules that would not apply when chained together, although they had all been successfully validated on their own. Such circumstances may arise by minor mismatches between one rule's R and another rule's L. A typical example is the incorrect case for pronouns after passive-to-active transformation, described earlier.

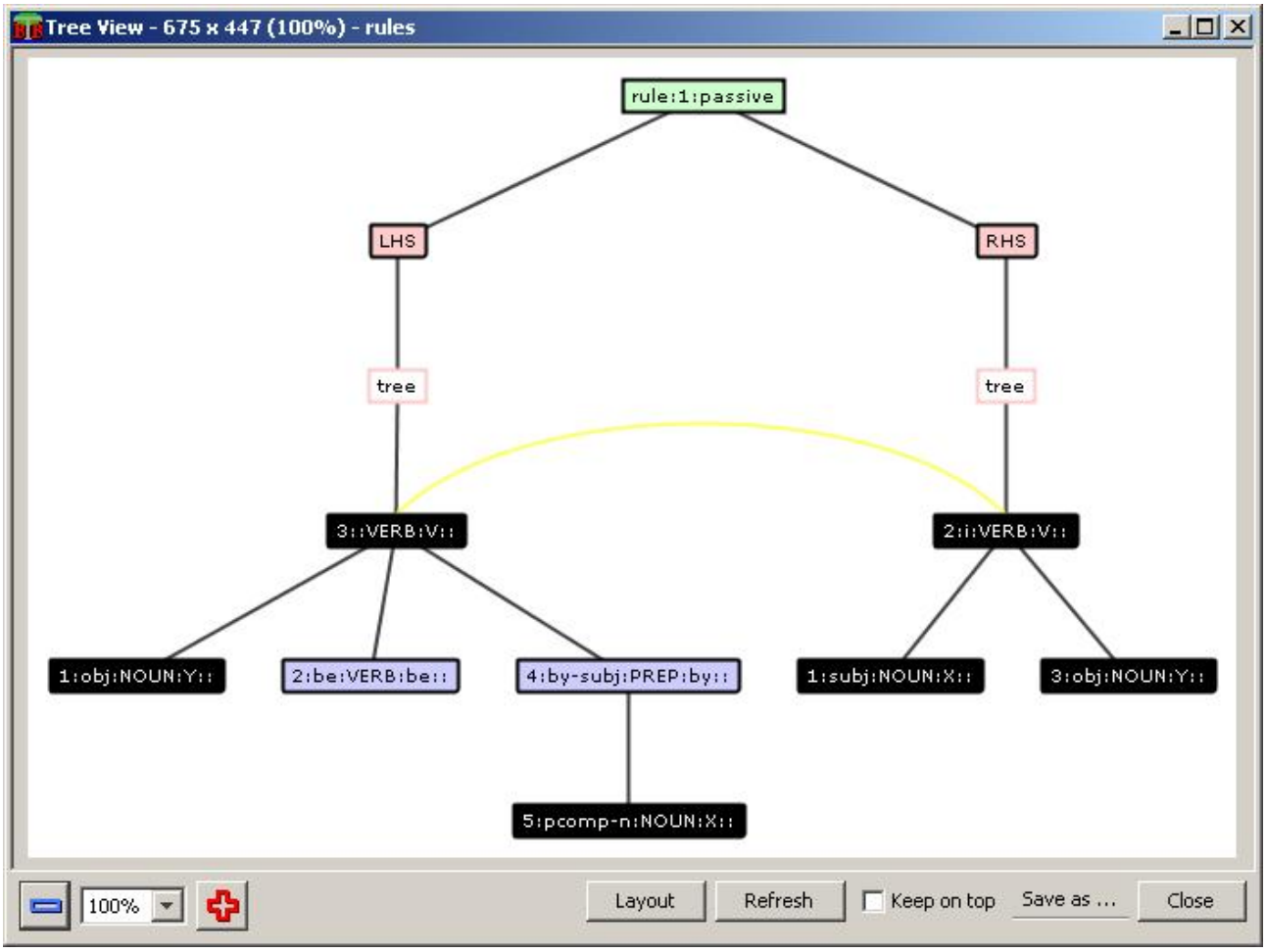


Figure 6.9: Passive rule displayed in the ClarkSystem environment

6.5.3 Rule Development Environment

Manual rule engineering requires a convenient and productive environment for editing and viewing rules. We used the ClarkSystem (Simov et al., 2003), an XML-based system for corpora development. It allows definition of custom XML schemes, and includes a visualization module. Figure 6.9 shows the visualization of the passive rule in the ClarkSystem, and figure 6.10 lists the XML representation of this rule.

```
<rule id="1" name="passive">
  <LHS>
    <tree>
      <node id="1" parent="3" rel="obj" tag="NOUN" type="var">
        <lemma>Y</lemma>
      </node>
      <node id="2" parent="3" rel="be" tag="VERB" type="word">
        <lemma>be</lemma>
      </node>
      <node id="3" tag="VERB" type="var">
        <lemma>V</lemma>
      </node>
      <node id="4" parent="3" rel="by-subj" tag="PREP" type="word">
        <lemma>by</lemma>
      </node>
      <node id="5" parent="4" rel="pcomp-n" tag="NOUN" type="var">
        <lemma>X</lemma>
      </node>
    </tree>
  </LHS>
  <RHS>
    <tree>
      <node id="1" parent="2" rel="subj" tag="NOUN" type="var">
        <lemma>X</lemma>
       </node>
       <node id="2" rel="i" tag="VERB" type="var">
        <lemma>V</lemma>
       </node>
       <node id="3" parent="2" rel="obj" tag="NOUN" type="var">
         <lemma>Y</lemma>
       </node>
    </tree>
  </RHS>
</rule>
```

Figure 6.10: XML encoding of the passive rule

6.6 Summary

In this chapter we presented a novel comprehensive collection of manually-constructed entailment rules modeling common linguistic phenomena. This collection comprises inference and polarity rules, both generic and lexicalized. We presented and discussed the linguistic phenomena underlying these rules, as well as our methodology for robust derivation of concrete parser-specific rules from these high-level descriptions. The inferential utility of this rule base is assessed empirically in chapter 8 in several application settings. We believe that our rule base provides a good starting point towards a more complete collection of generic rules, and plan to extend it in future work.

Chapter 7

A Proof-Based RTE System: Integrating Approximate Entailment Classification

7.1 Introduction

Chapters 4-5 presented a semantic inference formalism over parse-based representations, and its efficient implementation using the compact forest data structure. The implemented inference engine allows application of diverse types of semantic knowledge, represented uniformly as entailment rules.

However, as we pointed out in chapter 2, knowledge-based inference is usually not sufficient for practical entailment recognition systems, which typically combine it with more heuristic methods for approximate entailment classification. In this chapter we describe our complete entailment recognition system, which is based on modular combination of the inference engine with an approximate entailment classifier. The inference engine employs the generic rules described in chapter 6, as well as

a variety of large-scale rule bases that were derived from available semantic resources, or extracted automatically from texts. The approximate entailment classifier is based on supervised machine learning, and its implemented features follow the common practice in state of the art RTE systems, while adapting them to our setting.

The rest of this chapter is organized as follows. Section 7.2 describes the system architecture. Section 7.3 describes the usage of the inference engine within the system, focusing on the incorporated rule bases and the implemented search strategy. Finally, section 7.4 completes the system presentation with the description of the classification module.

7.2 System Overview

The input for the system is a text-hypothesis pair (t, h), where the text ranges from a single sentence to a short paragraph, and the hypothesis is a single sentence, typically a simple assertion. The system comprises three stages:

1. Preprocessing of t and h.

2. Knowledge-based inference.

3. Feature extraction and entailment classification.

A block diagram for the system is shown in figure 7.1. Preprocessing includes dependency parsing using Minipar (Lin, 1998), named entity recognition using the Stanford NER (Finkel et al., 2005) and co-reference resolution using OpenNLP[1]. In addition, we have developed a normalization module for numbers, which is applied first. The result is a set of dependency trees for the text, and a single tree for the hypothesis, with additional annotations of co-reference and named entities.

[1] http://opennlp.sourceforge.net/

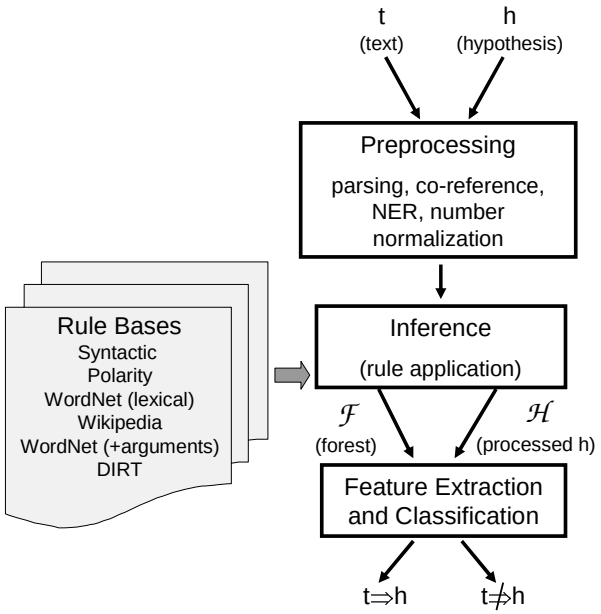


Figure 7.1: RTE system architecture

Next, the inference engine applies to the text trees entailment rules derived from diverse knowledge sources, aiming to generate the hypothesis or get as close as possible to it. The set of inferred trees is efficiently represented as a compact forest $\mathcal{F}$. The polarity annotation rules, and a small set of 'canonization' entailment rules (such as passive-to-active transformation) are also applied to the hypothesis, resulting in a modified hypothesis tree $\mathcal{H}$.

The resulting pair $(\mathcal{F}, \mathcal{H})$ is passed to the final stage - feature extraction and entailment classification. Entailment classification is based on two criteria: (a) How close we managed to get to the hypothesis? This is represented by a set of lexical and lexical-syntactic features measuring the level of coverage of $\mathcal{H}$ by $\mathcal{F}$. It is assumed that high coverage indicates entailment; (b) Identified mismatches between $\mathcal{H}$ and $\mathcal{F}$, indicating non-entailment.

7.3 Knowledge-Based Inference

7.3.1 Rule Bases

Our RTE system makes use of the generic rule base presented in chapter 6, including both the syntactic inference rules and the polarity annotation rules. Inference rules that merely extract a subtree out of the source tree without changing its structure (such as the relative clause rule) were excluded. These rules are useful for exact inference that aims to generate the hypothesis, and were used in the evaluation of such inferences (cf. section 8.1). However, the currently implemented approximate classification features are focused on matching substructures of the hypothesis in the forest (to be described in section 7.4), and hence do not take advantage of such extractions.

In addition to the generic rules, the system integrates inference knowledge from a variety of large-scale semantic resources, introduced in section 2.3. The information derived from these resources is represented uniformly as entailment rules in our formalism. Some examples for such rules were shown in table 4.1. The following resources have been used:

WordNet: We extracted from WordNet (Fellbaum, 1998) lexical rules based on the *synonym*, *hyponym* (a word is entailed by its hyponym, e.g. '*dog*→ *animal*'), *hyponym instance* and *derivation* relations.

Wikipedia: We used the lexical rulebase of Shnarch et al. (2009), who extracted rules such as *'Janis Joplin → singer'* from Wikipedia based on both its metadata (e.g. links and redirects) and text definitions, using patterns such as '*X is a Y*'.[1]

[1]In addition to the extraction methods described in (Shnarch et al., 2009), we employed two additional methods. First, extraction of entailments among terms that are redirected to the same page. Second, generalization of rules with the same RHS and common LHS head, but different

DIRT: The DIRT algorithm (Lin and Pantel, 2001) learns from a corpus entailment rules between binary predicates, e.g. *'X is fond of Y→X likes Y'*. We used the version described in (Szpektor and Dagan, 2007), which learns canonical rule forms.

Argument-Mapped WordNet (AmWN): A resource for entailment rules between predicates, covering both verbal and nominal forms (Szpektor and Dagan, 2009), including their argument mapping, based on WordNet and NomLex-plus (Meyers et al., 2004), verified statistically through intersection with the unary-DIRT algorithm (Szpektor and Dagan, 2008).

These automatically-extracted inference rules lacked two attributes defined in our formalism: rule type (substituion/introduction) and explicit alignments (beyond alignments between R's variables and their L counterparts, which are defined by default). These attributes are added automatically using the following heuristics:

1. If the roots of L and R have the same part-of-speech, then it is a substitution rule (e.g. '*X buy Y→ Y was sold to X*'). Otherwise (e.g. '*Y's acquisition by X→ Y was sold to X*'), it is an introduction rule.

2. The roots of L and R are assumed to be aligned.

7.3.2 Search

In the current system we implemented a simple search strategy, in the spirit of (de Salvo Braz et al., 2005): first, we applied three exhaustive iterations of generic rules. Since these rules have low fan-out (few possible right-hand-sides for a given left-hand-side) it is affordable to apply and chain them more freely. At each iteration

modifiers. For instance, the rules '*Ferrari F430→ car*' and '*Ferrari Ascari→ car*' are generalized into '*Ferrari→ car*'.

we first find all rule matches, and then apply all matched rules. To avoid repeated identical rule applications, we mark newly added nodes at each iteration, and in the next iteration consider only matches containing new nodes. We then perform a single iteration of all other lexical and lexical-syntactic rules, applying them only if their L part was matched in $\mathcal{F}$ and their R part was matched in $\mathcal{H}$. Further investigation of effective search heuristics over our representation is left for future research.

7.4 Entailment Classification[1]

While available semantic knowledge can often bring t closer to h, complete derivation of h from t is not feasible for most RTE (t, h) pairs. Therefore, a robust RTE system must complement knowledge-based inference with an approximate entailment classification module, which would tolerate partial proofs resulting from incomplete knowledge. Our classifier aims to measure how close the proof system managed to get to h, as well as to detect cues for non-entailment (cf. section 2.2).

The classifier is trained on labeled (t, h) examples from the RTE development sets. After the inference engine is applied to the given t and h, a feature vector is extracted from the resulting $(\mathcal{F}, \mathcal{H})$ pair. We used the SVMperf package (Joachims, 2005, 2006) to train a linear kernel SVM on these examples.

Features can be broadly categorized into two subsets:

1. *Lexical* features that depend solely on the lexical items in $\mathcal{F}$ and $\mathcal{H}$. These features consider only the nodes of the trees and completely ignore the edges (section 7.4.1).

2. *Lexical-syntactic* features that consider both lexical and structural matches (or mismatches) between $\mathcal{H}$ and $\mathcal{F}$, taking both nodes and edges into account (sections 7.4.2–7.4.3).

[1] Joint work with Jonathan Berant.

Unlike several other systems (cf. section 7.4), our features do not presuppose a fixed alignment between $\mathcal{H}$ nodes and $\mathcal{F}$. Instead, each substructure of $\mathcal{H}$ under consideration (a node, an edge, a predicate and its arguments etc.) may be matched anywhere in $\mathcal{F}$. This allows a more flexible combination of information from various consequents in $\mathcal{F}$, even when the relevant co-reference information necessary to combine these pieces is missing. On the other hand, alignment is likely to better capture global structures. We address this by introducing a novel feature that aims to match larger substructures of $\mathcal{H}$ in $\mathcal{F}$. This feature and its efficient computation are described in section 7.4.3.

7.4.1 Lexical Features

Our lexical features comprise coverage features and polarity-based features.

Coverage Features The following features check if the words in $\mathcal{H}$ are present in $\mathcal{F}$. We assume that good lexical coverage correlates well with entailment. The first five features measure the proportion of uncovered distinct verbs[1]/nouns/adjectives and adverbs/Named Entities/numbers in $\mathcal{H}$. The last coverage feature measures the proportion of covered distinct words[2] in $\mathcal{H}$.

Polarity Features We conjecture that mismatched polarity is indicative of non-entailment. Three binary features are thus extracted:

1. Whether there exists a noun/verb in $\mathcal{H}$ such that all its matches in $\mathcal{F}$ have a mismatching polarity.

2. Whether there exists a noun/verb in $\mathcal{H}$, which has a match in $\mathcal{F}$ with opposite

[1]The verb 'be' is discarded during features extarction.

[2]Only the following Minipar parts-of-speech are taken into account: Verb, Noun, Adjective/Adverb and Other.

polarity (i.e. *positive* vs. *negative*), but doesn't have a match with the same polarity.

3. Whether there exists a noun/verb in $\mathcal{H}$, which has a match in $\mathcal{F}$ with incompatible but non-opposite polarity (i.e. the polarity of one of the nouns/verbs is *unkonwn*), but doesn't have a match with the same polarity.

Note that the first feature is the disjunction of the second and third features.

7.4.2 Local Lexical-Syntactic Features

Local lexical-syntactic features consider matching of local structures of $\mathcal{H}$ in $\mathcal{F}$. Two types of local lexical-syntactic features are extracted: predicate-argument features and edge coverage features.

Predicate-argument features: If $\mathcal{F}$ entails $\mathcal{H}$, then the predicates in $\mathcal{H}$ should be matched in $\mathcal{F}$ along with their arguments. Predicates include verbs (aside for the verb "be") or subject complements in copular sentences (for example, *smart* in *"Joseph is smart."*). Arguments are the daughters of the predicate node in $\mathcal{H}$.[1] Four features are computed for each $(\mathcal{F}, \mathcal{H})$ pair. We categorize every predicate in $\mathcal{H}$ that has a match in $\mathcal{F}$ into one or more of four possible categories:

1. *complete match* - a matching predicate exists in $\mathcal{F}$ with matching arguments and dependency relations.
2. *partial match* - a matching predicate exists in $\mathcal{F}$ with some matching arguments and dependency relations.
3. *opposite match* - a matching predicate exists in $\mathcal{F}$ with some matching arguments but non-matching dependency relations.

[1] When the dependent is a preposition or a clause we take the complement of the preposition or the head of the clause respectively as the dependent.

4. *no match* - no matching predicate in $\mathcal{F}$ has any matching arguments.

If some predicate is categorized as a *complete match* it will not be in any other category. Finally, we compute the four features for the $(\mathcal{F}, \mathcal{H})$ pair: the proportion of predicates in $\mathcal{H}$ that have a *complete match* in $\mathcal{F}$, and three binary features checking if there is any predicate in $\mathcal{H}$ categorized as a *partial match/opposite match/no match*. Since the *subject* and *object* arguments are crucial for textual entailment, we compute four similar features only for the subset of predicates which have these arguments (while ignoring other arguments).

Edge coverage features We say that an edge in $\mathcal{H}$ is *matched* in $\mathcal{F}$ if there is an edge in $\mathcal{F}$ with a matching *relation* name, *source* node and *target* node. We say that an edge in $\mathcal{H}$ is *loosely-matched* if there is some directed path in $\mathcal{F}$ from a matching *source* node to a matching *target* node. Based on these definitions we extract two features: the proportion of $\mathcal{H}$ edges *matched/loosely matched* in $\mathcal{F}$.[1]

7.4.3 A Global Lexical-Syntactic Feature

All the lexical-syntactic features hitherto presented focus on local syntactic dependencies. We would like to compute the coverage of $\mathcal{H}$ by $\mathcal{F}$ for larger and more complex substructures. Kernel functions are often used to compute the similarity between complex structures such as dependency trees. Collins and Duffy (2001) introduced a dependency tree kernel, which measures the similarity between two trees by counting their common subtrees.

While the general idea of finding common subtrees seems attractive as an approximate syntactic measure, applying the dependency tree kernel measure as-is for textual entailment has several drawbacks: First, this measure, like any kernel function, is symmetric, while entailment is a directional relation: we want all $\mathcal{H}$ subtrees

[1] We only look at a subset of $\mathcal{H}$ edges labeled with relevant dependency relations.

to be matched in $\mathcal{F}$, but the opposite is not required. Second, the dependency tree kernel counts the number of times each $\mathcal{H}$ subtree appears in $\mathcal{F}$ while we only wish to know if it appears in $\mathcal{F}$ or not. This is important since $\mathcal{F}$ may contain redundant structures resulting from rule application. Third, Collins and Duffy assign equal weight to all nodes in the dependency tree, while it is likely that the closer the node is to the root, the more significant it is for entailment. Finally, we need to adapt the tree kernel measure to operate on a compact forest and a tree, rather than on a pair of trees.

To address these issues, we have adapted the dependency tree kernel function to our needs. We measure how well the subtrees in $\mathcal{H}$ are covered by $\mathcal{F}$, weighted according to the proximity to the root of $\mathcal{H}$. This feature is computed in two steps, detailed below. First, we find for each pair of nodes $n_H \in \mathcal{H}$ and $n_F \in \mathcal{F}$ the maximal-weight subtree rooted at both n_H and n_F, with respect to some node weighting function, and store its weight in a table. Note that unlike tree kernels, we do not count the number of common substructures[1], but only consider the maximal-weight subtree. In the second step, we compute an overall structural coverage score based on these weights.

Table Construction For each node pair $n_H \in \mathcal{H}$ and $n_F \in \mathcal{F}$, we would like to compute $MaxWght(n_H, n_F)$, the maximal weight of a common subtree rooted at both n_H and n_F. The weight of a subtree is defined (recursively) as the sum of its node weights, given by some node weighting function $wght(n)$. In our implementation, the weight of a node is decreased according to its distance from the root:

$$wght(n) = 1 + max_{v \in \mathcal{H}} dist_{\mathcal{H}}(v) - dist_{\mathcal{H}}(n) \tag{7.1}$$

where $dist_{\mathcal{H}}(n)$ is the distance of n from the root of $\mathcal{H}$.

[1]Though this could be easily computed.

$MaxWght$ can be defined recursively as:

$$MaxWght(n_H, n_F) = \begin{cases} 0 & \text{if } n_H \neq n_F \\ wght(n_H) + & \\ BM(n_H, n_F) & \text{otherwise} \end{cases} \tag{7.2}$$

where $BM(n_H, n_F)$ is the weight of the best match of n_H daughter subrees in the daughter subtrees of n_F, formally defined as:

$$BM(n_H, n_F) = \max_{f \in daughterMatches(n_H, n_F)} \sum_{v \in daughters_f(n_H)} MaxWght(v, f(v)) \tag{7.3}$$

$daughterMatches(n_H, n_F)$ is the set of all matches[1] in $\mathcal{F}$ of any subtree composed of n_H and some (possibly empty) subset of its daughters, where n_H is mapped to n_F. $daughters_f(n_H)$ denotes the subset of n_H daughters mapped by the match function f. It can be shown that the matches evaluated in this process are valid, i.e. they are all embedded subtrees in $\mathcal{F}$. We efficiently compute $MaxWght$ for each pair of nodes $n_H \in \mathcal{H}$ and $n_F \in \mathcal{F}$ in a bottom-up manner using dynamic programming.

We note that finding the best match cannot be done independently for each outgoing edge of n_H, otherwise multiple edges in $\mathcal{H}$ could be mapped to the same d-edge in $\mathcal{F}$. Nevertheless, computing the best match $BM(n_H, n_F)$ can still be done efficiently since this is an instance of the *assignment problem*, which is solvable in polynomial time (Burkard et al., 2009). The assignment problem is defined as finding a match in a weighted bipartite graph such that the sum of the edge weights (costs) is maximized. In our case, we aim to find a maximal-weight match between the outgoing edges of n_H and the outgoing d-edges of n_F, where the cost of mapping an edge e whose target node is t_e to a d-edge d whose matching target node is t_d is $MaxWght(t_e, t_d)$.

[1] cf. match definition in section 5.3

Feature Computation After computing the table of costs for each pair of nodes in $\mathcal{H}$ and $\mathcal{F}$, a global similarity GS between $\mathcal{H}$ and $\mathcal{F}$ can be easily computed. For each node n_H in $\mathcal{H}$, we compute the maximal weight of a subtree rooted in n_H that is matched in $\mathcal{F}$. GS is obtained by summing these weights over n_H nodes:

$$GS(\mathcal{H},\mathcal{F}) = \sum_{n_H \in \mathcal{H}} \max_{n_F \in \mathcal{F}} MaxWght(n_H, n_F)$$

and the final feature is taken as the normalized similarity $\frac{GS(\mathcal{H},\mathcal{F})}{GS(\mathcal{H},\mathcal{H})}$.

7.5 Summary

This chapter described a complete textual entailment recognition system. The main components in the systems are the inference engine which utilizes diverse large-scale entailment rule bases to generate a (compact) forest of consequents, and an approximate entailment classifier, which measures that coverage of the hypothesis by these consequents, while aiming to detect mismatches that may indicate non-entailment. The system is evaluated empirically in the next chapter.

Chapter 8

Evaluation

This chapter presents an empirical evaluation of our entailment system as a whole, as well as evaluation of its individual components. Both the quality of the system's output (in terms of accuracy, precision and recall) and its computational efficiency (in terms of running time and space) are evaluated, using various application settings.

We first evaluate the knowledge-based inference engine. Section 8.1 reports an experiment in which the engine aims to strictly prove simple template hypotheses, representing binary predicates, from texts sampled from a large corpus. Its precision and estimated recall on this task are evaluated in several configurations, each employing a different subset of rules. The results show that our proof-based approach outperforms the common practice of matching the target predicate in the text as-is, and demonstrates recall/precision tradeoffs obtained by applying different rule subsets.

Next, in section 8.2 we evaluate the efficiency of our engine implementation using the compact forest data structure. We found that using the compact forest reduces run time and space by orders of magnitude, comparing to a naïve implementation of our formalism that generates each inferred consequent explicitly (henceforth, referred to as *explicit inference*). The size of the compact forest remains small after applying

hundreds of rules, making our engine highly scalable.

Finally, section 8.3 describes the evaluation of the complete entailment system, including the approximate entailment classifier. Our system achieves competitive accuracy on the RTE benchmarks, and knowledge-based inference is shown to contribute to the overall accuracy. We also show that each of the individual knowledge sources has positive marginal contribution. In addition, we present feature analysis for the approximate entailment classification module, which provides some insight on our feature set.

8.1 Proof System Evaluation

In this experiment we evaluate the inference engine on finding strict proofs, i.e. the inference process must derive precisely the target hypothesis (or an instantiation of it, in the case of template hypotheses, which contain variables, as defined in section 4.8). Thus, we should evaluate its precision over text-hypothesis pairs for which a complete proof chain is found, using the available rules. We note that the PASCAL RTE datasets are not suitable for this purpose. These rather small datasets include many pairs for which entailment recognition requires approximate matching, as currently it is not realistic to assume sufficient knowledge that will enable a complete exact proof. As an alternative we chose a Relation Extraction (RE) setting, for which complete proofs can be achieved for a large number of corpus sentences. In this setting, the system needs to identify in sentences pairs of arguments for a target semantic relation (e.g. *'X buy Y'*).

8.1.1 System Configuration

Unlike the rest of the experiments in this chapter (sections 8.2–8.3), which are based on the RTE system of chapter 7, in this experiment we used an earlier version of

the engine and the rule bases. The engine in this experiment does not make use of the compact forest, but rather it generates each consequent explicitly. Polarity annotation rules are applied to the original text t, and to each inferred consequent, prior to any inference rule application. Some of the rules specify polarity values to be matched by L, which allowed blocking of certain inappropriate rule applications, such as for negated predicates. This was useful for inferences such as *"John managed to see the movie. ⇒ John saw the movie."*, which should be avoided if *managed* is negated (cf. section 6.3.2). The following rule bases have been used in this experiment:

Generic linguistic rules We used the generic rule base presented in chapter 6, including both inference and the polarity annotation rules. This early version did not include the lexicalized polarity rules derived from VerbNet and from the PARC lexicon.

Lexical-Syntactic Rules *Nominalization rules:* Entailment rules such as 'X's acquisition of $Y \rightarrow X$ acquired Y' capture the relations between verbs and their nominalizations. These rules were derived automatically (Ron, 2006) from Nomlex, a hand-coded database of English nominalizations (Macleod et al., 1998), and from WordNet.

Automatically Learned Rules: DIRT (Lin and Pantel, 2001) and TEASE (Szpektor et al., 2004) are two state-of-the-art unsupervised algorithms that learn lexical-syntactic inference rules. DIRT identifies semantically related templates in a local corpus using distributional similarity over the templates' variable instantiations. TEASE acquires entailment relations from the Web for a given input template I by identifying characteristic variable instantiations shared by I and other templates. Both algorithms provide for a given input template a ranked list of output templates.

Some of the learned rules are linguistic paraphrases, e.g. 'X confirm $Y \rightarrow X$ approve Y', while others capture world knowledge, e.g. 'X buy $Y \rightarrow X$ own Y'. These algorithms do not learn the entailment direction of the rule, which reduces their accuracy when applied in any given direction. For each system, we considered the top 15 bi-directional rules learned for each template.

Generic default rules these rules are used to define default behavior, in situations where no case-by-case rules are available. We used one default rule that allows removal of any modifiers from nodes. Desirably, specific rules should be specified in future work to capture more precisely many cases that are currently handled by this default rule.

8.1.2 Evaluation Process

We use a sample of test template hypotheses that correspond to typical RE relations, such as *'X approve Y'*. We then identify in a large test corpus sentences from which an instantiation of the test hypothesis is proved. For example, the sentence *"the budget was approved by the parliament"* is found to prove the instantiated hypothesis *"parliament approve budget"* (via the passive-to-active inference rule). Finally, a sample of such candidate sentences-hypothesis pairs are judged manually for true entailment. The process was repeated to compare different system configurations.

Since the publicly available sample output of TEASE is much smaller than the other resources, we selected from this resource 9 transitive verbs that may correspond to typical RE predicates[1], forming test templates by adding subject and object variable nodes. For example, for the verb *accuse* we have constructed the template '$X_{\text{NOUN}} \xleftarrow{subj} accuse_{\text{VERB}} \xrightarrow{obj} Y_{\text{NOUN}}$'.

For each test template h we need to identify in the corpus sentences from which

[1]The verbs are *approach, approve, consult, lead, observe, play, seek, sign, strike.*

it can be proved by our system. To find efficiently proof chains that generate h from corpus sentences we combined forward and backward (Breadth-First) search over the available rules. First, backward search is used over the lexical-syntactic rules, starting with rules whose right-hand-side is identical to the test template. While backward chaining the DIRT/TEASE and nominalization rules, this process generates a set of templates t_i, all of them proving (deriving) h. For example, for the hypothesis *'X approve Y'* we may generate the template *'X confirm Y'*, through backward application of a DIRT/TEASE rule, and then further generate the template *'confirmation of Y by X'*, through a nominalization rule. Since the templates t_i are generated by lexical-syntactic rules, which modify open-class lexical items, they may be considered as "lexical expansions" of h.

Next, for each specific t_i we generate a search engine query composed of the open-class words in t_i. This query fetches from the corpus candidate sentences, from which t_i might be proven using the generic linguistic rules (recall that these rules do not modify open-class words). To that end we apply a forward search that applies the generic rules, starting from a candidate sentence s and trying to derive t_i by a sequence of rule applications. If successful, the variables in t_i are instantiated (cf. section 4.8). Consequently, we know that, under these variable instantiations, h can be proved from s (since s derives t_i which in turn derives h).

The above search for sentences that prove each test template was performed over the Reuters RCV1 corpus, CD#2, applying Minipar for parsing. Through random sampling we obtained 30 sentences that prove each of the 9 test templates, yielding a total of 270 pairs of a sentence and an instantiated hypothesis for each of the four tested configurations, described below (1080 pairs overall). These pairs were split for entailment judgment between two human annotators. The annotators achieved, on a sample of 100 shared examples, an agreement level of 87%, and a Kappa value of 0.71 (corresponding to "substantial agreement").

#	Configuration	Precision	Yield
1	BASELINE (embed h anywhere in s)	67.0%	2,414
2	PROOF (embed h at the root of s)	78.5%	1,426
3	PROOF + GENERIC RULES	74.8%	2,967
4	PROOF + GENERIC AND LEXICAL-SYNTACTIC RULES	23.6%	18,809

Table 8.1: Proof system evaluation

8.1.3 Results

We tested 4 configurations of the proof system:

1. BASELINE: The baseline configuration follows the prominent approach in graph-based entailment systems: the system simply tries to embed the given hypothesis anywhere in the candidate sentence tree s, while only negative or unknown polarity (detected by the annotation rules) may block embedding.

2. PROOF: In this configuration h has to be strictly generated from the candidate sentence s. The only inference rule available is the default rule for removing modifiers (polarity annotation rules are active as in BASELINE). Notice that this configuration is equivalent to embedding h in s with the root of h matched at the root of s, since modifiers that are not part of the match can be removed from s by the default rule. However, if h is embedded elsewhere in s it will not be extracted, as opposed to the BASELINE configuration.

3. PROOF + GENERIC RULES: As PROOF, plus generic linguistic rules.

4. PROOF + GENERIC AND LEXICAL-SYNTACTIC RULES: As the previous configuration, plus lexical-syntactic rules.

For each system configuration we measure *precision*, the percentage of examples judged as correct (entailing), and average *extrapolated yield*, which is the expected

number of truly entailing sentences in the corpus that would be proved as entailing by the system. The extrapolated yield for a specific template is calculated as the number of sample sentences judged as entailing, multiplied by the sampling proportion. The average is calculated over all test templates. We note that, similar to IR evaluations, it is not possible to compute true recall in our setting since the total number of entailing sentences in the corpus is not known (recall is equal to the yield divided by this total). However, it is straightforward to measure *relative* recall differences among different configurations based on the yield. Thus, using these two measures estimated from a large corpus it is possible to conduct robust comparison between different configurations, and reliably estimate the impact of different rule types. Such analysis is not possible with the RTE datasets, which are rather small, and their hand-picked examples do not represent the actual distribution of linguistic phenomena.

The results are reported in Table 8.1. First, comparing the results for PROOF with the results for BASELINE, we observe that the requirement for matching h at the root of s (i.e. at the main clause of s), rather than allowing it to be matched anywhere in s improves the precision considerably over the baseline (by 11.5%), while reducing the yield by nearly 40%. The PROOF configuration avoids errors resulting from improper extraction of embedded clauses, e.g. inferring *"parliament approve budget"* from the sentence *"Income taxes will be increased if the parliament approves the budget"*.

Remarkably, using the generic inference rules, our system is able to gain back the lost yield in PROOF and further surpass the yield of the baseline configuration. In addition, a higher precision than the baseline is obtained (a 7.8% difference), which is statistically significant at a $p < 0.05$ level, using z test for proportions. This demonstrates that our principled proof approach appears to be superior to the more heuristic baseline embedding approach, and exemplifies the contribution of our generic rule base. Overall, generic rules were used in 46% of the proofs.

Adding the lexical-syntactic rules increased the yield by a factor of six(!). This shows the importance of acquiring lexical-syntactic variability patterns. However, the precision of DIRT and TEASE is currently quite low, causing overall low precision. Manual filtering of rules learned by these systems is currently required in order to obtain reasonable precision.

Error analysis revealed that for the third configuration PROOF + GENERIC RULES, a significant 65% of the errors are due to parsing errors, most notably incorrect dependency relation assignment, incorrect POS assignment, incorrect argument selection, incorrect analysis of complex verbs (e.g. *play down* in the text vs. *play* in the hypothesis) and ungrammatical sentence fragments. Another 30% of the errors represent conditionals, negation and modality phenomena, most of which could be handled by additional rules, some making use of more elaborate syntactic information such as verb tense. The remaining, and rather small, 5% of the errors represent truly ambiguous sentences which would require considerable world knowledge for successful analysis.

8.2 Compact Forest Efficiency Evaluation

Next, we evaluate the efficiency of compact inference (cf. chapter 5) in the setting of recognizing textual entailment, using the RTE-3 and RTE-4 datasets (Giampiccolo et al., 2007, 2008). These datasets consist of (*text, hypothesis*) pairs, which need to be classified as *entailing/non entailing*. Our first experiment shows, using the generic inference rule set, that compact inference outperforms explicit inference (efficiency-wise) by orders of magnitude (Section 8.2.1). The second experiment shows that compact inference scales well to a full-blown RTE setting with several large-scale rule bases, where up to hundreds of rules are applied per text (Section 8.2.2).

	Compact	Explicit	Ratio
Time (msec)	61	24,184	396
Rule applications	12	123	10
Node count	69	5,901	86
Edge endpoints	141	11,552	82

Table 8.2: Compact vs. explicit inference, using generic rules. Results are averaged per text-hypothesis pair.

8.2.1 Compact vs. Explicit Inference

To compare explicit and compact inference we randomly sampled 100 pairs from the RTE-3 development set, and parsed the *text* in each pair using Minipar (Lin, 1998). In order to avoid memory overflow for explicit inference, we applied to these sentences only the subset of generic inference rules described in section 7.3. To make a fair comparison, we aimed to make the explicit inference implementation reasonably efficient, e.g. by preventing multiple generations of the same tree by different permutations of the same rule applications. Both configurations perform rule application iteratively, until no new matches are found. In each iteration we first find all rule matches and then apply all matching rules. We compare run time, number of rule applications, and the overall generated size of nodes and edges, where edge size is represented by the sum of its endpoints (2 for a regular edge, $|S_d| + |T_d|$ for a d-edge).

The results are summarized in Table 8.2. As expected, the results show that compact inference is by orders of magnitude more efficient than explicit inference. To avoid memory overflow, inference was terminated after reaching 100,000 nodes. 3 out of the 100 test texts reached that limit with explicit inference, while the maximal node count for compact inference was only 268. The number of rule applications is reduced thanks to the sharing of common subtrees in the compact forest, by which a single rule application operates simultaneously over a large number of embedded trees. The results suggest that scaling to larger rule bases and longer inference chains

	RTE3-Dev		RTE4	
	Avg.	Max.	Avg.	Max.
Rule applications	14	275	15	110
Node count	71	606	80	357
Edge endpoints	155	1,741	173	1,062

Table 8.3: Application of compact inference to the RTE-3 Dev. and RTE-4 datasets, using all rule types.

would be feasible for compact inference, but prohibitive for explicit inference.

8.2.2 Application to an RTE System

The goal of the second experiment was to assess that compact inference scales well for broad entailment rule bases. In this experiment we used the knowledge-based inference component of our RTE system, whose configuration, including rule bases and search strategy, was described in section 7.3. Overall, these rule bases contain millions of rules.

Table 8.3 provides statistics on rule applications using all rule bases, over the RTE-3 development set and the RTE-4 dataset[1]. Overall, the primary result is that the compact forest indeed accommodates well extensive rule applications from large-scale rule bases. The resulting forest size is kept small, even in the maximal cases which were causing memory overflow for explicit inference.

[1]Running time is not included since most of it was dedicated to rule fetching, which was rather slow for our available implementation of some resources. The elapsed time was a few seconds per (t, h) pair.

8.3 Complete RTE System Evaluation

The previous sections evaluated our knowledge-based inference engine (the proof system) with respect to the quality of its output (precision, recall) as well as its computational efficiency (time, space). We now turn to evaluate the complete RTE system described in chapter 7, which combines the inference engine with an approximate classification module, described in section 7.4. The system was trained on the RTE-3 development set, and was tested on the RTE3 and RTE-4 test sets (no development set was released for RTE-4). Co-reference substitution was disabled due to the insufficient accuracy of the co-reference resolution tool we used. We first report its overall performance, and then provide some analysis of both the inference and the classification modules.

The accuracies obtained in this experiment are shown in Table 8.4 (under the "infernece" column). The results on RTE-3 are quite competitive: compared to our 66.4%, only 3 teams out of the 26 who participated in RTE-3 scored higher than 67%, and three more systems scored between 66% and 67%. The results for RTE4 rank 9-10 out of 26, with only 6 teams scoring higher by more than 1%. Overall, these results show that our system is well-situated in the state of the art for the RTE task.

8.3.1 Usage and contribution of knowledge bases

In order to evaluate the accuracy gain from knowledge-based inference, we ran the system with the inference module disabled, so that entailment classification is applied directly to the initial parse tree of the text. The results are shown under the "no inference" column of table 8.4. Comparing these results to the full system accuracy ("inference"), we see that inference improved the accuracy on both test sets. The contribution was more prominent for the RTE-4 dataset. These results illustrate a typical contribution of current knowledge sources for current RTE systems. This

	Accuracy		
Test set	No inference	Inference	Δ
RTE3	64.6%	66.4%	1.8%
RTE4	57.5%	60.6%	*3.1%

Table 8.4: Inference contribution to RTE performance. The system was trained on the RTE-3 development set. * indicates statistically significant difference (at level $p < 0.02$, using McNemar's test).

contribution is likely to increase with current and near future research, on topics such as extending and improving knowledge resources, applying them only in semantically suitable contexts, improved classification features and broader search strategies.

Tables 8.5 and 8.6 illustrate the usage and contribution of individual rule bases. Table 8.5 shows the distribution of rule applications over the various rule bases. Table 8.6 presents ablation study showing the marginal accuracy gain for each rule base. These results show that each of the rule bases is applicable for a large portion of the pairs, and contributes to the overall accuracy.

Rule base	RTE3-Dev		RTE4	
	Rules	App	Rules	App
WordNet	0.6	1.2	0.6	1.1
AmWN	0.3	0.4	0.3	0.4
Wikipedia	0.6	1.7	0.6	1.3
DIRT	0.5	0.7	0.5	1.0
Generic	4.7	10.4	5.4	11.5
Polarity	0.2	0.2	0.2	0.2

Table 8.5: Average number of rule applications per (t, h) pair, for each rule base. *App* counts each rule application, while *Rules* ignores multiple matches of the same rule in the same iteration.

Rule base	ΔAccuracy (RTE4)
WordNet	0.8%
AmWN	0.7%
Wikipedia	1.0%
DIRT	0.9%
Generic	0.4%
Polarity	0.9%

Table 8.6: Contribution of various rule bases. Results show accuracy loss on RTE-4, obtained when removing each rule base (ablation tests).

8.3.2 Feature Analysis

Finally, we tested the contribution of the various classification features to the correct classification of RTE-4 examples. Since no development set was released for RTE-4, we randomly split the RTE-4 dataset into two halves (500 pairs each), using one half for training and the other for testing (the system was not trained on RTE-3 as in the previous experiment to avoid dataset variance). The overall system accuracy in the setting was 58.8%, and inference contribution was 3%, similar to the RTE-4 result in the previous experiment.

For each feature, we applied the following measures:

1. Ablation test, measuring the effect of removing the feature.

2. Training a single-feature SVM classifier, measuring its discriminative power, and

3. Squared Pearson correlation (R^2) between the feature value and the correct label, represented as ± 1.

The results are shown in Table 8.7. According to both the 1-feature classifier and the Pearson correlation, the most prominent features are lexical coverage, noun coverage, and our novel global lexical-syntactic measure. Remarkably, lexical coverage

Group	Feature	Ablation	1-Feature Classifier	R^2 Correlation	Non-zero	Non-zero precision
Lexical coverage	verbs	57.0	54.4	0.014	29.0%	
	nouns	58.2	57.6	0.046	38.2%	
	adj/adv	58.2	52.8	0.013	14.4%	
	NE	58.0	52.4	0.016	9.2%	0.70
	numbers	57.8	50.2	0.003	3.6%	0.67
	all	58.2	57.8	0.052	99.0%	
Polarity	mismatch	58.4	50.6	0.009	2.8%	0.79
	opposite mismatch	58.2	50.0	0.007	1.4%	0.86
	unknown mismatch	58.0	49.0	0.002	1.4%	0.71
Predicate-argument Subject/Object	complete	58.2	49.0	0.002	5.2%	0.62
	partial	58.6	49.0	0.003	15.6%	
	opposite	58.6	48.0	0.002	09.8%	
	no match	58.4	49.0	0.000	12.0%	
Predicate-argument All	complete	58.6	52.4	0.003	7.8%	0.62
	partial	57.8	53.6	0.005	23.0%	
	opposite	58.0	49.0	0.005	18.8%	
	no match	58.0	49.0	0.000	25.6%	
Edge coverage	edge coverage	58.6	54.0	0.033	86.8%	
	loose-edge coverage	58.8	54.0	0.020	77.2%	
Global (GS)		58.4	56.0	0.039	99.6%	
All features		58.8				

Table 8.7: Feature Analysis Over RTE-4

alone achieves accuracy of 57.8%. Ablation tests show that none of the features degrade the results. However, since many of the features are correlated, removing a single feature usually has limited impact. We also examined sparse features, which are active (non-zero) for less than 10% of the pairs. Since these features are irrelevant for most of the pairs, we measured their precision only for pairs where the feature is non-zero. As shown in the table, although these features are sparse, they provide valuable information when active. In particular, the polarity features were found to be quite accurate.

8.4 Summary

This chapter presented empirical evaluation of our inference engine and the RTE system that was built around it. Both output quality (precision, recall, accuracy) and computational efficiency (time, space) were investigated. We also evaluated the contribution of the various knowledge bases, including the generic rule base developed as part of this thesis, and the variety of other semantic resources that were incorporated into the system. We further discuss the empirical results and their implications in chapter 10, which summarizes the contributions and achievements of this thesis.

Chapter 9

Related Work

The main focus of this thesis was the representation and use of semantic knowledge for entailment inference at the lexical-syntactic level. In chapter 1 we defined our requirements from a knowledge-based entailment engine: (a) It should be based on a principled formulation of knowledge representation and inference mechanisms; (b) It should represent and apply diverse types of inference knowledge in a simple, unified fashion; (c) The engine should also scale well to allow combination and chaining of inferences from multiple large-scale knowledge sources. In chapter 2 we reviewed previous textual entailment systems, and concluded that they only partially meet these requirements. In this chapter we compare our entailment engine to related approaches presented in chapter 2, with respect to the above requirements (section 9.1).

Beyond the comparison to related RTE work, we also compare certain aspects of our work to relevant work in other NLP areas. In particular, we discusses previous work on packed representations, and compares them to our compact forest (section 9.2).

9.1 RTE Systems

Previous RTE systems usually restricted both the type of allowed transformations and the search space. Systems based on lexical (word-based or phrase-based) matching of h in t typically applied only lexical rules (without variables), where both sides of the rule are matched directly in t and h (Haghighi et al., 2005; MacCartney et al., 2008). Our inference formalism is more expressive, allowing also syntactic and lexical-syntactic transformations as well as rule chaining.

Hickl (2008) derived from a given (t, h) pair a small set of *discourse commitments*, which are quite similar to the kind of consequents we derive by our syntactic and lexical-syntactic rules. These consequents were fed into the next stages of the RTE system – lexical alignment and entailment classification. The commitments were generated by a mixture of several different tools and techniques, compared to our generic unified inference process, and commitment generation efficiency was not discussed.

Braz et al. (2005) presented a semantic inference framework which "augments" the text representation with only the right-hand-side of an applied rule, and in this respect is similar to ours. However, in their work, both rule application and the semantics of the resulting "augmented" structure were not fully specified. In particular, the distinction between individual consequents was lost in the augmented graph. By contrast, our compact inference is fully formalized and is provably equivalent to an expressive, well-defined formalism operating over individual trees, where each inferred consequent can be recovered from the compact forest.

Recently, MacCartney and Manning (MacCartney and Manning, 2009) proposed a model of natural language inference which, similarly to our framework, operates directly on parse-based representations. Their work extends previous work on *natural logic* (Valencia, 1991), which has focused on semantic containment and monotonicity, by incorporating semantic exclusion and implicativity. They model the inference

of h from t as a sequence of atomic edits, each can be thought of as generating an intermediate premise. Their calculus computes the semantic relation between the source and the derived premise by propagating the semantic relation from the local edit upward through the parse tree according to the properties of intermediate nodes. For example, it can correctly infer that "*Some first-year students arrived* $\Rightarrow$ *Some students arrived*", but "*Every first-year student arrived* $\Leftarrow$ *Every student arrived*". The composition of these semantic relations along the inference chain yields the semantic relation holding between t and h. Their contribution seems complementary to ours: in both approaches the inference of h from t is modeled as a sequence of atomic steps (*rule applications* or *edits*). The focus of our framework is the representation and application of diverse types of transformations needed for entailment inference, as well as efficient representation of possible inference chains. Application of an inference rule is assumed to always generate an entailed consequent, and polarity rules may be used to detect situations where this assumption does not hold and block rule application. By comparison, the formalism of MacCartney and Manning assumes rather simple edit operations, and is focused on precise predication of the semantic relation between t and h for a given sequence of edits that transform t into h. Thus, combining these two complementary approaches seems a natural direction for future research.

9.2 Packed representations

Packed representations in various NLP tasks share common principles, which also underly our compact forest: factoring out common substructures and representing choice as local disjunctions. Applying this general scheme to individual problems typically requires specific representations and algorithms, depending on the type of alternatives that should be represented and the specified operations for creating them.

In our work, alternatives are created by rule application, where a newly derived subtree is set as an alternative to existing subtrees. Alternatives are specified locally using d-edges.

Packed chart representations for parse forests were introduced in classical parsing algorithms such as CYK and Earley (Jurafsky and Martin, 2008), and have been extended in later work for various purposes (Maxwell III and Kaplan, 1991; Kay, 1996). Alternatives in the parse chart stem from syntactic ambiguities, and are specified locally as the possible decompositions of each phrase into its sub-phrases.

Packed representations have been utilized also in transfer-based machine translation. Emele and Dorna (1998) translated packed source language representation to packed target language representation while avoiding unnecessary unpacking during transfer. Unlike our rule application, in their work transfer rules *preserve* ambiguity stemming from source language, rather than *generating* new alternatives. Mi et al. (2008) applied statistical machine translation to a source language parse forest, rather than to the 1-best parse. Their transfer rules are tree-to-string, contrary to our tree-to-tree rules, and chaining is not attempted (rules are applied in a single top-down pass over the source forest), and thus their representation and algorithms are quite different from ours.

Chapter 10

Conclusion

Five years after its introduction by Dagan and Glickman (2004), textual entailment (TE) has become an emerging research field in natural language processing. TE allows generic, unified modeling of semantic inferences that arise in various text-understanding applications, which makes it an appealing framework for applied semantics. The holy grail of TE research is the development of high-quality entailment engines that would encapsulate much of the inferences needed by semantic applications. Using such entailment engines within these applications would be a major step forward for natural language understanding. As we surveyed in chapter 1, initial attempts to utilize entailment technology within semantic applications have shown very promising results.

The goal of this thesis was therefore to develop a first version of such a generic entailment engine. Following the common practice in semantic applications, and RTE systems in particular, we aimed at an engine operating over parse-based (lexical-syntactic) representations. This level of representation is expressive enough to represent much of the required inferences, while avoiding the complexities of semantic interpretation into logical forms. Our manual analysis of the RTE dataset, described in chapter 3, confirmed the appropriateness of lexical-syntactic representations for

this task.

Clearly, making progress towards precise, wide-coverage engines depends on the availability of broad, high quality semantic knowledge sources, as well as on the ability of the entailment engine to utilize such knowledge effectively. Our research addressed both the representation and use of existing semantic knowledge and the development of novel knowledge resources for entailment, and made several contributions in both areas.

Our RTE analysis (chapter 3) emphasized the need to combine various types of semantic knowledge, such as lexical-syntactic paraphrases, syntactic transformations, synonyms, morphological derivations, and lexical world-knowledge relations. However, semantic applications operating over parse-based representations typically make limited and application-specific use of semantic knowledge, and often employ multiple knowledge representations and inference mechanisms. These applications largely lack a generic framework that specifies how semantic knowledge should be represented and applied.

This thesis introduced a more principled, well-formalized approach for semantic inference at the lexical-syntactic level. In chapter 4 we presented a semantic inference formalism over parse trees. Our formalism represents semantic knowledge uniformly as entailment rules, and specifies a small set of inference mechanisms for their application. In our formalism, application of an inference rule has a clear, intuitive interpretation as generating a new sentence parse (a *consequent*), semantically entailed by the source sentence. We also utilize the notion of entailment rules for unified modeling of inferences based on co-reference relations and trace annotation. While our formalism is simple and compact, it was shown to be quite expressive. It successfully represents diverse types of inferences, and allows their combination and composition. In our view, well-formalized entailment models are fundamental for the establishment and the advancement of this field, analogously to the role formal

models have played in parsing and machine translation.

A complementary theoretical and practical contribution of this thesis is a novel packed data structure (*compact forest*) and a corresponding inference algorithm (*compact inference*) for efficient application of entailment rules (chapter 5). Since the number of inferred trees (consequents) is potentially exponential in the number of applicable rules, such packed data structure was necessary for a practical implementation of our inference framework. Our compact forest can represent efficiently a large number of inferred trees by sharing common subtrees, while allowing the recovery of each represented tree. Each rule application adds only the rule RHS to the forest, and operates simultaneously on all the represented trees that share the same LHS match. We proved that the compact inference algorithm correctly implements our formalism, and showed that it allows exponential-to-linear complexity reduction, comparing to explicit generation of each consequent. Empirically, compact inference was found to reduce run time and space by orders of magnitude, and scaled well to very large rule bases and hundreds of rule applications. These results establish the feasibility of practical entailment engines based on our inference formalism.

Using our formalism, we have composed a novel comprehensive entailment rule base modeling common linguistic phenomena (chapter 6). The rule base comprises inference and polarity annotation rules, both generic and lexicalized. We presented a taxonomy of generic linguistic phenomena that are relevant for entailment, as well as our methodology for robust modeling of such phenomena by concrete parser-specific entailment rules.

The inference engine and entailment rule base developed in this thesis were evaluated on two tasks. The first task was unsupervised relation extraction from a large corpus, where the engine aimed to prove an instantiation of the target relation from candidate sentences found in the corpus. Our proof-based approach was shown to be significantly more precise than the common practice of embedding the relation

anywhere in the sentence. Using our generic inference rules, the inference engine outperformed the embedding baseline on both precision and (extrapolated) recall. Adding automatically-learned entailment rules substantially increased the recall, although the current precision of such resources requires manual filtering of acquired rules.

The engine was also evaluated on the recognising textual entailment (RTE) benchmarks. The RTE system is a pipeline of knowledge-based inference using our proof system, followed by an approximate entailment classifier that aims to quantify the remaining gaps and mismatches between the text and its consequents and the hypothesis. In addition to the generic inference and polarity rules, the engine used entailment rules from various sources, including WordNet and its enhancement with with argument structure (amWN), lexical-syntactic rules learned automatically from corpus (DIRT), and lexical rules extracted from Wikipedia. Knowledge-based inference improved the overall accuracy, and each of the knowledge sources was shown to be beneficial. Yet, the current contribution of semantic knowledge to RTE systems, including ours, is still limited. This contribution is likely to increase with current and near future research, on topics such as extending and improving knowledge resources, applying them only in semantically suitable contexts, improved classification features and broader search strategies.

We conclude by proposing some major future enhancements for the inference engine. One natural extension is to advance from sentence-level inference to the discourse level. The engine currently handles only nominal co-reference, via the co-reference substitution mechanism (section 4.5). In future research we would like to capture and utilize further discourse-level information, such as co-reference between events, "global" events and entities available in the discourse and so on. We would also like to enhance our framework with rule weights or probabilities, which may combine scores originating from the resource itself and our prior confidence in that resource.

A probabilistic transformation-based framework for entailment was introduced recently by Harmeling (2009). His framework contains a small set of transformations applied in a fixed order, whose probabilities are estimated from the RTE development set. Combining the expressive power of our framework with probabilistic modeling in the spirit of Harmeling seems like a promising direction for future research. Another direction that seems well worth exploring is improving our search strategies. We expect that allowing deeper rule chaining would enable better utilization of the available knowledge sources, compared to the current search strategy, which does not attempt chaining of lexicalized rules. While our compact inference opens the way for exploring large search spaces, it would still be necessary to complement it with an effective heuristic search strategy. This can be done by implementing an A^*-like algorithm, and defining an appropriate heuristic function based on the rule weights and the estimated remaining distance from the hypothesis.

Bibliography

A. Andreevskaia, Z. Li, and S. Bergler. Partial predicate argument structure matching for entailment determinationr. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944*, pages 332–343. Springer, 2006.

C. L. Baker. *English Syntax - 2nd Edition*. Mit Press, 1995. ISBN 978-0262521987.

R. Bar-Haim, I. Szpecktor, and O. Glickman. Definition and analysis of intermediate entailment levels. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 55–60, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. The Second PASCAL Recognising Textual Entailment Challenge. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, 2006.

R. Bar-Haim, I. Dagan, I. Greental, and E. Shnarch. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*, 2007.

R. Bar-Haim, J. Berant, and I. Dagan. A compact forest for scalable inference over entailment and paraphrase rules. In *EMNLP*, pages 1056–1065, Singapore, August 2009a.

R. Bar-Haim, J. Berant, I. Dagan, I. Greental, S. Mirkin, E. Shnarch, and I. Szpektor. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of the First Text Analysis Conference (TAC 2008)*, 2009b.

R. Barzilay and L. Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL 2003*, pages 16–23, Edmonton, Canada, 2003.

R. Barzilay and K. R. McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL 2001*, pages 50–57, Toulose, France, 2001.

R. Bhagat and D. Ravichandran. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of ACL-08: HLT*, 2008. URL http://www.aclweb.org/anthology/P/P08/P08-1077.

J. Bos and K. Markert. Recognising textual entailment with logical inference techniques. In *EMNLP*, 2005.

J. Bos and K. Markert. When logical inference helps determining textual entailment (and when it doesnt). In *Proceedings of The Second PASCAL Recognising Textual Entailment Challenge*, 2006.

R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009. ISBN 9780898716634.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2001.

I. Dagan and O. Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. PASCAL workshop on Text Understanding and Mining, 2004.

I. Dagan, O. Glickman, and B. Magnini. The PASCAL Recognising Textual Entailment Challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944*, pages 177–190. Springer, 2006.

M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006.

R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *AAAI*, pages 1043–1049, 2005.

M. C. Emele and M. Dorna. Ambiguity preserving machine translation using packed representations. In *Proceedings of Coling-ACL*, 1998.

C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press, 1998.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. doi: http://dx.doi.org/10.3115/1219840.1219885. URL http://dx.doi.org/10.3115/1219840.1219885.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

D. Giampiccolo, H. Trang Dang, B. Magnini, I. Dagan, and B. Dolan. The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the TAC 2008 Workshop*, 2008.

O. Glickman and I. Dagan. Identifying lexical paraphrases from a single corpus: a case study for verbs. In *Proceedings of RANLP 2003*, 2003.

A. D. Haghighi, A. Y. Ng, and C. D. Manning. Robust textual inference via graph matching. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*, 2005.

S. Harabagiu, A. Hickl, and F. Lacatusu. Satisfying information needs with multi-document summaries. *Inf. Process. Manage.*, 43(6):1619–1642, 2007. ISSN 0306-4573. doi: http://dx.doi.org/10.1016/j.ipm.2007.01.004.

S. Harmeling. Inferring textual entailment with a probabilistically sound calculus*. *Natural Language Engineering*, 15(4):459–477, 2009.

A. Hickl. Using discourse commitments to recognize textual entailment. In *Proceedings of the 22nd Intenational Conference on Computational Linguistics (Coling)*, pages 337–344, 2008.

A. Hickl and S. Harabagiu. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*. Association for Computational Linguistics, 2006.

A. Hickl, J. Bensley, J. Williams, K. Roberts, B. Rink, and Y. Shi. Recognizing textual entailment with LCCs GROUNDHOG system. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, 2006.

A. Iftene and A. Balahur-Dobrescu. Answer validation on English and Romanian languages. In *Working Notes for the CLEF 2008 Workshop*, 2008.

T. Joachims. A support vector method for multivariate performance measures. In L. D. Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), August 7-11, 2005, Bonn, Germany*, pages 377–384. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-180-5. URL `http://doi.acm.org/10.1145/1102351.1102399`.

T. Joachims. Training linear SVMs in linear time. In *KDD*, pages 217–226. ACM, 2006.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, 2008.

M. Kay. Chart generation. In *Proceedings of ACL*, 1996.

J. Kazama and K. Torisawa. Exploiting Wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL*, 2007.

K. Kipper. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

M. Kouylekov and B. Magnini. Tree edit distance for textual entailment. In *Recent Advances in Natural Language Processing (RANLP)*, 2005.

J. R. Landis and G. G. Koch. The measurements of observer agreement for categorical data. *Biometrics*, 33:159–174, 1997.

J. Lehmann, C. Bizer, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 2009.

B. Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL, 1993. ISBN 9780226475332.

D. Lin. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*, 1998.

D. Lin and P. Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.

B. MacCartney and C. D. Manning. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics (IWCS-8)*, 2009.

B. MacCartney, T. Grenager, M.-C. de Marneffe, D. Cer, and C. D. Manning. Learning to recognize features of valid textual entailments. In *HLT-NAACL*, pages 41–48, 2006.

B. MacCartney, M. Galley, and C. D. Manning. A phrase-based alignment model for natural language inference. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing*, 2008.

C. Macleod, R. Grishman, A. Meyers, L. Barrett, and R. Reeves. Nomlex: A lexicon of nominalizations. In *In Proceedings of Euralex98*, pages 187–193, 1998.

J. T. Maxwell III and R. M. Kaplan. A method for disjunctive constraint satisfaction. In M. Tomita, editor, *Current Issues in Parsing Technology*. Kluwer Academic Publishers, 1991.

A. Meyers, R. Reeves, C. Macleod, R. Szekeley, V. Zielinska, and B. Young. The cross-breeding of dictionaries. In *Proceedings of LREC*, 2004.

H. Mi, L. Huang, and Q. Liu. Forest-based translation. In *Proceedings of ACL-08: HLT*, 2008.

S. Mirkin, I. Dagan, and E. Shnarch. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of EACL*, Athens, Greece, 2009.

R. Nairn, C. Condoravdi, and L. Karttunen. Computing relative polarity for textual inference. In *Proceedings of International workshop on Inference in Computational Semantics (ICoS-5)*, 2006.

M. Negri, M. Kouylekov, and B. Magnini. Detecting expected answer relations through textual entailment. In *CICLing*, pages 532–543, 2008.

R. D. Nielsen, W. Ward, and J. H. Martin. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(4):479–501, 2009.

B. Pang, K. Knight, and D. Marcu. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL 2003*, Edmonton, Canada, 2003.

S. P. Ponzetto and M. Strube. Deriving a large-scale taxonomy from wikipedia. In *AAAI*, pages 1440–1445, 2007.

R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A Comprehensive Grammar of the English Language*. Longman, 1985. ISBN 978-0582517349.

D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of ACL 2002*, Philadelphia, PA, 2002.

L. Romano, M. Kouylekov, I. Szpektor, I. Dagan, and A. Lavelli. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL 2006*, 2006.

T. Ron. Generating entailment rules based on online lexical resources. Master's thesis, Computer Science Department, Bar-Ilan University, 2006.

Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman. Automatic paraphrase acquisition from news articles. In *Proceedings of Human Language Technology Conference (HLT 2002)*, San Diego, USA, 2002.

E. Shnarch, L. Barak, and I. Dagan. Extracting lexical reference rules from Wikipedia. In *Proceedings of ACL-IJCNLP*, 2009.

K. I. Simov, A. Simov, M. Kouylekov, K. Ivanova, I. Grigorov, and H. Ganev. Development of corpora within the CLaRK system: The BulTreeBank project experience. In *EACL*, pages 243–246, 2003.

R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Coling-ACL*, pages 801–808, Morristown, NJ, USA, 2006a. Association for Computational Linguistics.

R. Snow, L. Vanderwende, and A. Menezes. Effectively using syntax for recognizing false entailment. In *HLT-NAACL*, pages 33–40, 2006b.

I. Szpektor and I. Dagan. Learning canonical forms of entailment rules. In *Proceedings of RANLP*, 2007.

I. Szpektor and I. Dagan. Learning entailment rules for unary templates. In *Proceedings of COLING*, 2008.

I. Szpektor and I. Dagan. Augmenting WordNet-based inference with argument mapping. In *Proceedings of ACL-IJCNLP Workshop on Applied Textual Inference (TextInfer)*, 2009.

I. Szpektor, H. Tanev, I. Dagan, and B. Coppola. Scaling web based acquisition of entailment patterns. In *Proceedings of EMNLP*, 2004.

I. Szpektor, I. Dagan, R. Bar-Haim, and J. Goldberger. Contextual preferences. In *Proceedings of ACL-08: HLT*, pages 683–691, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1078`.

M. Tatu and D. Moldovan. A semantic approach to recognizing textual entailment. In *EMNLP*, 2005.

M. Tatu and D. Moldovan. COGEX at RTE3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

M. Tatu, B. Iles, J. Slavick, A. Novischi, and D. Moldovan. COGEX at the Second Recognizing Textual Entailment Challenge. In *The Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, 2006.

V. S. Valencia. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam, 1991.

L. Vanderwende and W. B. Dolan. What syntax can contribute in the entailment task. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944*, pages 205–216. Springer, 2006.

E. M. Voorhees and D. Harman. Overview of the sixth Text REtrieval Conference (TREC-6). In *TREC*, pages 1–24, 1997.

R. Wang and G. Neumann. Recognizing textual entailment using a subsequence kernel method. In *AAAI*, pages 937–942, 2007.

F. m. Zanzotto, M. Pennacchiotti, and A. Moschitti. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(4):551–582, 2009.

Appendix A

Compact Forest Complete Proofs[1]

In this chapter we provide complete proofs for the correctness of the compact inference algorithm presented in chapter 5. We start with a few definitions.

Definition Let $L \rightarrow R$ be a rule matched and applied to a compact forest $\mathcal{F}$. As in section 5.3, let l be a subtree of some represented tree $t \in \mathcal{T}(\mathcal{F})$, in which L is matched. Recall that S_L was defined as l excluding nodes matched by dual-leaf variables, and similarly S_R was defined as a copy of R without its dual-leaf variables that is generated and inserted into $\mathcal{F}$ as part of rule application. The roots of S_L and S_R are denoted r_L and r_R respectively. We say that a node $s \in S_R$ is *tied to* a node $s' \in S_L$, if s is set as a source node for one of the outgoing d-edges of s', due to alignment sharing or dual leaf variable sharing.

The graph operations performed when applying a rule $L \rightarrow R$ on a cDAG $\mathcal{G}$ can be summarized as follows:

1. Adding the subtree S_R to $\mathcal{G}$.

2. Setting r_R as a target node of a d-edge in $\mathcal{G}$.

[1] Joint work with Jonathan Berant.

3. Setting nodes in S_R that are tied to nodes in S_L as source nodes for d-edges in $\mathcal{G}$, according to the rules of variable sharing and dual leaf variable sharing. Recall that these d-edges are by not part of S_L.

In the next two lemmas we establish that the inference process generates only cDAGs:

Lemma 1 *Every node in a cDG generated by the inference process has at most one incoming d-edge.*

Proof By construction, in the initial forest each node has at most one incoming d-edge. Each rule application adds a subtree S_R, whose nodes have at most one incoming d-edge. Last, the root r_R, which initially has no incoming edge is set as a target for a single d-edge during rule application (the incoming d-edge of r_L). Therefore, the lemma follows by induction on the number of rule applications.

Lemma 2 *A cDG $\mathcal{G}$ generated by the inference process is a cDAG.*

Proof We would like to show that $\mathcal{G}$ does not contain a cycle of e-edges. We prove by induction on the number of rule applications. The initial cDG created is a cDAG since it is composed of the initial trees, whose roots are the children of the forest root. We now show that applying a rule $L \to R$ to a cDAG $\mathcal{G}_n$, results in another cDAG $\mathcal{G}_{n+1}$.

Assume by contradiction that $\mathcal{G}_{n+1}$ now contains a simple cycle of e-edges C. Applying a rule does not add any e-edges between nodes in $\mathcal{G}_n$. Therefore, C must pass through r_R, the root of S_R and contain an e-edge (p, r_R). Since S_R is a tree, C must also leave S_R through an e-edge (u, v) ($u \in S_R$ and $v \notin S_R$). The cycle can be written as $p \to r_R \to ... \to u \to v \to ... \to p$. Notice that the path from v to p is fully contained in $\mathcal{G}_n$ since the cycle c is simple and entering S_R is possible only through r_R.

$L \to R$ must be a substitution rule, otherwise p would be the root of $\mathcal{G}_n$, which is impossible, since the root has no incoming d-edges. Therefore, r_R and r_L have the same single incoming d-edge, and the e-edge (p, r_L) exists in $\mathcal{G}_n$. In addition, u was added as a source node of a d-edge d in $\mathcal{G}_n$ since it is tied to some $u' \in S_L$, which is also a source node of d. Therefore, the path $r_L \to ... \to u' \to v$ exists in $\mathcal{G}_n$. Last, we know that the path from v to p is fully contained in $\mathcal{G}_n$, therefore we can construct a cycle $p \to r_L \to ... \to u' \to v \to ... \to p$ in $\mathcal{G}_n$, in contradiction to our assumption that $\mathcal{G}_n$ is a cDAG.

Lemma 2 allows us to restrict our focus to cDAGs only. We now define a generalization for embedded DAGs, which we term *embedded partial DAG*.

Definition An *embedded partial DAG* $G = (V, E)$ in a cDAG $\mathcal{G}$ rooted in a node $v \in \mathcal{V}$ is similar to an embedded DAG and is generated using the following process:

1. Initialize G with v alone
2. Repeat any number of iterations:
   (a) choose a node $s \in V$
   (b) choose a d-edge d that wasn't already chosen by s in a previous iteration. If all d-edges have been chosen - halt.
   (c) choose a target node $t \in T_d$ and add the e-edge $(s, t)_d$ to G.

Lemma 3 *Given a cDAG $\mathcal{G}$ generated by the inference process, the embedded partial DAGs rooted at any node n are all trees.*

Proof We prove by induction on the number of rule applications. In the initial cDAG, an embedded partial DAG is a subtree of some initial tree possibly with an e-edge from the root of $\mathcal{G}$ to the root of one of the initial trees. We now show that applying

a rule to a cDAG $\mathcal{G}_n$, resulting in a cDAG $\mathcal{G}_{n+1}$ maintains the desired invariant. We assume first that the rule is a substitution rule.

Assume by contradiction that after an application of a rule there is an embedded partial DAG T_{n+1} rooted at a node n that is not a tree. We can assume that n is not in S_R, otherwise, we can extend T_{n+1} by adding a path $p \rightarrow r_R \rightarrow ... \rightarrow n$, where p is a node outside S_R that is a source node of the incoming edge of r_R.

Since T_{n+1} is not a tree, there are two simple paths P_1 and P_2 from n that reach some node z from two different e-edges. z cannot be in S_R, since any two paths that meet in the subtree S_R, must first meet in its root r_R entering its incoming d-edge. But then we could construct in $\mathcal{G}_n$ the same two paths, only selecting r_L instead of r_R, in contradiction with the assumption of the induction. Clearly, either P_1 or P_2 must pass through the new subtree S_R, otherwise the two paths already existed in $\mathcal{G}_n$.

We first handle the case where, without loss of generality, P_1 passes through S_R and P_2 doesn't. P_1 passes through S_R and contains an e-edge (p, r_R). Since $z \notin S_R$, it also contains an e-edge (u, v) s.t. $u \in S_R$ and $v \notin S_R$. So P_1 can be written as $n \rightarrow ... \rightarrow p \rightarrow r_R \rightarrow ... \rightarrow u \rightarrow v \rightarrow ... \rightarrow z$. The paths from n to p and from v to z are in $\mathcal{G}_n$, because the only way to enter S_R is through r_R and P_1 is simple. We can now incrementally construct in $\mathcal{G}_n$ the following embedded partial DAG T_n: First, we construct P_2 and the section in P_1 from n to p as in T_{n+1}. Next, we expand p with the e-edge (p, r_L) instead of (p, r_R). We would like to expand T_n from r_L and reach z if possible.

As in lemma 2, u is tied to a node $u' \in S_L$ and therefore the e-edge (u', v) exists in $\mathcal{G}_n$. Therefore, there is a path P' from the root of S_L, r_L, through (u', v) to z. However, it is not guaranteed that the whole P' can be added to T_n. We try to expand T_n incrementally with P', at each step adding the next e-edge in the path. If we succeed, then T_n is an embedded graph in $\mathcal{G}_n$ with two paths to z, a contradiction.

If we fail, this can only be due to an e-edge $(z', t) \in P'$ we cannot add. Thus, z' must already be in P_2, and is a node for which there are two distinct paths in the embedded graph T_n, a contradiction. Notice that the path constructed is distinct from P_2 since it contains the e-edge (p, r_L) rather than r_R.

The remaining case is where both P_1 and P_2 pass through S_R and reach a node $z \notin S_R$. P_1 can be written as $n \rightarrow ... \rightarrow u_1 \rightarrow v_1 \rightarrow ... \rightarrow z$ and P_2 as $n \rightarrow ... \rightarrow u_2 \rightarrow v_2 \rightarrow ... \rightarrow z$, where u_1, $u_2 \in S_R$, and v_1, $v_2 \notin S_R$. Assume first that the e-edges (u_1, v_1) and (u_2, v_2) originate from the same d-edge d. Then $u_1 \neq u_2$, otherwise (u_1, v_1) and (u_2, v_2) could not be both in the same embedded partial DAG. u_1,u_2 are tied to the nodes u'_1, $u'_2 \in S_L$.

We show that $u'_1 \neq u'_2$: Assume by contradiction that $u'_1 = u'_2 = u'$. u' is tied to u_1 and u_2 due to alignment sharing or dual leaf variable sharing. u' cannot be tied to both u_1 and u_2 due to alignment sharing since alignment is a function from nodes in S_L to nodes in S_R. it cannot be tied to both due to dual leaf variable sharing since any variable appears only once in R. Last, if u' is tied to u_1 (without loss of generality) due to dual leaf variable sharing, then the d-edge d is part of the L. Therefore, u_2 will not include d as an aligned modifier, according to rule application rules, and thus u_2 will not be tied to u' due to alignment.

We can now construct an embedded graph T_n rooted at r_L in $\mathcal{G}_n$: Because S_L is part of the match of L in $\mathcal{G}_n$, we can construct an embedded graph rooted at r_L with a path to any node in S_L, in particular with paths to u'_1 and u'_2. Since $u'_1 \neq u'_2$, and both u'_1 and u'_2 are souce nodes of d, which is not part of S_L, we can expand these two paths with the e-edges (u'_1, v_1) and (u'_2, v_1) of d and get an embedded graph in $\mathcal{G}_n$ that is not a tree, a contradiction.

Now suppose that the e-edges (u_1, v_1) and (u_2, v_2) originate from different d-edges d_1 and d_2 respectively. u_1 and u_2 are tied to u'_1 and u'_2. Therefore, if $v_1 \neq v_2$ we can construct the following embedded graph T_n rooted at r_L: because S_L is part of

a match L, we can expand the paths in S_L from r_L to u'_1 and u'_2. Next, we add the e-edges (u'_1, v_1) of d_1 and (u'_2, v_2) of d_2. Recall that d_1 and d_2 are not in S_L and can therefore be used for expansion. We try to expand this embedded graph to include the paths from v_1 and v_2 to z. If we succeed, we have two paths in T_n leading to z. If we fail we have two paths in T_n meeting at some other node z', as explained above. Last, if $v_1 = v_2 = v$, then v is a node in $\mathcal{G}_n$ that has two incoming d-edges, contradicting lemma 1.

The case of an introduction rule is quite similar but simpler. If P_1 passes through S_R and P_2 doesn't, then n must be the root of the compact forest (the only node with a path to r_R). However, in this case n has a single outgoing d-edge, and therefore all its outgoing e-edges are disjoint (i.e. cannot be part of the same embedded DAG). Thus, P_2 must also pass through r_R - a contradiction. If both P_1 and P_2 pass through S_R, the proof is identical to the case of a substitution rule.

Corollary 1 *Two nodes in an embedded tree T generated by the inference process are never source nodes of the same d-edge.*

Proof Assume by contradiction that s and s' are source nodes of the same d-edge d in T. We can prune from T all subtrees rooted at nodes from T_d and get an embedded partial graph T' that does not contain d. Let t be a target node in T_d. We can expand s and s' through d with the e-edges (s, t) and (s, t') and obtain an embedded partial DAG that is not a tree, in contradiction to lemma 3.

Theorem 1 *The inference process generates a compact forest.*

Proof Lemma 2 guarantees that the inference process produces a cDAG $\mathcal{G}$. Lemma 3 guarantees that all of the embedded DAGs rooted at the root of $\mathcal{G}$, r (which are also embedded partial DAGs), are trees. $\mathcal{G}$ also has a single root. Hence, $\mathcal{G}$ is a compact forest. ∎

Theorem 2 *Given a rule base $\mathcal{R}$ and a set of initial trees T, a tree t is represented by a compact forest derivable from T by the inference process $\Leftrightarrow$ t is a consequent of T according to the inference formalism*

Proof ($\Leftarrow$) We first show completeness by induction on the number of rule applications n. if $n = 0$ then t is one of the initial trees and is represented by the initial compact forest. Let t_{n+1} be a tree derived in the formalism by applying a sequence of $n+1$ rules. We would like to show that t_{n+1} is represented in a derivable compact forest. t_{n+1} was derived by applying the rule $L \rightarrow R$ on the tree t_n. According to the inductive assumption t_n is represented in a compact forest $\mathcal{F}$ derivable by the inference process. Therefore, the rule $L \rightarrow R$ can be matched and applied in $\mathcal{F}$. We assume $L \rightarrow R$ is a substitution rule since the case of an introduction rule is similar. t_{n+1} is almost identical to t_n except it contains the subtree R instead of L with instantiated variables and aligned modifiers. It is easy to verify that after application of $L \rightarrow R$ on $\mathcal{F}$ resulting in $\mathcal{F}'$, $\mathcal{F}'$ will contain an embedded tree t that is almost identical to t_n except that the root of S_R, r_R, will be chosen instead of the root of S_L, r_L, and the rest of S_R can also be chosen with the appropriate instantiated variables and modifiers. Therefore $t_{n+1} = t$ is contained in $\mathcal{F}'$ as required. Notice that t is guaranteed to be a tree according to theorem 1.

($\Leftarrow$) Next, we prove soundness by induction on the number of rule applications in the forest. At initialization, all of the initial trees are consequents. Given a tree t_{n+1} represented by a compact forest $\mathcal{F}_{n+1}$ derived by $n+1$ rule applications, we would like to show that t_{n+1} is a consequent in the formalism.

If t_{n+1} was already represented by the compact forest after n rule applications, then according to the assumption of the induction it is a consequent in the formalism. If not, then t_{n+1} is a new embedded tree created after the application of the rule $L \rightarrow R$. Therefore, t_{n+1} contains the entire subtree S_R. We will now incrementally construct an embedded tree t_n represented by $\mathcal{F}_n$ such that t_{n+1} is the result of applying $L \rightarrow R$

on t_n.

For a substitution rule, we first construct the part of t_{n+1} that does not include the subtree rooted at r_R. For an introduction rule, we take any path from the forest's root to r_L. Next, we construct S_L through r_L instead of S_R through r_R. This is possible since according to theorem 1 all embedded graphs are trees and therefore the nodes of S_L are not already in t_n. We then look at the set of e-edges $(s,t) \in t_{n+1}$ such that $s \in S_R$ and $t \notin S_R$. Let (s,z) be such an edge originating from a d-edge d and S_z be the subtree rooted at z in t_{n+1}. Notice that S_z was already part of $\mathcal{F}_n$. s is tied to $s' \in S_L$ and therefore s' is a source node of d. We can expand t_n to include the edge (s',z) and S_z if s' is not already used with the d-edge d in t_n. This is guaranteed because d is not part of S_L (only d-edges that are not part of S_L are shared). Finally, we complete the construction of t_n by expanding arbitrarily any unused outgoing d-edge of t_n's nodes, until a complete embedded tree is obtained.

We have constructed an embedded tree t_n in $\mathcal{F}_n$. Therefore, according to the inductive assumption, t_n is a consequent in the formalism. t_n contains S_L and an instantiation of the dual leaf variables. Therefore, it is matched by L and the rule $L \rightarrow R$ can be applied. It is easy to verify that an application of the rule on t_n will yield t_{n+1}, as required. Thus, t_{n+1} is also a consequent in the formalism. ∎