NATURAL LANGUAGE INFERENCE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Bill MacCartney

June 2009

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Christopher D. Manning) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dan Jurafsky)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Stanley Peters)

Approved for the University Committee on Graduate Studies.

Abstract

Inference has been a central topic in artificial intelligence from the start, but while automatic methods for formal deduction have advanced tremendously, comparatively little progress has been made on the problem of *natural language inference* (NLI), that is, determining whether a natural language hypothesis h can justifiably be inferred from a natural language premise p. The challenges of NLI are quite different from those encountered in formal deduction: the emphasis is on informal reasoning, lexical semantic knowledge, and variability of linguistic expression. This dissertation explores a range of approaches to NLI, beginning with methods which are robust but approximate, and proceeding to progressively more precise approaches.

We first develop a baseline system based on overlap between bags of words. Despite its extreme simplicity, this model achieves surprisingly good results on a standard NLI evaluation, the PASCAL RTE Challenge. However, its effectiveness is limited by its failure to represent semantic structure.

To remedy this lack, we next introduce the Stanford RTE system, which uses typed dependency trees as a proxy for semantic structure, and seeks a low-cost alignment between trees for p and h, using a cost model which incorporates both lexical and structural matching costs. This system is typical of a category of approaches to NLI based on approximate graph matching. We argue, however, that such methods work best when the entailment decision is based, not merely on the degree of alignment, but also on global features of the aligned $\langle p, h \rangle$ pair motivated by semantic theory.

Seeking still greater precision, we devote the largest part of the dissertation to developing an approach to NLI based on *natural logic*. We present a new model of

natural logic which extends the monotonicity calculus of van Benthem and Sánchez-Valencia to incorporate semantic exclusion and implicativity. We define an expressive set of entailment relations—including representations of both semantic containment and semantic exclusion—and described the algebra of their joins. We then describe a model of compositional entailment capable of determining the entailment relation between two sentences connected by a sequence of edits. We introduce the concept of projectivity signatures, which generalizes the concept of monotonicity classes to cover the exclusion relations. And, we show how our framework can be used to explain inferences involving implicatives and non-factives.

Next, we describe the NatLog system, a computational implementation of our model of natural logic. NatLog is the first robust, general-purpose system for natural logic inference over real English sentences. The NatLog system decomposes an inference problem into a sequence of atomic edits which transforms p into h; predicts a lexical entailment relation for each edit using a statistical classifier; propagates these relations upward through a syntax tree according to semantic properties of intermediate nodes; and joins the resulting entailment relations across the edit sequence. We demonstrate the practical value of the NatLog system in evaluations on the FraCaS and RTE test suites.

Finally, we address the problem of alignment for NLI, by developing a model of phrase-based alignment inspired by analogous work in machine translation, including an alignment scoring function, inference algorithms for finding good alignments, and training algorithms for choosing feature weights.

Acknowledgments

To write a dissertation is a mighty undertaking, and I could not have reached the finish line without the influence, advice, and support of many colleagues, friends, and family. My thanks are due first and foremost to my advisor, Chris Manning, for being willing to take me on as a student, and for being unfailingly generous with his time and his support throughout my graduate career. Chris provided consistently good advice, helped me to situate my work in a broader context, and gave me tremendous freedom in pursuing my ideas, while always ensuring that I was making progress toward some fruitful outcome. I will be forever grateful for his help.

I am deeply obliged to the other members of my dissertation committee for their advice and encouragement. From Dan Jurafsky, I received a steady stream of positive feedback that helped to keep my motivation strong. Stanley Peters was of great help in working out various theoretical issues, and in making connections to the literature of formal semantics. I am indebted to Johan van Benthem not only for his pioneering work on the role of monotonicity in natural language inference—which was the direct inspiration for my model of natural logic—but also for pushing me to consider the theoretical implications of my work. Perceptive questions from Mike Genesereth also helped greatly to sharpen my ideas.

I owe special thanks to David Beaver, who first brought my attention to the topic of natural logic and the concept of monotonicity, and thereby set me on a path which culminated in this dissertation.

The faculty, students, and visitors involved with the Stanford NLP Group have provided a congenial and stimulating environment in which to pursue my studies. I am particularly indebted to my frequent collaborators—including Marie-Catherine

de Marneffe, Michel Galley, Teg Grenager, and many others—who shared my struggles and helped to shape my thoughts. I am also grateful to past officemates Iddo Lev, Roger Levy, Kristina Toutanova, Jeff Michels, and Dan Ramage for their warm companionship and for countless fascinating conversations.

Finally, I offer my deepest thanks to my siblings—Leslie, Jim, and Doug—whose love and support has sustained me; to my parents, Louis and Sharon, who instilled in me the love of learning that has fueled all my efforts; and to Destiny, who believed in me most when I believed in myself least.

Contents

List of Tables

List of Figures

Chapter 1

The problem of natural language inference

1.1 What is natural language inference?

Natural language inference (NLI) is the problem of determining whether a natural language hypothesis h can reasonably be inferred from a natural language premise p. Of course, inference has been a central topic in artificial intelligence (AI) from the start, and over the last five decades, researchers have made tremendous progress in developing automatic methods for formal deduction. But the challenges of NLI are quite different from those encountered in formal deduction: the emphasis is on informal reasoning, lexical semantic knowledge, and variability of linguistic expression, rather than on long chains of formal reasoning. The following example may help to illustrate the difference:

(1) p *Several airlines polled saw costs grow more than expected, even after adjusting for inflation.*

h *Some of the companies in the poll reported cost increases.*

In the NLI problem setting, (1) is considered a valid inference, for the simple reason that an ordinary person, upon hearing p, would likely accept that h follows. Note, however, that h is not a strict logical consequence of p: for one thing, *seeing*

cost increases does not necessarily entail *reporting* cost increases—it is conceivable that every company in the poll kept mum about increasing costs, perhaps for reasons of business strategy. That the inference is nevertheless considered valid in the NLI setting is a reflection of the informality of the task definition.

Although NLI involves recognizing an *asymmetric* relation of inferability between p and h, an important special case of NLI is the task of recognizing a *symmetric* relation of approximate semantic equivalence (that is, paraphrase) between p and h. (It is a special case because, if we have a system capable of determining whether h can be inferred from p, then we can detect semantic equivalence simply by running the system both "forwards" and "backwards".) Recognizing approximate semantic equivalence between *words* is comparatively straightforward, using manually constructed thesauri such as WordNet (Fellbaum et al. 1998) or automatically constructed thesauri such as that of Lin (1998). But the ability to recognize when two *sentences* are saying more or less the same thing is far more challenging, and if possible, could be of enormous benefit to many language processing tasks. We describe a few potential applications in the next section.

An intrinsic property of the NLI task definition is that the problem inputs are expressed in natural language. Research on methods for automated deduction, by contrast, typically assumes that the problem inputs are already expressed in some formal meaning representation, such as the language of first-order logic. This fact alone reveals how different the problem of NLI is from earlier work on logical inference, and places NLI squarely within the field of natural language processing (NLP): in developing approaches to NLI, we will be concerned with issues such as syntactic parsing, morphological analysis, word sense disambiguation, lexical semantic relatedness, and even linguistic pragmatics—topics which are the bread and butter of NLP, but are quite foreign to logical AI.

Over the last few years, there has been a surge of interest in the problem of NLI, centered around the PASCAL Recognizing Textual Entailment (RTE) Challenge (Dagan et al. 2005) and within the U.S. Government AQUAINT program. Researchers working on NLI can build on the successes achieved during the last decade in areas such as syntactic parsing and computational lexical semantics, and begin to tackle

the more challenging problems of sentence-level semantics.

1.2 Applications of NLI

The NLI task can serve as a stringent test of a system's language processing abilities. Any system which can reliably identify implications of natural language sentences must have a good understanding of how language works: it must be able to deal with all manner of linguistic phenomena and broad variability of semantic expression. Indeed, a capacity for reliable, robust, open-domain natural language inference is arguably a necessary condition for full natural language understanding (NLU), which for decades has been seen by many as the holy grail of NLP research. After all, a system which cannot identify the implications of a sentence cannot be said to understand the sentence. Some might make the stronger claim that a capacity for NLI is in fact a sufficient condition for NLU—that a system's ability to recognize the consequences of a sentence is all the evidence we need that it has understood the sentence. This claim, however, is difficult to embrace: one might argue that true understanding requires not merely recognizing consequences, but generating them, and moreover that full understanding of speaker meaning (as opposed to sentence meaning) depends on sophisticated models of discourse, pragmatics, human cognition, and world knowledge.

While full NLU remains a distant goal, a robust, reliable facility for NLI could enable a broad range of immediate applications. In this section we outline a few such applications.

Question answering. In open-domain question answering (QA), the challenge is to return a textual expression, extracted from a large collection of documents, which provides a good answer to a question posed in natural language, such as *Who was Lincoln's Secretary of State?* As Harabagiu and Hickl (2006) showed, an effective NLI system can serve as a key component of a QA system: it can be used to evaluate whether the target question (or some transformation of the question into a declarative form) can be inferred from candidate answers extracted from the source document.

For example, an NLI system should be able to recognize that ___ *was Lincoln's Secretary of State* can be inferred from the candidate answer *William H. Seward served as Secretary of State under President Abraham Lincoln.* A capacity for NLI is even more directly applicable to the CLEF Answer Validation Exercise,[1] in which each problem consists of a question, a proposed answer, and a supporting text, and the goal is simply to return a boolean value indicating whether the answer is correct for the question, given the text.

Semantic search. The goal of semantic search is to provide the ability to retrieve documents from a very large collection (such as the World Wide Web) based on the semantic content (rather than simply the surface words) of the documents and the search query. If a user searches for *people demonstrating against free trade*, most existing keyword-based search engines will return only documents containing the terms *demonstrating*, *free*, and *trade*. However, a document containing the sentence *Protestors chanted slogans opposing the agreement to drop trade barriers* might very well be responsive to the user's query, even though it fails to contain most of the search terms. If an NLI system were used to identify approximate semantic equivalence between search queries and sentences in source documents, then one could offer a form of semantic retrieval not available in current keyword-based search. (Of course, applying NLI to every document on the Web at query time is infeasible, so using NLI to enable semantic search will require clever approaches to "semantic indexing" and/or pre-filtering of candidate documents.)

Automatic summarization. A key challenge in automatic summarization is the elimination of redundancy. This is especially so in *multi-document summarization*, where the summary is constructed from multiple source documents, such as a collection of news stories describing the same event. Here, the ability of an effective NLI system to recognize sentence-level semantic equivalence can therefore be enormously helpful: NLI can be used to ensure that the summary does not contain any sentences

[1] `http://nlp.uned.es/clef-qa/ave/`

that can be inferred from the rest of the summary. An even more basic goal of summarization is *correctness*, i.e., the summary must accurately reflect the content of the source document(s). In *extractive summarization* (Das and Martins 2007), where the summary is constructed from snippets extracted from the source document(s), this is rarely an issue. But whether or not the extractive strategy is used, NLI can be useful in ensuring correctness, by checking that the the summary is implied by the source document(s). The application of NLI to summarization has been explored by Lacatusu et al. (2006).

Evaluation of machine translation systems. A relatively new application for NLI is the automatic evaluation of the output of machine translation (MT) systems, explored by Padó et al. (2009). Rapid iterative development of MT systems depends critically upon automatic evaluation measures. In the past, MT researchers have relied upon evaluations such as BLEU, which measures n-gram overlap between a candidate translation and human-generated reference translations for the same sentence. However, researchers have long bemoaned the limitations of BLEU and its cousins: because such scoring metrics operate over surface forms, they are unable to accommodate syntactic and semantic reformulations (Callison-Burch et al. 2006). An effective NLI system can mitigate this problem, by assessing approximate semantic equivalence between a candidate translation and a reference translation: if the candidate entails (and is entailed by) the reference, then it is probably a good translation, even if its surface form is quite different from the reference.

1.3 NLI task formulations and problem sets

Although we have presented NLI as a single well-defined task, in fact the problem of NLI has been formulated in a number of different ways over time, by different researchers with different aims. Correspondingly, a number of different NLI problem sets have been constructed, with varying characteristics. While the construction of such problem sets has generally been guided by a preconceived notion of the task to be solved, to some extent each of available problem sets provides an implicit task

definition: for a particular problem set, the objective is simply to answer as many problems correctly as possible.

In this section we survey various formulations of the NLI task, and introduce NLI problem sets to which we will return in later chapters.

1.3.1 The FraCaS test suite

The FraCaS test suite of NLI problems (Cooper et al. 1996) was one product of the FraCaS Consortium, a large collaboration in the mid-1990s aimed at developing a range of resources related to computational semantics. It contains 346 NLI problems, each consisting of one or more premise sentences, (usually) followed by a question sentence and an answer. While the FraCaS test suite expresses the "goal" of each problem as a question, the standard formulation of the NLI task involves determining the entailment relation between a premise and a declarative hypothesis. Thus, for the purpose of this work, we converted each FraCaS question into a declarative hypothesis, using a process described in greater detail in section 7.8.1.

The FraCaS problems contain comparatively simple sentences, and the premise and question/hypothesis sentences are usually quite similar. Despite this simplicity, the problems are designed to cover a broad range of semantic and inferential phenomena, including quantifiers, plurals, anaphora, ellipsis, adjectives, comparatives, temporal reference, verbs, and propositional attitudes. Figure 1.1 shows a representative selection of FraCaS problems.

Most FraCaS problems are labeled with one of three answers: YES means that the hypothesis can be inferred from the premise(s), NO means that the hypothesis contradicts the premise(s), and UNK means that the hypothesis is compatible with (but not inferable from) the premise(s). The distribution of answers is *not* balanced: about 59% of the problems have answer YES, while 28% have answer UNK, and 10% have answer NO.[2]

About 45% of the FraCaS problems contain multiple premises. Most of these have just two premises, but some have more, and one problem has five. Figure 1.2 shows

[2]The remaining 3% of problems cannot be straightforwardly assigned to one of these three labels. See section 7.8.1 for details.

§1: Quantifiers			
38	p	No delegate finished the report.	
	h	Some delegate finished the report on time.	NO
48	p	At most ten commissioners spend time at home.	
	h	At most ten commissioners spend a lot of time at home.	YES
§2: Plurals			
83	p	Either Smith, Jones or Anderson signed the contract.	
	h	Jones signed the contract.	UNK
§3: Anaphora			
141	p	John said Bill had hurt himself.	
	h	Someone said John had been hurt.	UNK
§4: Ellipsis			
178	p	John wrote a report, and Bill said Peter did too.	
	h	Bill said Peter wrote a report.	YES
§5: Adjectives			
205	p	Dumbo is a large animal.	
	h	Dumbo is a small animal.	NO
§6: Comparatives			
233	p	ITEL won more orders than APCOM.	
	h	ITEL won some orders.	YES
§7: Temporal reference			
258	p	In March 1993 APCOM founded ITEL.	
	h	ITEL existed in 1992.	NO
§9: Attitudes			
335	p	Smith believed that ITEL had won the contract in 1992.	
	h	ITEL won the contract in 1992.	UNK

Figure 1.1: Some single-premise inference problems from the FraCaS test suite.

§6: Comparatives			
238	p	ITEL won twice as many orders than APCOM. APCOM won ten orders.	
	h	ITEL won twenty orders.	YES
§7: Temporal reference			
284	p	Smith wrote a report in two hours. Smith started writing the report at 8 am.	
	h	Smith had finished writing the report by 11 am.	YES

Figure 1.2: Some multiple-premise inference problems from the FraCaS test suite.

some examples of FraCaS problems with multiple premises.

The FraCaS test suite will be described in greater detail in section 7.8.1.

1.3.2 Recognizing Textual Entailment (RTE)

A more recent, and better-known, formulation of the NLI task is the Recognizing Textual Entailment (RTE) Challenge, which has been organized every year for the past four years (Dagan et al. 2005, Bar-Haim et al. 2006, Giampiccolo et al. 2007; 2008).[3] The RTE Challenge was initiated under the auspices of the European Commission's PASCAL project, but in 2008 found a new home as part of the NIST Text Analysis Conference. In each year of the RTE Challenge, organizers have produced a test set containing several hundred NLI problems; in most years they have also released a large development set. (See table 1.1 for details.)

Each RTE problem consists of a premise p, a hypothesis h, and an answer label. The premises were collected "in the wild" from a variety of sources, commonly from newswire text. They tend to be fairly long (averaging 25 words in RTE1, 28 words in RTE2, 30 words in RTE3, and 39 words in RTE4), and sometimes contain more than one sentence (a trend which has also increased over time). The hypotheses, by contrast, are single sentences, manually constructed for each premise, and are quite

[3]There will be a fifth round of the RTE Challenge in 2009.

name	year	sponsor	development set	test set
RTE1	2005	PASCAL	576 problems	800 problems
RTE2	2006	PASCAL	800 problems	800 problems
RTE3	2007	PASCAL	800 problems	800 problems
RTE4	2008	NIST	—	1000 problems

Table 1.1: High-level characteristics of the RTE problem sets.

short (averaging 11 words in RTE1, 8 words in RTE2, and 7 words in RTE3 and RTE4). Examples of RTE problems are shown in figures 4.1 and 7.4.

In the first three years, the RTE Challenge was presented as a binary classification task: the goal was simply to determine whether p entailed h (answer YES) or not (answer NO). Problem sets were designed to be balanced, containing equal numbers of YES and NO answers. Beginning with RTE4, participants were encouraged (but were not obliged) to make three-way predictions, distinguishing cases in which h contradicts p from those in which h is compatible with, but not entailed by, p (as in the FraCaS evaluation).

Several characteristics of the RTE problems should be emphasized. Examples are derived from a broad variety of sources, including newswire; therefore systems must be domain-independent. The inferences required are, from a human perspective, fairly superficial: no long chains of reasoning are involved. However, there are "trick" questions expressly designed to foil simplistic techniques. (Problem 2081 of figure 4.1 is a good example.) The definition of entailment is informal and approximate: whether a competent speaker with basic knowledge of the world would typically infer h from p. Entailments will certainly depend on linguistic knowledge, and may also depend on world knowledge; however, the scope of required world knowledge is left unspecified.

Despite the informality of the problem definition, human judges exhibit very good agreement on the RTE task, with agreement rate of 91–96% (Dagan et al. 2005). In principle, then, the upper bound for machine performance is quite high. In practice, however, the RTE task is exceedingly difficult for computers. Participants in the first PASCAL RTE workshop reported accuracy from 50% to 59% (Dagan et al.

2005). In later RTE competitions, higher accuracies were achieved, but this is partly attributable to the RTE test sets having become intrinsically easier—see section 2.2 for discussion.

1.4 Previous approaches to NLI

Past work on the problem of NLI has explored a wide spectrum of approaches, ranging from robust-but-shallow approaches based on approximate measures of semantic similarity, to deep-but-brittle approaches based on full semantic interpretation. In this section we characterize these contrasting approaches, and point the way toward a middle ground.

1.4.1 Shallow approaches

To date, the most successful NLI systems have relied on simple surface representations and approximate measures of lexical and syntactic similarity to ascertain whether the meaning of h is subsumed by the meaning of p. This category of shallow approaches includes systems based on lexical or semantic overlap (Glickman et al. 2005), pattern-based relation extraction (Romano et al. 2006), or approximate matching of predicate-argument structure (MacCartney et al. 2006, Hickl et al. 2006).

As example (1) in section 1.1 demonstrates, full semantic interpretation is often not necessary to determining inferential validity. For example, a bag-of-words model like that of Glickman et al. (2005) would approach example (1) by matching each word in h to the word in p with which it is most similar, matching *Some* to *Several*, *companies* to *airlines*, *poll* to *polled*, *reported* to *saw*, *cost* to *costs*, and *increases* to *grow*. Since most words in h can be matched quite well to a word in p (the most dubious match is that of *reported* to *saw*), a bag-of-words model would likely (and rightly) predict that the inference in (1) is valid. (The bag-of-words approach is explored in greater depth in chapter 2.)

The bag-of-words approach is robust and broadly effective—but it's terribly imprecise, and is therefore easily led astray. Consider problem 2081 in figure 4.1. Clearly,

the inference is invalid, but the vanilla bag-of-words model cannot recognize this. Not only does every word in h also appear in p, but in fact the whole of h is an exact substring of p. A related problem with the bag-of-words model is that, because it ignores predicate-argument structure, it can't distinguish between *Booth shot Lincoln* and *Lincoln shot Booth*.

Both of these shortcomings can be mitigated by including syntactic information in the matching algorithm, an approach exemplified by the Stanford RTE system, which we'll describe more fully in chapter 4. Nevertheless, all shallow approaches will struggle with such commonplace phenomena as antonymy, negation, non-factive contexts, and verb-frame alternation. And, crucially, they all depend on an assumption of upward monotonicity.[4] To see how this can go wrong, consider changing the quantifiers in our example from *Several* and *Some* to *Every* and *All* (or to *Most*, or to *No* and *None*). The lexical similarity of h to p will scarcely be affected, but the inference will no longer be valid, because the poll may have included automakers who did not report cost increases. And constructions which are not upward monotone are surprisingly widespread—they include not only negation (*not*) and many quantifiers (e.g., *less than three*), but also conditionals (in the antecedent), superlatives (e.g., *tallest*), and countless other prepositions (e.g., *without*), verbs (e.g., *avoid*), nouns (e.g., *denial*), adjectives (e.g., *unable*), and adverbs (e.g., *rarely*). In order properly to handle inferences involving these phenomena, an approach with greater precision is required.

1.4.2 Deep approaches

For a semanticist, the most obvious approach to NLI relies on full semantic interpretation: first, translate p and h into some formal meaning representation, such as first-order logic (FOL), and then apply automated reasoning tools to determine inferential validity. This kind of approach has the power and precision we need to handle

[4] If a linguistic expression occurs in an *upward monotone* context (the default), then replacing it with a more general expression preserves truth; if it occurs in an *downward monotone* context (such as under negation), then replacing it with a more specific expression preserves truth. For a fuller explanation, see section 6.2.1.

negation, quantifiers, conditionals, and so on, and it can succeed in restricted domains, but it fails badly on open-domain NLI evaluations such as RTE. The difficulty is plain: truly *natural* language is fiendishly complex, and the full and accurate translation of natural language into formal representations of meaning presents countless thorny problems, including idioms, ellipsis, anaphora, paraphrase, ambiguity, vagueness, aspect, lexical semantics, the impact of pragmatics, and so on.

Consider for a moment the difficulty of fully and accurately translating the premise of example (1) into first-order logic. We might choose to use a neo-Davidsonian event representation (though this choice is by no means essential to our argument). Clearly, there was a polling event, and some airlines were involved. But what is the precise meaning of *several*? Would three airlines be considered several? Would one thousand? Let's agree to leave these comparatively minor issues aside. We also have some costs, and a growing event of which the costs are the subject (a single growing event, or one for each airline?). And the growth has some quantity, or magnitude, which exceeds some expectation (one expectation, or many?), held by some agent (the airlines?), which was itself relative to some notion (held by whom?) of inflation (of what, and to what degree?). This is beginning to seem like a quagmire—yet this is absolutely the norm when interpreting real English sentences, as opposed to toy examples.

The formal approach faces other problems as well. Many proofs can't be completed without axioms encoding background knowledge, but it's not clear where to get these. And let's not forget that many inferences considered valid in the NLI setting (such as (1)) are not actually strict logical consequences. Bos and Markert (2005b) provided a useful reality check when they tried to apply a state-of-the-art semantic interpreter to the RTE1 test set—they were able to find a formal proof for fewer than 4% of the problems (and one-quarter of those proofs were incorrect). Thus, for the problem of open-domain NLI, the formal approach is a nonstarter. To handle inferences requiring greater precision than the shallow methods described in section 1.4.1, we'll need a new approach.

1.5 The natural logic approach to NLI

In the second half of this dissertation, we'll explore a middle way, by developing a model of what Lakoff (1970) called *natural logic*,[5] which he defined as a logic whose vehicle of inference is natural language. Natural logic thus characterizes valid patterns of inference in terms of syntactic forms which are as close as possible to surface forms. For example, the natural logic approach might sanction the inference in FraCaS problem 48 (figure 1.1) by observing that: in *downward monotone* contexts, inserting an intersective modifier preserves truth; *a lot of* is an intersective modifier; and *At most ten* is a downward monotone quantifier. Natural logic thus achieves the semantic precision needed to handle inferences like FraCaS problem 48, while sidestepping the difficulties of full semantic interpretation.

The natural logic approach has a very long history,[6] originating in the syllogisms of Aristotle and continuing through the medieval scholastics and the work of Leibniz. It was revived in recent times by van Benthem (1988; 1991) and Sánchez Valencia (1991), whose *monotonicity calculus* explains inferences involving semantic containment and inversions of monotonicity, even when nested, as in *Nobody can enter without a valid passport* $\models$ *Nobody can enter without a passport.* However, because the monotonicity calculus lacks any representation of semantic exclusion, it fails to explain many simple inferences, such as *Stimpy is a cat* $\models$ *Stimpy is not a poodle*, or FraCaS problem 205 (figure 1.1).

Another model which arguably belongs to the natural logic tradition (though not presented as such) was developed by Nairn et al. (2006) to explain inferences involving implicatives and factives, even when negated or nested, as in *Ed did not forget to force Dave to leave* $\models$ *Dave left.* While the model bears some resemblance to the monotonicity calculus, it does not incorporate semantic containment or explain interactions between implicatives and monotonicity, and thus fails to license inferences such as *John refused to dance* $\models$ *John didn't tango.*

[5] Natural logic is not to be confused with *natural deduction*, a proof system for first-order logic.

[6] For a useful overview of the history of natural logic, see van Benthem (2008). For recent work on theoretical aspects of natural logic, see (Fyodorov et al. 2000, Sukkarieh 2001, van Eijck 2005).

In chapter 6, we'll present a new theory of natural logic which extends the monotonicity calculus to account for negation and exclusion, and also incorporates elements of Nairn's model of implicatives.

1.6 Overview of the dissertation

In this dissertation, we explore a range of approaches to NLI, beginning with methods which are robust but approximate, and proceeding to progressively more precise approaches.

We begin in chapter 2 by developing a baseline system based on overlap between bags of words, an approach introduced in section 1.4.1. Despite its extreme simplicity, this model achieves surprisingly good results on an evaluation using the RTE data sets (introduced in section 1.3.2). However, its effectiveness is limited by its failure to represent semantic structure.

Next, we consider the problem of alignment for NLI (chapter 3). We examine the relation between NLI alignment and the similar problem of alignment in machine translation (MT). We develop a new phrase-based model of alignment for NLI—the MANLI system—which is inspired by analogous work in MT alignment, and includes an alignment scoring function, inference algorithms for finding good alignments, and training algorithms for choosing feature weights. We also undertake the first comparative evaluation of various MT and NLI aligners on an NLI alignment task, and show that the MANLI system significantly outperforms its rivals.

To remedy the shortcomings of the bag-of-words model, we next introduce the Stanford RTE system (chapter 4), which uses typed dependency trees as a proxy for semantic structure, and seeks a low-cost alignment between trees for p and h, using a cost model which incorporates both lexical and structural matching costs. This system is typical of a category of approaches to NLI based on approximate graph matching. We argue, however, that such methods work best when the entailment decision is based, not merely on the degree of alignment, but also on global features of the aligned $\langle p, h \rangle$ pair motivated by semantic theory.

Seeking still greater precision, we devote the largest part of the dissertation to developing an approach to NLI based on natural logic. In chapters 5 and 6, we present a new model of natural logic which extends the monotonicity calculus of van Benthem and Sánchez-Valencia to incorporate semantic exclusion and implicativity. In chapter 5, we define an expressive set of entailment relations—including representations of both semantic containment and semantic exclusion—and described the algebra of their joins. Then, in chapter 6, we describe a model of compositional entailment capable of determining the entailment relation between two sentences connected by a sequence of edits. We introduce the concept of projectivity signatures, which generalizes the concept of monotonicity classes to cover the exclusion relations. And, we show how our framework can be used to explain inferences involving implicatives and non-factives.

In chapter 7, we describe the NatLog system, a computational implementation of our model of natural logic. NatLog is the first robust, general-purpose system for natural logic inference over real English sentences. The NatLog system decomposes an inference problem into a sequence of atomic edits which transforms p into h; predicts a lexical entailment relation for each edit using a statistical classifier; propagates these relations upward through a syntax tree according to semantic properties of intermediate nodes; and joins the resulting entailment relations across the edit sequence. We demonstrate the practical value of the NatLog system in evaluations on the FraCaS and RTE test suites.

Finally, in chapter 8, we summarize the contributions of the dissertation, and offer some thoughts on the future of natural language inference.

Chapter 2

The bag-of-words approach

How can we begin to approach the task of building a working model of natural language inference? For logicians and semanticists, the most obvious approach relies on full semantic interpretation: translate the premise p and hypothesis h of an NLI problem into a formal meaning representation, such as first-order logic, and then apply automated reasoning tools. However, the recent surge in interest in the problem of NLI was initiated by researchers coming from the information retrieval (IR) community, which has exploited very lightweight representations, such as the the *bag-of-words* representation, with great success. If we approach the NLI task from this direction, then a reasonable starting point is to represent p and h simply by (sparse) vectors encoding the counts of the words they contain, and then to predict inferential validity using some measure of vector similarity. Although the bag-of-words representation might seem unduly impoverished, bag-of-words models have been shown to be surprisingly effective in addressing a broad range of NLP tasks, including word sense disambiguation, text categorization, and sentiment analysis.

We first alluded to the bag-of-words approach in section 1.4, where we described it as one end of a spectrum of approaches to the NLI problem. Indeed, the bag-of-words approach exemplifies what is perhaps the most popular and well-explored category of approaches to NLI: those which operate solely by measuring approximate lexical similarity, without regard to syntactic or semantic structure. Our goal in this chapter is not to propose any particularly novel concepts or techniques; rather, we aim merely

to flesh out the details of a specific way of implementing the bag-of-words approach, to evaluate its effectiveness on standard NLI test suites, and thereby to establish a baseline against which to compare later results, both for alignment (chapter 3) and for entailment prediction (chapters 4 and 7).

2.1 A simple bag-of-words entailment model

In this section, we describe a very simple probabilistic model of natural language inference which relies on the bag-of-words representation. It thus ignores altogether the syntax—and even the word order—of the input sentences, and makes no attempt at semantic interpretation. The model depends only on some measure of lexical similarity between individual words. The precise similarity function used is not essential to the model; the choice of similarity function can be viewed as a model parameter. Despite its simplicity, this model achieves respectable results on standard evaluations of natural language inference, even when using a very crude lexical similarity function.

2.1.1 Approach

Our approach is directly inspired by the probabilistic entailment model described in Glickman et al. (2005). Let $P(h|p)$ denote the probability that premise p supports an inference to (roughly, *entails*) hypothesis h. We assume that h is supported by p only to the degree that each individual word h_j in h is supported by p. We also assume that the probability that a given word in h is supported is independent of whether any other word in h is supported. We can thus factor the probability of entailment as follows:

$$P(h|p) = \prod_j P(h_j|p)$$

In addition, we assume that each word in h derives its support chiefly from a single word in p. Consequently, the probability that a given word in h is supported by p can be identified with the max over the probability of its support by the individual

words of p:

$$P(h_j|p) = \max_i P(h_j|p_i)$$

By this decomposition, we have expressed the overall probability that p supports h in terms of a function of the probabilities of support between pairs of individual words p_i and h_j:

$$P(h|p) = \prod_j \max_i P(h_j|p_i)$$

The expression $P(h_j|p_i)$ can be interpreted as a sort of *lexical entailment score* between words p_i and h_j. We make no attempt to explain what it might mean for words (as opposed to declarative propositions) to stand in an entailing relationship. Rather, we focus on the purely pragmatic goal of choosing a lexical scoring function which does a good job of predicting entailment between p and h. Glickman et al. (2005) used a lexical scoring function based on web co-occurrence statistics. The model we'll describe in the following pages uses a much more simple-minded lexical scoring function. However, we'll consider other ways of enhancing the model's effectiveness, including weighting the words in h by their frequency in a large corpus; generating not only a "forward entailment" score $P(h|p)$ but also a "reverse entailment" score $P(p|h)$; and combining these scores in a maximum entropy model.

Note that, by matching each word in h to the word in p which best supports it, this approach can also be viewed as inducing an *alignment* between the words of h and p, akin to word alignments in statistical machine translation (Brown et al. 1993). In fact, this model is closely analogous to the "heuristic model" of word alignment described by Och and Ney (2003).

2.1.2 The lexical scoring function

The model's most important parameter is a *lexical scoring function*, which maps ordered pairs of words to real values in the interval $[0, 1]$ which indicate the degree to which one word "supports" the other. The model itself leaves unspecified the details of how such scores should be interpreted and calibrated. But, intuitively, a good lexical scoring function should assign a score close (or equal) to 1 to a pair of similar

(or identical) words, and will assign a score close to 0 to a pair of dissimilar words.

The best candidates for lexical scoring functions will therefore be measures of lexical similarity, or perhaps the somewhat weaker notion of lexical relatedness. Candidate scoring functions might include:

- measures of string similarity
- similarity functions based on vector-space models, such as latent semantic analysis (LSA) (Landauer and Dumais 1997)
- measures of distributional similarity, like that proposed by Lin (1998)
- taxonomy-based scoring functions, such as one based on the WordNet-based semantic distance measure of Jiang and Conrath (1997)

Alternatively, a lexical scoring function might be designed as a hybrid, which combines information from several component scoring functions. One advantage of a hybrid scoring function is that it can incorporate information from low-coverage, high-precision scoring functions (for example, lexical resources providing information about acronyms) which would be inappropriate for use by themselves. However, the design of a hybrid scoring function presents several problems. How should the component scores be combined: via the max function, or perhaps as a weighted average? Another difficulty is that the component functions may produce very different distributions of scores, particularly if some are designed as low-recall, high-precision measures (e.g., acronym resources), while others are designed as high-recall, low-precision measures (e.g., distributional similarity). How do we define and implement a suitable calibration of scores between such component functions, so that scores from different component functions can meaningfully be compared?

Note that while many lexical scoring functions will be symmetric, this is not strictly required. For example, if we have access to a lexical resource containing hyponymy relations (such as WordNet (Fellbaum et al. 1998)), we may wish to define a scoring function which assigns a high score to hyponym pairs (such as $\langle mammal, horse \rangle$) but not to hypernym pairs (such as $\langle horse, mammal \rangle$).

While the selection (or design) of a lexical scoring function presents many difficult choices, in this chapter we'll sidestep this complexity. Instead, we'll develop a model

based on an extremely simple-minded lexical scoring function. Our *lemma string similarity* function computes a lexical similarity score based on the Levenshtein string edit distance (Levenshtein 1966) between the lemmas (base forms) of the input words. In order to achieve the appropriate normalization, we divide the edit distance by the maximum of the length of the two lemmas, and subtract the result from 1:

$$sim(w_1, w_2) = 1 - \frac{dist(lemma(w_1), lemma(w_2))}{max(|lemma(w_1)|, |lemma(w_2)|)}$$

(Here $lemma(w)$ denotes the lemma of word w; $dist()$ denotes Levenshtein string edit distance; and $|\cdot|$ denotes string length.)

2.1.3 Per-word alignment costs

In the following discussion, it will be convenient to work in terms of costs, rather than scores. Given our lexical scoring function sim, we can define a *lexical cost function*, which expresses the cost of aligning hypothesis word h_j with premise word p_i as the negative logarithm of their similarity score:

$$cost(h_j, p_i) = -\log sim(h_j, p_i)$$

The *cost* function has range $[0, \infty]$: aligning equal words will have cost 0; aligning very different words will have very high cost. And, of course, if sim is symmetric, then *cost* will be symmetric as well.

Following the approach described in section 2.1.1, the cost of aligning a given hypothesis word h_j to the premise p will be the minimum of the costs of aligning h_j to each possible premise word p_i:

$$cost(h_j|p) = \min_i cost(h_j, p_i)$$

(Note that $cost(h_j|p)$ is *asymmetric*, even if $cost(h_j, p_i)$ is symmetric—this is the reason for the conditioning bar.)

We can improve robustness by putting an arbitrary upper bound, *maxcost*, on

word alignment costs; otherwise, a single hard-to-align hypothesis word can completely block inference.

$$cost(h_j|p) = \min(maxcost, \min_i cost(h_j, p_i))$$

Lower values for *maxcost* make the model "looser": since every word can be aligned more cheaply, we're more likely to predict equivalence between the premise and the hypothesis. Conversely, higher values for *maxcost* make the model "stricter". One interpretation of *maxcost* is that it represents the cost of inserting a word into h which doesn't correspond to anything in p.

2.1.4 Total alignment costs

The overall cost of aligning h to p can now be computed as the sum of the costs of aligning the individual words in h.

$$cost(h|p) = \sum_j cost(h_j|p) = \sum_j \min(maxcost, \min_i cost(h_j, p_i))$$

However, intuition suggests that we would like to be able to treat some words in h as more important than others. It is of no great consequence if we are unable cheaply to align small function words (such as *the*, *for*, or *as*) in h; it matters much more whether we can find a suitable alignment for big, rare words (such as *Ahmadinejad*). To capture this intuition, we can define a weight function which represents the importance of each word in alignment as a non-negative real value. We then define the total cost of aligning h to p to be the weighted sum of the costs of aligning the individual words in h:

$$cost(h|p) = \sum_j weight(h_j) \cdot cost(h_j|p)$$

Just as there were many possible choices for the lexical scoring function, we have many possible choices for this lexical weight function $weight(h_j)$. Clearly, we want common words to receive less weight, and rare words to receive higher weight. A

plausible weight function might also take account of the part of speech of each word. However, to keep things simple, we'll use a weight function based on the *inverse document frequency* (IDF) of each word in a large corpus. If N is the number of documents in our corpus and N_w is the number of documents containing word w, then we define:

$$idf(w) = -\log(N_w/N)$$

Since the value of this function will be infinite for words which do not appear in the corpus, we'll define our weight function to place an upper bound on weights. We'll scale IDF scores into the range $[0, 1]$ by dividing each score by the highest score observed in the corpus (typically, this is the score of words observed exactly once), and we'll assign weight 1 to words not observed in the corpus. If W is the set of words observed in the corpus, then we define the weight function by:

$$weight(h_j) = \min\left[1, \frac{idf(h_j)}{\max_{w \in W} idf(w)}\right]$$

For example, when based on a typical corpus (the English Gigaword corpus), this function assigns weight 0.0004 to *the*, 0.1943 to *once*, 0.6027 to *transparent*, and 1.000 to *Ahmadinejad*.

2.1.5 Predicting entailment

Let's return to our top-level concern: how do we predict whether an inference from p to h is valid? Now that we have defined the function $cost(h|p)$ expressing the cost of aligning h to p, we have two possible avenues. The first is to use $cost(h|p)$ directly, following the approach described in section 2.1.1, to assign a probability to the entailment from p to h:

$$P(h|p) = \exp(-cost(h|p))$$

However, this method is rather inflexible: it utilizes only one measure of the overlap between p and h, and provides no automatic way to tune the threshold above which

to predict entailment.

The second is to use features based on our cost function as input to a machine learning classifier. Rather than using $cost(h|p)$ directly to predict entailment, we construct a feature representation of the entailment problem using quantities derived from the *cost* function, and train a statistical classifier to predict entailment using these feature representations as input. For this purpose, we used a simple maximum entropy (logistic regression) classifier with Gaussian regularization (and regularization parameter $\sigma = 1$).

One advantage of this approach is that it enables us automatically to determine the appropriate threshold beyond which to predict entailment. Another is that it allows us to combine different kinds of information about the similarity between p and h. For example, our feature representation can include not only the cost of aligning h to p, but also the reverse, that is, the cost of aligning p to h. (If p and h are nearly identical, then both $cost(h|p)$ and $cost(p|h)$ should be low; whereas if p subsumes h but also includes extraneous content—a common situation in NLI problems—then $cost(h|p)$ should be low, but $cost(p|h)$ should be high.) We experimented with a number of different features based on the *cost* function, shown in figure 2.1. Intuitively, `feqScore` is intended to represent the likelihood that p entails h; `reqScore`, that h entails p; `eqvScore`, that p and h are mutually entailing (i.e., equivalent); `fwdScore`, that p entails h and not vice-versa; `revScore`, that h entails p and not vice-versa; and `indScore`, that neither p and h entails the other (i.e., they are independent). However, these interpretations are merely suggestive—one of the virtues of the machine learning approach is that a (suitably regularized) classifier can learn to exploit whichever features carry signal, and ignore the rest, whether or not the features well capture the intended interpretations.

Clearly, there is considerable redundancy between these features, but because maximum entropy classifiers (unlike, say, Naïve Bayes classifiers) handle correlated features gracefully, this is not a source of concern. However, in experiments, we found little benefit to including all of these features; using just the `feqScore` and `reqScore` features gave results roughly as good as including any additional features.

$$
\begin{aligned}
\texttt{hCost} &= \mathit{cost}(h|p) \\
\texttt{pCost} &= \mathit{cost}(p|h) \\
\texttt{feqScore} &= \exp(-\texttt{hCost}) \\
\texttt{reqScore} &= \exp(-\texttt{pCost}) \\
\texttt{eqvScore} &= \texttt{feqScore} \cdot \texttt{reqScore} \\
\texttt{fwdScore} &= \texttt{feqScore} \cdot (1 - \texttt{reqScore}) \\
\texttt{revScore} &= (1 - \texttt{feqScore}) \cdot \texttt{reqScore} \\
\texttt{indScore} &= (1 - \texttt{feqScore}) \cdot (1 - \texttt{reqScore})
\end{aligned}
$$

Figure 2.1: Possible features for a maximum-entropy model entailment based on the bag-of-words model.

2.2 Experimental results

Despite its simplicity, the bag-of-words model described in section 2.1 yields respectable performance when applied to real NLI problems such as those in the RTE test suites. In order to establish a baseline against which to compare results in later chapters, we performed experiments designed to quantify the effectiveness of the bag-of-words approach. In particular, we used the simple lemma string similarity function described in section 2.1.2 as our lexical scoring function; we placed an upper bound of 10 on per-word alignment costs, as described in section 2.1.3; we combined per-word alignment costs using IDF weighting, as described in section 2.1.4; and we made entailment predictions using a maximum-entropy classifier with Gaussian regularization, using only the features `feqScore` and `reqScore`, as described in section 2.1.5.

Table 2.1 shows the results of evaluating this model on various combinations of training and testing data from the RTE test suites. For comparison, table 2.2 shows results from teams participating in the first three RTE competitions. The figures lead us to a number of surprising observations.

First, the bag-of-words model performs unexpectedly well, especially in light of the fact that it is based on a simple-minded measure of string similarity. While it is not competitive with the best RTE systems, it achieves an accuracy comparable

Training data	Test data	% YES	P %	R %	Acc %
RTE1 dev	RTE1 dev	49.7	53.9	53.7	54.0
	RTE1 test	50.1	53.6	53.8	53.6
RTE2 dev	RTE2 dev	52.2	56.7	59.2	57.0
	RTE2 test	52.4	57.3	60.0	57.6
RTE3 dev	RTE3 dev	50.6	70.1	68.9	68.9
	RTE3 test	57.5	62.4	70.0	63.0
RTE3 test	RTE3 dev	46.9	71.2	64.8	68.4
	RTE3 test	53.8	63.0	66.1	62.8

Table 2.1: Performance of the bag-of-words entailment model described in section 2.1 on various combinations of RTE training and test data (two-way classification). The columns show the training data used, the test data used, the proportion of YES predictions, precision and recall for the YES class, and accuracy.

Evaluation	System	Acc %
RTE1 test	Best: Glickman et al. 05	58.6
	Average (13 teams)	54.8
	Median (13 teams)	55.9
RTE2 test	Best: Hickl et al. 06	75.4
	Average (23 teams)	59.8
	Median (23 teams)	59.0
RTE3 test	Best: Hickl et al. 07	80.0
	Average (26 teams)	62.4
	Median (26 teams)	62.6

Table 2.2: Results from teams participating in the first three RTE competitions. For each RTE competition, we show the accuracy achieved by the best-performing team, along with average and median accuracies over all participating teams. Each RTE competition allowed teams to submit two runs; for those teams which did so, we used the run with higher accuracy in computing all statistics.

to the average and median accuracies achieved by all RTE participants in each of the RTE competitions. Indeed, when evaluated on the RTE3 test set (after training on the RTE3 development set), the bag-of-words model achieved an accuracy (63%) higher than that of more than half of the RTE3 participants. Many of those teams had pursued approaches which were apparently far more sophisticated than the bag-of-words model, yet our results show that they could have done better with a more naïve approach.

Note that, while the system of Glickman et al. (2005) was essentially a pure bag-of-words model, few or none of the systems submitted to RTE2 or RTE3 could be described as such. Most of those which came closest to measuring pure lexical overlap (e.g., Adams (2006), Malakasiotis and Androutsopoulos (2007)) exploited additional information as well, such as the occurrence of negation in p or h. For comparison, Zanzotto et al. (2006) describe experiments using a simple statistical lexical system, similar to the one described in this chapter. They were able to achieve accuracy close to 60% on both RTE1 and RTE2, and found that simple lexical models outperformed more sophisticated lexical models, thus confirming our findings.

The second surprise in table 2.1 is the degree of variance in the results across data sets. A long-standing criticism of the RTE test suites has been that, because the task is loosely defined and the problems are not drawn from any "natural" data distribution, there is no way to ensure consistency in the nature and difficulty of the NLI problems from one RTE data set to the next. Many in the RTE community have the subjective perception that the character of the problems has changed significantly from one RTE test suite to the next; the results in table 2.1 show that there are substantial objective differences as well. Using our simple bag-of-words model as a yardstick, the RTE1 test suite is the hardest, while the RTE2 test suite is roughly 4% easier, and the RTE3 test suite is roughly 9% easier. Moreover, in RTE3 there is a marked disparity between the difficulty of the development and test sets. Whichever data set we train on, the RTE3 development set appears to be roughly 6% easier than the RTE3 test set—certainly, an undesirable state of affairs.

A final observation about the results in table 2.1 is that the model does not appear to be especially prone to overfitting. Half of the rows in table 2.1 show

results for experiments where the test data is the same as the training data, yet these results are not appreciably better than the results for experiments where the test data is different from the training data. However, this result is not too surprising—because the maximum entropy classifier uses just two features, it has only three free parameters, and thus is unlikely to overfit the training data.

Chapter 3

Alignment for natural language inference

In order to recognize that *Kennedy was killed* can be inferred from *JFK was assassinated*, one must first recognize the correspondence between *Kennedy* and *JFK*, and between *killed* and *assassinated*. Consequently, most current approaches to NLI rely, implicitly or explicitly, on a facility for *alignment*—that is, establishing links between corresponding entities and predicates in the premise p and the hypothesis h.

Recent entries in the annual Recognizing Textual Entailment (RTE) competition (Dagan et al. 2005) have addressed the alignment problem in a variety of ways, though often without distinguishing it as a separate subproblem. Glickman et al. (2005) and Jijkoun and de Rijke (2005), among others, have explored approaches like that presented in chapter 2, based on measuring the degree of lexical overlap between bags of words. While ignoring structure, such methods depend on matching each word in h to the word in p with which it is most similar—in effect, an alignment. At the other extreme, Tatu and Moldovan (2007) and Bar-Haim et al. (2007) have formulated the inference problem as analogous to proof search, using inferential rules which encode (among other things) knowledge of lexical relatedness. In such approaches, the correspondence between the words of p and h is implicit in the steps of the proof.

Increasingly, however, the most successful RTE systems have made the alignment problem explicit. Marsi and Krahmer (2005) and MacCartney et al. (2006)

first advocated pipelined system architectures containing a distinct alignment component, a strategy crucial to the top-performing systems of Hickl et al. (2006) and Hickl and Bensley (2007). However, each of these systems has pursued alignment in idiosyncratic and poorly-documented ways, often using proprietary data, making comparisons and further development difficult.

In this chapter we undertake the first systematic study of alignment for NLI.[1] We propose a new NLI alignment system (the MANLI system) which uses a phrase-based representation of alignment, exploits external resources for knowledge of semantic relatedness, and capitalizes on the recent appearance of new supervised training data for NLI alignment. In addition, we examine the relation between alignment for NLI and alignment for machine translation (MT), and investigate whether existing MT aligners can usefully be applied in the NLI setting.

3.1 NLI alignment vs. MT alignment

The alignment problem is familiar in MT, where recognizing that *she came* is a good translation for *elle est venue* requires establishing a correspondence between *she* and *elle*, and between *came* and *est venue*. The MT community has developed not only an extensive literature on alignment (Brown et al. 1993, Vogel et al. 1996, Marcu and Wong 2002, DeNero et al. 2006), but also standard, proven alignment tools such as GIZA++ (Och and Ney 2003). Can off-the-shelf MT aligners be applied to NLI? There is reason to be doubtful. Alignment for NLI differs from alignment for MT in several important respects, including:

- Most obviously, it is monolingual rather than cross-lingual, opening the door to utilizing abundant (monolingual) sources of information on semantic relatedness, such as WordNet.
- It is intrinsically asymmetric: p is often much longer than h, and commonly contains phrases or clauses which have no counterpart in h.

[1] The material in this chapter is derived in large part from (MacCartney et al. 2008).

- Indeed, one cannot assume even approximate semantic equivalence—usually a given in MT. Because NLI problems include both valid and invalid inferences, the semantic content of h may diverge substantially from p. An NLI aligner must be designed to accommodate frequent unaligned words and phrases.

- Little training data is available. MT alignment models are typically trained in unsupervised fashion, inducing lexical correspondences from massive quantities of sentence-aligned bitexts. While NLI aligners could in principle do the same, large volumes of suitable data are lacking. NLI aligners must therefore depend on smaller quantities of supervised training data, supplemented by external lexical resources. Conversely, while existing MT aligners can make use of dictionaries, they are not designed to harness other sources of information on degrees of semantic relatedness.

Consequently, the tools and techniques of MT alignment may not transfer readily to NLI alignment. We investigate the matter empirically in section 3.4.2.

3.2 The MSR alignment data

Until recently, research on alignment for NLI has been hampered by a paucity of high-quality, publicly available data from which to learn. Happily, that has begun to change, with the release by Microsoft Research (MSR) of gold-standard alignment annotations (Brockett 2007) for inference problems from the second Recognizing Textual Entailment (RTE2) challenge (Bar-Haim et al. 2006). To our knowledge, we are the first to exploit this data for training and evaluation of NLI alignment models.

The RTE2 data consists of a development set and a test set, each containing 800 inference problems. Each problem consists of a premise p and a hypothesis h. The premises contain 29 words on average; the hypotheses, 11 words. Each problem is marked as a valid or invalid inference (50% each); however, these annotations are ignored during alignment, since they would not be available during testing of a complete NLI system.

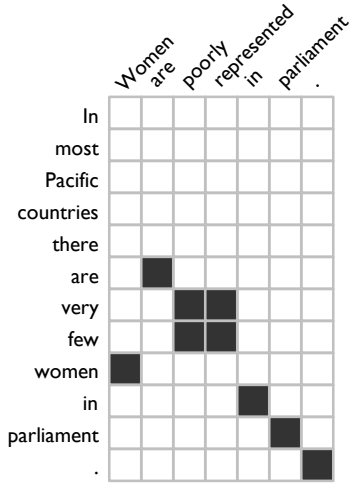

Figure 3.1: The MSR gold standard alignment for problem 116 from the RTE2 development set.

The MSR annotations use an alignment representation which is token-based, but many-to-many, and thus allows implicit alignment of multi-word phrases. Figure 3.1 shows an example in which *very few* has been aligned with *poorly represented*.

In the MSR data, every alignment link is marked as SURE or POSSIBLE. In making this distinction, the annotators have followed a convention common in MT, which permits alignment precision to be measured against both SURE and POSSIBLE links, while recall is measured against only SURE links. In this work, however, we have chosen to ignore POSSIBLE links, embracing the argument made by Fraser and Marcu (2007) that their use has impeded progress in MT alignment models, and that SURE-only annotation is to be preferred.

Each RTE2 problem was independently annotated by three people, following carefully designed annotation guidelines. Inter-annotator agreement was high: Brockett (2007) reports Fleiss' kappa[2] scores of about 0.73 ("substantial agreement") for mappings from h tokens to p tokens; and all three annotators agreed on $\sim$70% of proposed

[2]Fleiss' kappa generalizes Cohen's kappa to the case where there are more than two raters.

links, while at least two of three agreed on more than 99.7% of proposed links,[3] attesting to the high quality of the annotation data. For this work, we merged the three independent annotations, according to majority rule,[4] to obtain a gold-standard annotation containing an average of 7.3 links per RTE problem.

3.3 The MANLI aligner

In this section, we describe the MANLI aligner, a new alignment system designed expressly for NLI alignment. The MANLI system consists of four elements:

- a phrase-based representation of alignment,
- a feature-based linear scoring function for alignments,
- a decoder which uses simulated annealing to find high-scoring alignments, and
- perceptron learning to optimize feature weights.

In the following pages, we'll describe each of these elements in turn.

3.3.1 A phrase-based alignment representation

MANLI uses an alignment representation which is intrinsically phrase-based. (Following the usage common in MT, we use "phrase" to mean any contiguous span of tokens, not necessarily corresponding to a syntactic phrase.) We represent an alignment E between a premise p and a hypothesis h as a set of phrase edits $\{e_1, e_2, \ldots\}$, each belonging to one of four types:

- an EQ edit connects a phrase in p with an equal (by word lemmas) phrase in h
- a SUB edit connects a phrase in p with an unequal phrase in h
- a DEL edit covers an unaligned phrase in p
- an INS edit covers an unaligned phrase in h

DEL(In_1)
DEL($most_2$)
DEL($Pacific_3$)
DEL($countries_4$)
DEL($there_5$)
EQ(are_6, are_2)
SUB($very_7$ few_8, $poorly_3$ $represented_4$)
EQ($women_9$, $Women_1$)
EQ(in_{10}, in_5)
EQ($parliament_{11}$, $parliament_6$)
EQ($._{12}$, $._7$)

Figure 3.2: The MSR gold standard alignment for RTE2 problem 116 (the same problem as in figure 3.1), represented as a set of phrase edits. Note that DEL and INS edits of size > 1 are possible in principle, but are not used in our training data.

As an example, figure 3.2 shows the same alignment as in figure 3.1, but represented as a set of phrase edits.

Alignments are constrained to be one-to-one at the phrase level: every token in p and h belongs to exactly one phrase, which participates in exactly one edit (possibly DEL or INS). However, the phrase representation permits alignments which are many-to-many at the token level. In fact, this is the chief motivation for the phrase-based representation: we can align *very few* and *poorly represented* as units, without being forced to make an arbitrary choice as to which word goes with which word. Moreover, our scoring function can make use of lexical resources which have information about semantic relatedness of multi-word phrases, not merely individual words.

About 23% of the MSR gold-standard alignments are not one-to-one (at the token level), and are therefore technically unreachable for MANLI, which is constrained to generate one-to-one alignments. However, by merging contiguous token links into phrase edits of size > 1, most MSR alignments (about 92%) can be straightforwardly converted into MANLI-reachable alignments. For the purpose of model training (but *not* for the evaluation described in section 3.4.4), we generated a version of the MSR

[3]The SURE/POSSIBLE distinction is taken as significant in computing all these figures.
[4]The handful of three-way disagreements were treated as POSSIBLE links, and thus were not used.

data in which all alignments were converted to MANLI-reachable form.[5]

3.3.2 A feature-based scoring function

To score alignments, we use a simple feature-based linear scoring function, in which the score of an alignment is the sum of the scores of the edits it contains (including not only SUB and EQ edits, but also DEL and INS edits), and the score of an edit is the dot product of a vector encoding its features and a vector of weights. If E is a set of edits constituting an alignment, and $\mathbf{\Phi}$ is a vector of feature functions, the score s is given by:

$$s(E) = \sum_{e \in E} s(e) = \sum_{e \in E} \mathbf{w} \cdot \mathbf{\Phi}(e)$$

We'll explain how the feature weights $\mathbf{w}$ are set in section 3.3.4. The features used to characterize each edit are as follows:

Edit type features. We begin with boolean features encoding the type of each edit. We expect EQs to score higher than SUBs, and (since p is commonly longer than h) DELs to score higher than INSs.

Phrase features. Next, we have features which encode the sizes of the phrases involved in the edit, and whether these phrases are non-constituents (in syntactic parses of the sentences involved).

Semantic relatedness feature. For SUB edits, a very important feature represents the degree of semantic relatedness of the substituends, as a real value in $[0, 1]$. This relatedness score is computed as a max over a number of component scoring functions, some based on external lexical resources, including:

- various string similarity functions, of which most are applied to word lemmas

[5] About 8% of the MSR alignments contain non-contiguous links, most commonly because p contains two references to an entity (e.g., *Christian Democrats* and *CDU*) which are both linked to a reference to the same entity in h (e.g., *Christian Democratic Union*). In such cases, one or more links must be eliminated to achieve a MANLI-reachable alignment. We used a string-similarity heuristic to break such conflicts, but were obliged to make an arbitrary choice in about 2% of cases.

- measures of synonymy, hypernymy, antonymy, and semantic relatedness, including a widely-used measure due to Jiang and Conrath (1997), based on manually constructed lexical resources such as WordNet and NomBank
- a function based on the well-known distributional similarity metric of Lin (1998), which automatically infers similarity of words and phrases from their distributions in a very large corpus of English text

The ability to leverage external lexical resources—both manually constructed and automatically induced—is critical to the success of MANLI.

Contextual features. Even when the lexical similarity for a SUB edit is high, it may not be a good match. If p or h contains multiple occurrences of the same word—which happens frequently with function words, and occasionally with content words—lexical similarity alone may not suffice to determine which is the right match. To remedy this, we introduce contextual features for SUB and EQ edits. A real-valued *distortion* feature measures the difference between the relative positions of the substituends within their respective sentences, while boolean *matching neighbors* features indicate whether the tokens before and after the substituends are equal or similar.

3.3.3 Decoding using simulated annealing

The problem of decoding—that is, finding a high-scoring alignment for a particular inference problem—is made more complex by our choice of a phrase-based alignment representation. For a model which uses a token-based representation (say, one which simply maps h tokens to p tokens), decoding is trivial, since each token can be aligned independently of its neighbors. (This is the case for the bag-of-words model described in chapter 2.) But with a phrase-based representation, things are more complicated. The segmentation into phrases is not given in advance, and every phrase pair considered for alignment must be consistent with its neighbors with respect to segmentation. Consequently, the decoding problem cannot be factored into a number of independent decisions.

Inputs

- an alignment problem $\langle p, h \rangle$
- a number of iterations N (e.g. 100)
- initial temperature T_0 (e.g. 40) and multiplier r (e.g. 0.9)
- a bound on edit size max (e.g. 6)
- an alignment scoring function, SCORE(E)

Initialize

- Let E be an "empty" alignment for $\langle p, h \rangle$
  (i.e., E contains only DEL and INS edits, no EQ or SUB edits)
- Set $\hat{E} = E$

Repeat for $i = 1$ to N

- Let $\{F_1, F_2, ...\}$ be the set of possible successors of E. To generate this set:
  - Consider every possible edit f up to size max
  - Let $C(E, f)$ be the set of edits in E which "conflict" with f (i.e., involve at least some of the same tokens as f)
  - Let $F = E \cup \{f\} \setminus C(E, f)$
- Let $s(F)$ be a map from successors of E to scores generated by SCORE
- Set $P(F) = \exp s(F)$, and then normalize $P(F)$, transforming the score map to a probability distribution
- Set $T_i = r \cdot T_{i-1}$
- Set $P(F) = P(F)^{1/T_i}$, smoothing or sharpening $P(F)$
- Renormalize $P(F)$
- Choose a new value for E by sampling from $P(F)$
- If SCORE(E) > SCORE($\hat{E}$), set $\hat{E} = E$

Return $\hat{E}$

Figure 3.3: The MANLI-ALIGN algorithm

To address this difficulty, we have devised a stochastic alignment algorithm, MANLI-ALIGN (figure 3.3), which uses a simulated annealing strategy. Beginning from an arbitrary alignment, we make a series of local steps, at each iteration sampling from a set of possible successors according to scores assigned by our scoring function. The sampling is controlled by a "temperature" which falls over time. At the beginning of the process, successors are sampled with nearly uniform probability, which helps to ensure that the space of possibilities is explored and local maxima are avoided. As the temperature falls, there is a ever-stronger bias toward high-scoring successors, so that the algorithm converges on a near-optimal alignment. Clever use of memoization helps to ensure that computational costs remain manageable. Using the parameter values suggested in figure 3.3, aligning an average RTE problem takes about two seconds.

Inputs

- training problems $\langle p_j, h_j \rangle$, $j = 1..n$
- corresponding gold-standard alignments E_j
- a number of learning epochs N (e.g. 50)
- a "burn-in" period $N_0 < N$ (e.g. 10)
- initial learning rate R_0 (e.g. 1) and multiplier r (e.g. 0.8)
- a vector of feature functions $\mathbf{\Phi}(E)$
- an alignment algorithm $\text{ALIGN}(p, h; \mathbf{w})$ which finds a good alignment for $\langle p, h \rangle$ using weight vector $\mathbf{w}$

Initialize

- Set $\mathbf{w} = 0$

Repeat for $i = 1$ to N

- Set $R_i = r \cdot R_{i-1}$, reducing the learning rate
- Randomly shuffle the training problems
- For $j = 1$ to n:
  - Set $\hat{E}_j = \text{ALIGN}(p_j, h_j; \mathbf{w})$
  - Set $\mathbf{w} = \mathbf{w} + R_i \cdot (\mathbf{\Phi}(E_j) - \mathbf{\Phi}(\hat{E}_j))$
- Set $\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|_2$ (L2 normalization)
- Set $\mathbf{w}[i] = \mathbf{w}$, storing the weight vector for this epoch

Return an averaged weight vector:

- $\mathbf{w}_{avg} = 1/(N - N_0) \sum_{i=N_0+1}^{N} \mathbf{w}[i]$

Figure 3.4: The MANLI-LEARN algorithm

While MANLI-ALIGN is not guaranteed to produce optimal alignments, there is reason to believe that it usually comes very close. After training, the alignment found by MANLI scored at least as high as the gold alignment for 99.6% of RTE problems.[6]

3.3.4 Perceptron learning

To tune the parameters $\mathbf{w}$ of the model, we use an adaptation of the averaged perceptron algorithm (Collins 2002), which has proven successful on a range of NLP tasks. The algorithm is shown in figure 3.4. After initializing $\mathbf{w}$ to 0, we perform N training epochs. (Our experiments used $N = 50$.) In each epoch, we iterate through the training data, updating the weight vector at each training example according to

[6]This figure is based on the MANLI-reachable version of the gold-standard data described in section 3.3.1. For the raw gold-standard data, the figure is 88.1%. The difference is almost entirely attributable to unreachable gold alignments, which tend to score higher simply because they contain more edits (and because the learned weights are mostly positive).

the difference between the features of the target alignment[7] and the features of the alignment produced by the decoder using the current weight vector. The size of the update is controlled by a learning rate which decreases over time. At the end of each epoch, the weight vector is normalized and stored. The final result is the average of the stored weight vectors, omitting vectors from a fixed number of epochs at the beginning of the run (which tend to be of poor quality). Using the parameter values suggested in figure 3.4, training runs on the RTE2 development set required about 20 hours.

3.4 Evaluating aligners on MSR data

In this section, we describe experiments designed to evaluate the performance of various alignment systems on the MSR gold-standard data described in section 3.2. For each system, we report precision, recall, and F-measure (F_1). These statistics are assessed per problem by counting aligned token pairs.[8] In a particular alignment problem, if A is the set of token pairs which are aligned in the guessed alignment, and S is the set of token pairs which are aligned in the gold alignment, then precision $P = |A \cap S|/|A|$, while recall $R = |A \cap S|/|S|$. F-measure is defined as usual by $F_1 = 2PR/(P+R)$.[9] P, R, and F_1 are then averaged over all problems in a problem set.[10] We also report the exact match rate: that is, the proportion of problems in which the guessed alignment exactly matches the gold alignment. The results are summarized in table 3.1.

[7]For training (but not for testing), we produced a version of the gold-standard data in which contiguous many-to-many token links (as in figure 3.1) were merged into SUB edits of size > 1 (and a few non-contiguous links were eliminated).

[8]For phrase-based alignments like those generated by MANLI, two tokens are considered to be aligned iff they are contained within phrases which are aligned.

[9]MT researchers conventionally report results in terms of alignment error rate (AER). Since we use only SURE links in the gold standard data (see section 3.2), AER is equivalent to $1 - F_1$.

[10]Note that this is a form of *macro-averaging*. MT evaluations conventionally use micro-averaging, which gives greater weight to problems containing more aligned pairs. This makes sense in MT, where the purpose of alignment is to induce phrase tables. But in NLI, where the ultimate goal is to maximize the number of inference problems answered correctly, it is more fitting to give all problems equal weight, and so we macro-average. We have also generated all results using micro-averaging, and found that the relative comparisons are not greatly affected.

3.4.1 A robust baseline: the bag-of-words aligner

As a baseline, we use a simple alignment algorithm which is essential the same as the bag-of-words model presented in chapter 2. Each hypothesis token h_j is aligned to the premise token p_i to which it is most similar, according to a lexical similarity function based on string edit distance between word lemmas. To generate alignment scores (which will be of interest in section 3.5), we simply normalize the *cost* function defined in section 2.1.4 by the length of h:

$$score(h|p) = \frac{1}{|h|} cost(h|p)$$

Despite the simplicity of this alignment model, its performance is fairly robust, with good recall. Its precision, however, its mediocre—chiefly because, by design, it aligns every hypothesis token with some premise token. The model could surely be improved by allowing it to leave some h tokens unaligned, but this was not pursued.

3.4.2 MT aligners: GIZA++ and Cross-EM

Given the importance of alignment for NLI, and the availability of standard, proven tools for MT alignment, an obvious question presents itself: why not use an off-the-shelf MT aligner for NLI? Although we have argued (section 3.1) that this is unlikely to succeed, to our knowledge, we are the first to investigate the matter empirically.[11]

The best-known and most-used MT aligner is undoubtedly GIZA++ (Och and Ney 2003), which contains implementations of various IBM models (Brown et al. 1993), as well as the HMM model of Vogel et al. (1996). Most practitioners use GIZA++ as a black box, via the Moses MT toolkit (Koehn et al. 2007). We followed this practice, running with Moses' default parameters on the RTE2 data to obtain asymmetric word alignments in both directions (p-to-h and h-to-p). We then performed symmetrization using the well-known INTERSECTION heuristic.

Unsurprisingly, the out-of-the-box performance was quite poor, with most words aligned apparently at random. Precision was fair (72%) but recall was very poor

[11] However, Dolan et al. (2004) explore a closely-related topic: using an MT aligner to identify paraphrases.

System	Data	P %	R %	F_1 %	E %
Bag-of-words	dev	57.8	81.2	67.5	3.5
(baseline)	test	62.1	82.6	70.9	5.3
GIZA++	dev	83.0	66.4	72.1	—
(using lexicon, $\cap$ heuristic)	test	85.1	69.1	74.8	—
Cross-EM	dev	67.6	80.1	72.1	—
(using lexicon, $\cap$ heuristic)	test	70.3	81.0	74.1	—
Stanford RTE	dev	81.1	61.2	69.7	0.5
	test	82.7	61.2	70.3	0.3
Stanford RTE	dev	81.1	75.8	78.4	—
(punctuation correction)	test	82.7	75.8	79.1	—
MANLI	dev	83.4	85.5	84.4	21.7
	test	85.4	85.3	85.3	21.3

Table 3.1: Performance of various aligners on the MSR RTE2 alignment data. The columns show the data set used (800 problems each); average precision, recall, and F-measure; and the exact match rate (see text).

(46%). Even equal words were usually not aligned—because GIZA++ is designed for cross-linguistic use, it does not consider word equality between source and target sentences. To remedy this, we supplied GIZA++ with a lexicon, using a trick common in the MT community: we supplemented the training data with synthetic data consisting of matched pairs of equal words. This gives GIZA++ a better chance of learning that, e.g., *man* should align with *man*. The result was a big boost in recall (+23%), and a smaller gain in precision. The results for GIZA++ shown in table 3.1 are based on using the lexicon and INTERSECTION. With these settings, GIZA++ properly aligned most pairs of equal words, but continued to align other words apparently at random.

Next, we compared the performance of INTERSECTION with other symmetrization heuristics defined in Moses—including UNION, GROW, GROW-DIAG, GROW-DIAG-FINAL (the default), and GROW-DIAG-FINAL-AND—and with asymmetric alignments

in both directions. While all these alternatives achieved better recall than INTERSECTION, all showed substantially worse precision and F_1. On the RTE2 test set, the asymmetric alignment from h to p scored 68% in F_1; GROW scored 58%; and all other alternatives scored below 52%.

As an additional experiment, we tested the Cross-EM aligner (Liang et al. 2006) from the BerkeleyAligner package on the MSR data. While this aligner is in many ways simpler than GIZA++ (it lacks any model of fertility, for example), its method of jointly training two simple asymmetric HMM models has outperformed GIZA++ on standard evaluations of MT alignment. As with GIZA++, we experimented with a variety of symmetrization heuristics, and ran trials with and without a supplemental lexicon. The results were broadly similar: INTERSECTION greatly outperformed alternative heuristics, and using a lexicon provided a big boost (up to 12% in F_1). Under optimal settings, the Cross-EM aligner showed better recall and worse precision than GIZA++, with F_1 just slightly lower. Like GIZA++, it did well at aligning equal words, but aligned most other words at random.

The mediocre performance of MT aligners on NLI alignment comes as no surprise, for reasons discussed in section 3.1. Above all, the quantity of training data is simply too small for unsupervised learning to succeed. A successful NLI aligner will need to exploit supervised training data, and will need access to additional sources of knowledge about lexical relatedness.

3.4.3 The Stanford RTE aligner

A better comparison is thus to an alignment system expressly designed for NLI. For this purpose, we used the alignment component of the Stanford RTE system, which will be described in chapter 4.[12] The Stanford aligner performs decoding and learning in a similar fashion to MANLI, but uses a simpler, token-based alignment representation, along with a richer set of features for alignment scoring. It represents alignments as an injective map from h tokens to p tokens. Phrase alignments are not directly representable, although the effect can be approximated by a pre-processing step which

[12]However, the experiments described here used a more recent version of the Stanford RTE system (best described in (Chambers et al. 2007)) than that which forms the basis of chapter 4.

collapses multi-token named entities and certain collocations into single tokens. The features used for alignment scoring include not only measures of lexical similarity, but also syntactic features encoding the similarity of paths through dependency parses between pairs of aligned tokens, which are intended to promote the alignment of similar predicate-argument structures.

Despite this sophistication, the out-of-the-box performance of the Stanford aligner is mediocre, as shown in table 3.1. The low recall figures are particularly noteworthy. However, a partial explanation is readily available: by design, the Stanford system ignores punctuation.[13] Because punctuation tokens constitute about 15% of the aligned pairs in the MSR data, this sharply reduces measured recall. However, since punctuation matters little in inference, such recall errors arguably should be forgiven. Thus, table 3.1 also shows adjusted statistics for the Stanford system in which all punctuation alignments are (generously) counted as correct.

Even after this adjustment, the recall figures are unimpressive. Error analysis reveals that the Stanford aligner does a poor job of aligning function words. About 13% of the aligned pairs in the MSR data are matching prepositions or articles; the Stanford aligner misses about 67% of such pairs. (By contrast, MANLI misses only 10% of such pairs.) While function words matter less in inference than nouns and verbs, they are not irrelevant, and because sentences often contain multiple instances of a particular function word, matching them properly is by no means trivial. If matching prepositions and articles were ignored (in addition to punctuation), the gap in F_1 between the MANLI and Stanford systems would narrow to about 2.8%.

Finally, the Stanford aligner is handicapped by its token-based alignment representation, often failing (partly or completely) to align multi-word phrases such as *peace activists* with *protesters*, or *hackers* with *non-authorized personnel*.

3.4.4 The MANLI aligner

As table 3.1 indicates, the MANLI aligner was found to outperform all other aligners evaluated on every measure of performance, achieving an F_1 score 10.5% higher than

[13] In fact, it operates on a dependency-graph representation from which punctuation is omitted.

GIZA++ and 6.2% higher than the Stanford aligner (even with the punctuation correction).[14] MANLI achieved a good balance between precision and recall, and matched more than 20% of the gold-standard alignments exactly.

Three factors seem to have contributed most to MANLI's success. First, MANLI is able to outperform the MT aligners principally because it is able to leverage lexical resources to identify the similarity between pairs of words such as *jail* and *prison*, *prevent* and *stop*, or *injured* and *wounded*. Second, MANLI's contextual features enable it to do better than the Stanford aligner at matching function words, a weakness of the Stanford aligner discussed in section 3.4.3. Third, MANLI gains a marginal advantage because its phrase-based representation of alignment permits it to properly align phrase pairs such as *death penalty* and *capital punishment*.

However, the phrase-based representation contributed far less than we had hoped. Setting MANLI's maximum phrase size to 1 (effectively, restricting it to token-based alignments) caused F_1 to fall by just 0.2%. We do not interpret this to mean that phrase alignments are not useful—indeed, about 2.6% of the links in the gold-standard data involve phrases of size > 1. Rather, we think it shows that we have failed to fully exploit the advantages of the phrase-based representation, chiefly because we lack lexical resources providing good information on similarity of multi-word phrases. Even if we had such resources, however, the consequent gains might be modest. Recent work by de Marneffe et al. (2009) suggests that the role played by multi-word expressions in RTE is smaller than one might expect, and is largely restricted to lexico-syntactic variations which can be captured by other means.

Error analysis suggests that there is ample room for improvement. A large proportion of recall errors (perhaps 40%) occur because the lexical similarity function assigns too low a value to pairs of words or phrases which are clearly similar, such as *conservation* and *protecting*, *server* and *computer networks*, *organization* and *agencies*, or *bone fragility* and *osteoporosis*. Better exploitation of lexical resources could help to reduce such errors. Another important category of recall errors (about 12%) result from the failure to identify one- and multi-word versions of the name of some entity, such as *Lennon* and *John Lennon*, or *Nike Inc.* and *Nike*. A special-purpose

[14] Reported results for MANLI are averages over 10 runs.

similarity function could help here. Note, however, that about 10% of the recall errors are unavoidable, given our choice of alignment representation, since they involve cases where the gold standard aligns one or more tokens on one side to a non-contiguous set of tokens on the other side.

Precision errors may be harder to reduce. These errors are dominated by cases where we mistakenly align two equal function words (49% of precision errors), two forms of the verb *to be* (21%), two equal punctuation marks (7%), or two words or phrases of other types having equal lemmas (18%). Because such errors often occur because the aligner is forced to choose between nearly equivalent alternatives, they may be difficult to eliminate. The remaining 5% of precision errors result mostly from aligning words or phrases rightly judged to be highly similar, such as *expanding* and *increasing*, *labor* and *birth*, *figures* and *number*, or *223,000* and *220,000*.

3.5 Using alignment to predict RTE answers

In section 3.4, we evaluated the ability of aligners to recover gold-standard alignments. But since alignment is just one component of the NLI problem, we might also examine the impact of different aligners on the ability to recognize valid inferences. If a high-scoring alignment indicates a close correspondence between h and p, does this also indicate a valid inference? As we will argue in chapter 4, there is more to inferential validity than close lexical or structural correspondence: negations, antonymy, modals, non-factive and implicative verbs, and other linguistic constructs can affect validity in ways hard to capture in alignment. Nevertheless, alignment score can be a strong predictor of inferential validity, and some NLI systems (e.g., that of Glickman et al. (2005)) rely entirely on some measure of alignment quality to predict validity.

If an aligner generates real-valued alignment scores, we can use the RTE data to test its ability to predict inferential validity with the following simple method. For a given RTE problem, we predict YES (valid) if its alignment score[15] exceeds a threshold

[15]For good results, it may be necessary to normalize the alignment score. Scores from MANLI were normalized by the number of tokens in the problem. The Stanford aligner performs a similar normalization internally.

System	data	acc %	avgP %
Bag-of-words aligner	dev	61.3	61.5
	test	57.9	58.9
Stanford RTE aligner	dev	63.1	64.9
	test	60.9	59.2
MANLI aligner	dev	59.3	69.0
(this work)	test	60.3	61.0
LCC (Hickl et al. 2006)	test	75.4	80.8
RTE2 entries (average)	test	58.5	59.1

Table 3.2: Performance of various aligners and complete RTE systems in predicting RTE2 answers. The columns show the data set used, accuracy, and average precision (the recommended metric for RTE2).

τ, and NO otherwise. We tune τ to maximize accuracy on the RTE2 development set, and then measure accuracy on the RTE2 test set using the same τ.

Table 3.2 shows results for several NLI aligners, along with some results for complete RTE systems, including the LCC system (the top performer at RTE2) and an average of all systems participating in RTE2. While none of the aligners rivals the performance of the LCC system, all achieve respectable results, and the Stanford and MANLI aligners outperform the average RTE2 entry. Thus, even if alignment quality does not determine inferential validity, many NLI systems could be improved by harnessing a well-designed NLI aligner.

3.6 Conclusion

While MT aligners succeed by unsupervised learning of word correspondences from massive amounts of bitext, NLI aligners are forced to rely on smaller quantities of supervised training data. With the MANLI system, we have demonstrated how to overcome this lack of data by utilizing external lexical resources, and how to gain additional power from a phrase-based representation of alignment.

Chapter 4

The Stanford RTE system

In this chapter, we describe the Stanford RTE system[1] for natural language inference, which represents an initial attempt to find a middle ground between, on the one hand, approaches to NLI based on lexical similarity, which are robust, but imprecise; and on the other, approaches based on full semantic interpretation, which are precise, but brittle. Because full, accurate, open-domain natural language understanding lies far beyond current capabilities, most early efforts in natural language inference sought to extract the maximum mileage from quite limited semantic representations. Some (like the bag-of-words model described in chapter 2) used simple measures of semantic overlap, but over time, the more interesting work largely converged on a graph-alignment approach, operating on semantic graphs derived from syntactic dependency parses, and using a locally-decomposable alignment score as a proxy for strength of entailment. (Below, we argue that even approaches relying on weighted abduction may be seen in this light.) In this chapter, we highlight the fundamental semantic limitations of this type of approach, and advocate a multi-stage architecture that addresses these limitations. The three key limitations are an *assumption of monotonicity*, an *assumption of locality*, and a *confounding of alignment*

[1]The Stanford RTE system has been developed over several years by a large team, including (in addition to this author) Rajat Raina, Trond Grenager, Marie-Catherine de Marneffe, Anna Rafferty, Christopher D. Manning, and many others. The material in this chapter is derived in large part from (MacCartney et al. 2006), and therefore reflects an earlier stage of development of the Stanford RTE system. The more recent evolution of the system is described in (Chambers et al. 2007, Padó et al. 2008).

and evaluation of entailment.

The system described in this chapter was developed around, and evaluated on, natural language inference problems from the PASCAL RTE1 data (Dagan et al. 2005), which includes a development set of 567 problems and a test set of 800 problems. Some example problems are shown in figure 4.1.

4.1 Approaching a robust semantics

In this section we try to give a unifying overview to recent work on robust natural language inference, to present fundamental limitations of prominent methods, and then to outline our approach to resolving them. Until 2006, all robust natural language inference systems of which we are aware employed a single-stage matching/proof process, differing mainly in the sophistication of the matching stage. The simplest approach is to base the entailment prediction on the degree of semantic overlap between the premise p and hypothesis h using models based on bags-of-words, bags-of-n-grams, TF-IDF scores, or something similar (Jijkoun and de Rijke 2005). (The model presented in chapter 2 typifies this approach.) Such models are too impoverished to be of much use, however. Semantic overlap is typically a symmetric relation, whereas entailment is clearly not. Moreover, because overlap models do not account for syntactic or semantic structure, they are easily fooled by examples like problem 2081 in figure 4.1.

A more sophisticated approach is to formulate the entailment prediction as a graph matching problem (Haghighi et al. 2005, de Salvo Braz et al. 2005). In this formulation, the input sentences p and h are represented as normalized syntactic dependency graphs (like the one shown in figure 4.2) and entailment is approximated with an alignment between the h graph and a portion of the corresponding p graph. Each possible alignment of the graphs has an associated score, and the score of the best alignment is used as an approximation to the strength of the entailment: a better-aligned hypothesis is assumed to be more likely to be entailed. To enable incremental search, alignment scores are usually factored as a combination of local terms, corresponding to the nodes and edges of the two graphs. Unfortunately, even

59	*p*	Two Turkish engineers and an Afghan translator kidnapped in December were freed Friday.
	h	translator kidnapped in Iraq NO
98	*p*	Sharon warns Arafat could be targeted for assassination.
	h	prime minister targeted for assassination NO
152	*p*	Twenty-five of the dead were members of the law enforcement agencies and the rest of the 67 were civilians.
	h	25 of the dead were civilians. NO
231	*p*	The memorandum noted the United Nations estimated that 2.5 million to 3.5 million people died of AIDS last year.
	h	Over 2 million people died of AIDS last year. YES
971	*p*	Mitsubishi Motors Corp.'s new vehicle sales in the US fell 46 percent in June.
	h	Mitsubishi sales rose 46 percent. NO
1806	*p*	Vanunu, 49, was abducted by Israeli agents and convicted of treason in 1986 after discussing his work as a mid-level Dimona technician with Britain's Sunday Times newspaper.
	h	Vanunu's disclosures in 1968 led experts to conclude that Israel has a stockpile of nuclear warheads. NO
2081	*p*	The main race track in Qatar is located in Shahaniya, on the Dukhan Road.
	h	Qatar is located in Shahaniya. NO

Figure 4.1: Illustrative examples from the RTE1 development set. For each problem, we show the ID number, the premise p and hypothesis h, and the correct answer. Though most of the problems shown have answer NO, the RTE data sets are actually balanced between YES and NO.

with factored scores, the problem of finding the best alignment of two graphs is NP-complete, so exact computation is intractable. Authors have proposed a variety of approximate search techniques. Haghighi et al. (2005) divide the search into two steps: in the first step they consider node scores only, which relaxes the problem to a weighted bipartite graph matching that can be solved in polynomial time, and in the second step they add the edge scores and hillclimb the alignment via an approximate local search.

A third approach, exemplified by Moldovan et al. (2003) and Raina et al. (2005), is to turn the syntactic representation into a neo-Davidsonian-style quasi-logical form, and to perform weighted abductive theorem proving in the tradition of Hobbs et al. (1988). We argue, however, that this style of weighted resolution theorem proving is actually isomorphic to the graph matching approach. For example, a neo-Davidsonian representation of the graph in figure 4.2 might correspond to the quasi-LF

$$rose(e_1) \wedge nsubj(e_1, x_1) \wedge sales(x_1) \wedge nn(x_1, x_2) \wedge Mitsubishi(x_2) \wedge$$
$$dobj(e_1, x_3) \wedge percent(x_3) \wedge num(x_3, x_4) \wedge 46(x_4)$$

There is a term corresponding to each node and arc, and the unit resolution steps at the core of resolution theorem proving consider matching an individual node or arc of the hypothesis with something from the premise, just as in the graph-matching approach.[2]

Finally, a few efforts (Akhmatova 2005, Fowler et al. 2005, Bos and Markert 2005a) have tried to translate sentences into formulas of first-order logic, in order to test logical entailment with a theorem prover. While in principle this approach does not suffer from the limitations we describe below, in practice it has not borne much fruit. Because few problem sentences can be accurately translated to logical form, and because logical entailment is a strict standard, recall tends to be poor.

The simple graph matching formulation of the problem belies three important issues. First, the above systems assume monotonicity: if a good match is found with a part of the premise, other material in the premise is assumed not to affect the validity

[2]A possible difference is that with a good supply of additional linguistic and world knowledge axioms—as is present in Moldovan et al. (2003) but not Raina et al. (2005)—the theorem prover may generate intermediate forms in the proof, but, nevertheless, individual terms are resolved locally without reference to global context.

of the match. But many entailment decisions are non-monotonic in the node and edge scores. Consider variants on problem 98 in figure 4.1. Suppose the hypothesis were *Arafat targeted for assassination*. This would allow a perfect graph match or zero-cost weighted abductive proof, because the hypothesis is a subgraph of the premise. However, this would be incorrect, because it ignores the modal operator *could*. Information that changes the validity of a proof can also exist outside a matching clause. Consider the alternate premise *Sharon denies Arafat is targeted for assassination*.[3]

The second issue is the assumption of locality. Locality is needed to allow practical search, but many entailment decisions rely on global features of the alignment, and thus do not naturally factor by nodes and edges. To take just one example, dropping a restrictive modifier preserves entailment in a positive context, but not in a negative one. For example, *Dogs barked loudly* entails *Dogs barked*, but *No dogs barked loudly* does not entail *No dogs barked*. These more global phenomena cannot be modeled with a factored alignment score.

The last issue arising in the graph matching approaches is the inherent difficulty in confounding alignment and entailment determination. The way to show that one graph element does not follow from another is to make the cost of aligning them high. However, since we are embedded in a search for the lowest cost alignment, this will just cause the system to choose an alternate alignment rather than recognizing a non-entailment. In problem 152 of figure 4.1, we would like the hypothesis to align with the first part of the premise, to be able to prove that civilians are not members of law enforcement agencies and conclude that the hypothesis does not follow from the premise. But a graph-matching system will to try to get non-entailment by making the matching cost between *civilians* and *members of law enforcement agencies* be very high. However, the likely result of that is that the final part of the hypothesis will align with *were civilians* at the end of the premise, assuming that we allow the alignment to "break" arcs.[4] Under this candidate alignment, the lexical alignments are perfect, and the only imperfect alignment is the subject arc of *were* is mismatched in

[3]This is the same problem labeled and addressed as *context* in (Tatu and Moldovan 2005).

[4]Robust systems need to allow matches with imperfect arc correspondence. For instance, given *Bill went to Lyons to study French farming practices*, we would like to be able to conclude that *Bill studied French farming* despite the small structural mismatch.

the two. A robust inference guesser will still likely conclude that there is entailment.

We propose that all three problems can be resolved in a multi-stage architecture, where the alignment phase is followed by a separate phase of entailment determination. Finding aligned content is naturally useful, and can be done by any search procedure. Compared to previous work, our approach to alignment emphasizes structural correspondence, and downplays issues like polarity and quantity, which can be left to a subsequent entailment decision. For example, our alignment scoring function is designed to encourage antonym matches, and ignore the negation of verb predicates. The ideas clearly generalize to evaluating several alignments, but we have found working with the one-best alignment to be adequate for the PASCAL RTE data. Given a good alignment, the determination of entailment reduces to a simple classification decision. The classifier can use hand-tuned weights, or it can be trained to minimize a relevant loss function using standard techniques from machine learning. The classifier is built over features designed to identify patterns of valid and invalid inference. For example, they can detect that it is okay to add a restrictive modifier if the passage has a universal quantifier (*All the students got on the bus* $\models$ *All the male students got on the bus*), whereas this form of inference is not supported in ordinary (upward-monotone) contexts. Because we already have a complete alignment, the classifier's decision can be conditioned on arbitrary *global* features of the aligned graphs, and it can detect inversions of monotonicity.

4.2 System

Our system has three stages: linguistic analysis, alignment, and entailment determination. In the following pages, we describe each of these stages in turn.

4.2.1 Linguistic analysis

In the first stage of processing, our goal is to compute linguistic representations of the premise and hypothesis that contain as much information as possible about their semantic content. We use *typed dependency graphs*, which contain a node for each word,

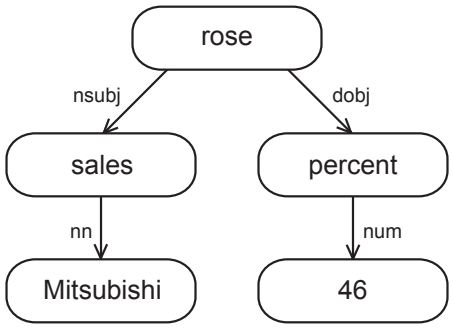

Figure 4.2: A typed dependency graph for problem 971 of figure 4.1.

and labeled edges representing the grammatical relations between words. An example of a typed dependency graph for problem 971 of figure 4.1 is given in figure 4.2. The advantage of this representation is that it contains much of the information about the entities and relations between them described by the sentence, while being easy to compute deterministically from a syntactic parse. The disadvantage is that it fails properly to represent many semantic phenomena; particularly egregious is its inability to represent quantification and modality.

Our general approach is first to parse the input sentences, and then to convert each of the resulting phrase structure trees to a typed dependency graph. We use the Stanford parser (Klein and Manning 2003), a statistical syntactic parser trained on the Penn TreeBank. To ensure correct parsing, we preprocess the sentences to collapse named entities and collocations into single tokens (by joining separate words with underscores). Named entities are identified by the Stanford Named Entity Recognizer (Finkel et al. 2005), a CRF-based NER system similar to that described by McCallum and Li (2003), and collocations are derived from consecutive words pairs in WordNet (Fellbaum et al. 1998).

We convert the phrase structure trees to typed dependency graphs using a set of deterministic hand-coded rules, as described by de Marneffe et al. (2006). Heads of the constituents are first identified using the Collins rules (Collins 2003), modified to retrieve semantic heads. For each grammatical relation, we define patterns over the phrase structure tree using a language similar to the `tgrep` utility. Each pattern is

rose	→	*fell*
sales	→	*sales*
Mitsubishi	→	*Mitsubishi_Motors_Corp.*
percent	→	*percent*
46	→	*46*

SCORE: −0.8962

Figure 4.3: A sample alignment for problem 971 of figure 4.1.

matched against each dependency in the tree, adding the most specific relation found as the typed dependency. The nodes in the final graph are then annotated with their associated word, part-of-speech (given by the parser), lemma (given by a finite-state transducer described by Minnen et al. (2001)) and named-entity tag (given by the NER tagger).

4.2.2 Alignment

The purpose of the second phase of processing is to find a good partial alignment between the typed dependency graphs representing the premise p and the hypothesis h. An alignment consists of a mapping from each node (word) in the h graph to a single node in the p graph, or to null.[5] Not all words in the premise graph are mapped to, since typically the premise contains much more information than needed to support the hypothesis. Furthermore, we do not require that the alignment contains a mapping for all words in the hypothesis. Figure 4.3 shows an example of an alignment for problem 971 of figure 4.1.

The space of alignments is large: there are $O((m+1)^n)$ possible alignments for a p graph with m nodes and an h graph with n nodes. We define a measure of alignment quality, and a procedure for identifying high scoring alignments. We choose a locally decomposable scoring function, such that the score of an alignment is the sum of the local node and edge alignment scores. Unfortunately, there is no polynomial time

[5]The limitations of using one-to-one alignments are mitigated by the fact that many multiword expressions (e.g., named entities, noun compounds, multiword prepositions) have been collapsed into single nodes during linguistic analysis.

algorithm for finding the exact best alignment. Instead we use an incremental beam search, combined with a node ordering heuristic, to do approximate global search in the space of possible alignments. We have experimented with several alternative search techniques, and found that the solution quality is not very sensitive to the specific search procedure used.

Our scoring measure is designed to favor alignments which align semantically similar subgraphs, irrespective of polarity. For this reason, nodes receive high alignment scores when the words they represent are semantically similar. Synonyms and antonyms receive the highest score, and unrelated words receive the lowest. Our hand-crafted scoring metric takes into account the word, the lemma, and the part of speech, and searches for word relatedness using a range of external resources, including WordNet, precomputed latent semantic analysis matrices, and special-purpose gazettes. Alignment scores also incorporate local edge scores, which are based on the shape of the paths between nodes in the premise graph which correspond to adjacent nodes in the hypothesis graph. Preserved edges receive the highest score, and longer paths receive lower scores.

4.2.3 Entailment determination

In the final stage of processing, we make a decision about whether or not h is entailed by p, conditioned on the typed dependency graphs of p and h, as well as the best alignment between them. Because we have a data set of examples that are labeled for entailment, we can use techniques from supervised machine learning to learn a classifier. We adopt the standard approach of defining a feature representation of the problem and then learning a linear decision boundary in the feature space. We focus here on the learning methodology; the next section covers the definition of the set of features.

Defined in this way, one can apply any statistical learning algorithm to this classification task, such as support vector machines, logistic regression, or naive Bayes. We used a logistic regression (or *maximum entropy*) classifier with a Gaussian prior parameter for regularization. We also compare our learning results with those achieved

by hand-setting the weight parameters for the classifier, effectively incorporating strong prior (human) knowledge into the choice of weights.

An advantage to the use of statistical classifiers is that they can be configured to output a probability distribution over possible answers rather than just the most likely answer. This allows us to get reliable confidence estimates for computing a confidence weighted score (see section 4.4). A major concern in applying machine learning techniques to this classification problem is the relatively small size of the training set, which can lead to overfitting problems. We address this by keeping the feature dimensionality small, and using high regularization penalties in training.

4.3 Feature representation

In the entailment determination phase, the entailment problem is reduced to a representation as a vector of 28 features, over which the statistical classifier described above operates. The alignment score is included among the features, but in order to outperform an alignment-only model, the remaining features must somehow capture whatever factors distinguish good (or bad) entailments from merely good (or bad) alignments. The features represent global characteristics of the p and h graphs and the alignment between them. The feature functions are complex, encoding substantial human knowledge. They are designed to capture salient patterns of entailment and non-entailment, with particular attention to contexts which reverse or block monotonicity, such as negations and quantifiers. This section describes several of the most important groups of features.

Polarity features. These features capture the presence (or absence) of linguistic markers of negative polarity contexts in both p and h, such as simple negation (*not*), downward-monotone quantifiers (*no*, *few*), restricting prepositions (*without*, *except*) and superlatives (*tallest*).

Adjunct features. These indicate the dropping or adding of syntactic adjuncts when moving from p to h.[6] For the common case of restrictive adjuncts, dropping an adjunct preserves truth (*Dogs barked loudly* $\models$ *Dogs barked*), while adding an adjunct does not (*Dogs barked* $\not\models$ *Dogs barked today*). However, in negative-polarity contexts (such as *No dogs barked*), this heuristic is reversed: adjuncts can safely be added, but not dropped. For example, in problem 59 of figure 4.1, h aligns well with p, but the addition of *in Iraq* indicates non-entailment.

We identify the "root nodes" of the problem: the root node of the h graph and the corresponding aligned node in the p graph. Using dependency information, we identify whether adjuncts have been added or dropped. We then determine the *polarity* (negative context, positive context or restrictor of a universal quantifier) of the two root nodes to generate features accordingly.

Antonymy features. Entailment problems might involve antonymy, as in problem 971 of figure 4.1. We check whether any aligned pairs of $\langle p, h \rangle$ words appear to be antonymous by consulting a pre-computed list of about 40,000 antonymous and other contrasting pairs derived from WordNet. For each antonymous pair, we generate one of three boolean features, indicating whether (i) the words appear in contexts of matching polarity, (ii) only the p word appears in a negative-polarity context, or (iii) only the h word does.

The antonymy features highlight the difference between the Stanford RTE system and alignment-only systems. An alignment-only model must put a high cost on aligning antonyms; otherwise it may predict that *Sales fell* entails *Sales rose*. But if antonyms are costly to align, it is not possible to correctly predict that *Sales fell, while profits rose* entails *Sales did not rise*. In contrast, our system is happy to align antonyms at low cost, and relies on the antonymy features to distinguish the cases where the antonymy aids or hinders the entailment.

[6] We employ the conventional syntactic distinction between the *arguments* of a verb (such as subject and object), which are presumed to be semantically essential, and the *adjuncts* (such as temporal modifiers), which are not.

Modality features. Modality features capture simple patterns of modal reasoning, as in problem 98 of figure 4.1, which illustrates the heuristic that possibility does not entail actuality. According to the occurrence (or not) of predefined modality markers, such as *must* or *maybe*, we map each of p and h to one of six modalities: POSSIBLE, NOT POSSIBLE, ACTUAL, NOT ACTUAL, NECESSARY, and NOT NECESSARY. The $\langle p, h\rangle$ modality pair is then mapped into one of the following entailment judgments: YES, WEAK YES, DON'T KNOW, WEAK NO, or NO. For example:

$$\begin{aligned} (\text{NOT POSSIBLE} \models \text{NOT ACTUAL})? &\Rightarrow \text{YES} \\ (\text{POSSIBLE} \models \text{NECESSARY})? &\Rightarrow \text{WEAK NO} \end{aligned}$$

Factivity features. The context in which a verb phrase is embedded may carry semantic presuppositions giving rise to (non-)entailments such as *The gangster tried to escape* $\not\models$ *The gangster escaped*. This pattern of entailment, like others, can be reversed by negative polarity markers (*The gangster managed to escape* $\models$ *The gangster escaped* while *The gangster didn't manage to escape* $\not\models$ *The gangster escaped*). To capture these phenomena, we compiled small lists of "factive" and non-factive verbs, clustered according to the kinds of entailments they create. We then determine to which factivity class the parent of the p node aligned with the root of h belongs. If the parent is not in the list, we only check whether the embedding context is an affirmative context or a negative one.

Quantifier features. These features are designed to capture entailment relations among simple sentences involving quantification, such as *Every company must report* $\models$ *A company must report* (or *The company*, or *IBM*). No attempt is made to handle multiple quantifiers or scope ambiguities. Each quantifier found in an aligned pair of $\langle p, h\rangle$ words is mapped into one of five quantifier categories: *no*, *some*, *many*, *most*, and *all*. The *no* category is set apart, while an ordering over the other four categories is defined. The *some* category also includes definite and indefinite determiners and small cardinal numbers. A crude attempt is made to handle negation by interchanging *no* and *all* in the presence of negation.

Features are generated given the categories of both p and h. One of four boolean

features is generated: BOTH 'NO' if both p and h quantifiers are *no*; ONE 'NO' if just one is; EXPAND if the h quantifier is "larger" (less restrictive) than the p quantifier; and CONTRACT if the reverse. Intuitively, BOTH 'NO' and EXPAND should favor entailments; ONE 'NO' and CONTRACT should disfavor. Unfortunately, these features are rarely useful on RTE data. Overwhelming, they are activated only for exact matches between definite or indefinite determiners. Very few quantifiers per se are identified.

Number, date, and time features. These are designed to recognize (mis-)matches between numbers, dates, and times, as in problems 1806 and 231 of figure 4.1. We do some normalization (e.g., of date representations) and have a limited ability to do fuzzy matching. In problem 1806, the mismatched years are correctly identified. Unfortunately, in problem 231, the significance of *over* is not grasped and a mismatch is reported.[7]

Alignment features. Our feature representation includes three real-valued features intended to represent the quality of the alignment: SCORE is the raw score returned from the alignment phase, while GOOD SCORE and BAD SCORE try to capture whether the alignment score is "good" or "bad" by computing the sigmoid function of the distance between the alignment score and hard-coded "good" and "bad" reference values.

4.4 Evaluation

We present results based on the PASCAL RTE1 Challenge, which was introduced in section 1.3.2. The RTE1 Challenge recommended two evaluation metrics: raw accuracy and confidence weighted score (CWS).[8] The CWS is computed as follows: for each positive integer k up to the size of the test set, we compute accuracy over the k most confident predictions. The CWS is then the average, over k, of these

[7] Later versions of the Stanford RTE system handle *over* correctly.

[8] In subsequent RTE competitions, the use of CWS was abandoned in favor of average precision.

Algorithm	RTE1 Dev Set		RTE1 Test Set	
	Acc %	CWS %	Acc %	CWS %
Random	50.0	50.0	50.0	50.0
Jijkoun et al. 05	61.0	64.9	55.3	55.9
Raina et al. 05	57.8	66.1	55.5	63.8
Haghighi et al. 05	—	—	56.8	61.4
Bos & Markert 05	—	—	57.7	63.2
Stanford RTE, alignment only	58.7	59.1	54.5	59.7
Stanford RTE, hand-tuned	60.3	65.3	**59.1**	**65.0**
Stanford RTE, learning	61.2	74.4	**59.1**	**63.9**

Table 4.1: Performance of various systems on the RTE1 development and test sets. The columns show accuracy and confidence weighted score (see text).

partial accuracies. Like raw accuracy, it lies in the interval $[0, 1]$, but it will exceed raw accuracy to the degree that predictions are well-calibrated.

Table 4.1 shows results for a range of systems and testing conditions. We report accuracy and CWS on each RTE1 data set. The baseline for all experiments is random guessing, which always attains 50% accuracy. Table 4.1 also displays comparable results from RTE1 submissions based on lexical similarity (Jijkoun and de Rijke 2005), graph alignment (Haghighi et al. 2005), weighted abduction (Raina et al. 2005), and theorem proving (Bos and Markert 2005a).

We then show results for the Stanford RTE system under several different training regimes. The row labeled "alignment only" describes experiments in which all features except the alignment score are turned off. We predict entailment just in case the alignment score exceeds a threshold which is optimized on development data. "Hand-tuning" describes experiments in which all features are on, but no training occurs; rather, weights are set by hand, according to human intuition. Finally, "learning" describes experiments in which all features are on, and feature weights are trained on the development data. The figures reported for development data performance therefore reflect overfitting; while such results are not a fair measure of overall performance, they can help us assess the adequacy of our feature set: if our features have

failed to capture relevant aspects of the problem, we should expect poor performance even when overfitting. It is therefore encouraging to see CWS above 70%. Finally, the figures reported for test data performance are the fairest basis for comparison. These are significantly better than our results for alignment only (Fisher's exact test, $p < 0.05$), indicating that we gain real value from our features. However, the gain over comparable results from other teams is not significant at the $p < 0.05$ level.

A curious observation is that the results for hand-tuned weights are as good or better than results for learned weights. A possible explanation runs as follows. Most of the features represent high-level patterns which arise only occasionally. Because the training data contains only a few hundred examples, many features are active in just a handful of instances; their learned weights are therefore quite noisy. Indeed, a feature which is expected to favor entailment may even wind up with a negative weight: the modal feature WEAK YES is an example. As shown in table 4.2, the learned weight for this feature was strongly negative—but this resulted from a single training example in which the feature was active but the hypothesis was not entailed. In such cases, we shouldn't expect good generalization to test data, and human intuition about the "value" of specific features may be more reliable.

Table 4.2 shows the values learned for selected feature weights. As expected, the features ADDED ADJUNCT IN 'ALL' CONTEXT, MODAL YES, and P IS FACTIVE were all found to be strong indicators of entailment, while DATE INSERT, DATE MODIFIER INSERT, WIDENING FROM P TO H all indicate lack of entailment. Interestingly, P HAS NEG MARKER and P & H DIFF POLARITY were also found to disfavor entailment; while this outcome is sensible, it was not anticipated or designed.

The real-valued alignment features GOOD SCORE and BAD SCORE proved to be highly informative. However, the more basic SCORE feature from which these are derived got a weight near 0, suggesting that it carries little additional information.

4.5 Conclusion

Many researchers have explored approaches to the problem of natural language inference which work by aligning semantic graphs, using a locally-decomposable alignment

Category	Feature	Weight
Adjunct	added adjunct in 'all' context	+1.40
Date	date mismatch	+1.30
Alignment	good score	+1.10
Modal	yes	+0.70
Modal	no	+0.51
Factive	p is factive	+0.46
...	...	...
Polarity	p & h same polarity	−0.45
Modal	don't know	−0.59
Quantifier	widening from p to h	−0.66
Polarity	p has neg marker	−0.66
Polarity	p & h diff polarity	−0.72
Alignment	bad score	−1.53
Date	date modifier insert	−1.57
Modal	weak yes	−1.92
Date	date insert	−2.63

Table 4.2: Learned weights for selected features. Positive weights favor entailment. Weights near 0 are omitted. Based on training on the PASCAL RTE1 development set, with a Gaussian smoothing parameter of 20.

score as a proxy for strength of entailment. We have argued that such models suffer from three crucial limitations: an assumption of monotonicity, an assumption of locality, and a confounding of alignment and entailment determination.

We have described the Stanford RTE system, which extends alignment-based systems while attempting to address these limitations. After finding the best alignment between premise and hypothesis, the Stanford RTE system extracts high-level semantic features of the entailment problem, and inputs these features to a statistical classifier to make an entailment decision. Using this multi-stage architecture, we report results on the PASCAL RTE1 data which surpass previously-reported results for alignment-based systems.

Chapter 5

Entailment relations

5.1 Introduction

This chapter marks a shift in direction.[1] Whereas the bag-of-words model (chapter 2) and the Stanford RTE system (chapter 4) (and, indeed, most existing RTE systems) approach the NLI task via approximate measures of the lexical and syntactic similarity of hypothesis h to premise p, in this chapter we take our first steps toward developing a more precise model of natural language entailment[2] based on *natural logic*. We begin by considering the fundamental representations to be used. What kind of answer do we want a model of entailment to give us? For that matter, what kinds of questions should we be able to ask of it? More precisely: if we view our entailment model as a function, what should be the types of its inputs and output?

The simplest kind of entailment model represents entailment as a binary relation

[1] The material in this chapter is derived in large part from (MacCartney and Manning 2009).

[2] The attentive reader may note a slight shift in terminology, from natural language *inference* to natural language *entailment*. In formal semantics, entailment is conventionally defined more narrowly than inferability: we say that p entails h just in case p cannot be true unless h is true. While the term "textual entailment" has frequently been used to describe the problem of natural language inference (NLI), we believe that "inference" is a more appropriate term for the general problem, since NLI problems can hinge on presuppositions and implicatures, as well as entailments. However, the model we develop in this chapter and the next focuses principally on entailments, and has little to say about other forms of inference; therefore it will be convenient to use the term "entailment".

between declarative expressions (sentences or short paragraphs). Viewed as a function, such a model accepts as input an ordered pair $\langle p, h \rangle$ of declarative expressions and returns as output a Boolean value indicating whether p entails h. Indeed, this problem representation is implicit in the most common definitions of NLI task, such as RTE. However, various elaborations are possible. Some work in NLI (Cooper et al. 1996) assumes a three-valued output space of entailment relations, while other work (Sánchez Valencia 1995) considers entailments between words and phrases as well as sentences.

In this chapter, we describe the problem representations used in previous approaches to NLI, and argue that none of them is fully adequate for our purposes. Instead, we introduce a problem representation in which the input is an ordered pair of linguistic expressions of any semantic type (words, phrases, or sentences) and the output is one of seven mutually exclusive *basic entailment relations*. We explain how to define these entailment relations for expressions of every semantic type, and we outline a relation algebra which enables us to join entailment relations across chains of expressions.

5.2 Representations of entailment

5.2.1 Entailment as two-way classification

The simplest formulation of the NLI task is as a binary decision problem: the relation between a premise p and a hypothesis h is to be classified as either *entailment* (h follows from p) or *non-entailment* (h does not follow from p). This formulation is used, for example, in the well-known RTE Challenge, in which h is typically a single sentence, and p may consist of one or (occasionally) several sentences. In this simple formulation, we make no distinction between unidirectional entailment and bidirectional entailment (or equivalence). Nor do we distinguish contradiction (if p is true, then h cannot be true) from simple non-entailment (if p is true, then h may or may not be true).

Though such a simple conception of entailment hardly needs to be formalized, we'll

do so, in order to facilitate comparison with the more complex representations of entailment to follow. Formally, the output labels ENTAILMENT and NON-ENTAILMENT may be interpreted as denoting relations between (that is, sets of ordered pairs of) declarative expressions. If $\mathbf{Dom}_T$ denotes the domain of declarative expressions (semantic type T), and $\mathbf{Dom}_T^2$ denotes the Cartesian product of $\mathbf{Dom}_T$ with itself, then we can define these relations as:

$$\text{ENTAILMENT} \stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \models h\}$$
$$\text{NON-ENTAILMENT} \stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \not\models h\}$$

Obviously, NON-ENTAILMENT is the set complement of ENTAILMENT, and thus every $\langle p, h\rangle \in \mathbf{Dom}_T^2$ can be assigned to exactly one of these two relations.

5.2.2 Entailment as three-way classification

The three-way formulation of the NLI task refines the binary formulation by dividing non-entailment into *contradiction* and *compatibility*. For example, in the FraCaS data set (Cooper et al. 1996), problems are annotated with one of three labels:[3]

- YES denotes entailment: h can be inferred from p.
- NO denotes contradiction: the negation of h can be inferred from p.
- UNK denotes compatibility: neither h nor its negation can be inferred from p.

Condoravdi et al. (2003) argued for the importance of entailment and contradiction detection (ECD) as a necessary condition for natural language understanding, and explored theoretical approaches to the task, but do not report empirical results.

To our knowledge, Harabagiu et al. (2006) provide the first empirical results for contradiction detection, but they evaluate their system on constructed corpora containing two specific kinds of contradiction: those featuring negation and those formed by paraphrases. A more general treatment is given by de Marneffe et al. (2008), who

[3] A few FraCaS problems do not fit neatly into any of these three categories. See section 7.8.1.

develop a typology of contradictions and report results on both contradictions found in the RTE data and "in the wild".

Over time, the three-way formulation has gained considerable traction in the NLI community. It was employed in the AQUAINT KBEval data sets (Crouch et al. 2005), and was introduced as a pilot task in the RTE3 competition (Giampiccolo et al. 2007, Voorhees 2008). In the RTE4 competition (Dang and Giampiccolo 2008), the three-way classification task became the standard (though two-way classification was retained as an optional task).

Formally, the output labels of the three-way formulation may be interpreted as denoting entailment relations defined as follows:

$$\begin{aligned}
\textsc{entailment} &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \models h\} \\
\textsc{contradiction} &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \models \neg h\} \\
\textsc{compatibility} &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \not\models h \wedge p \not\models \neg h\}
\end{aligned}$$

As in the binary formulation, the output labels form a partition of the input space, so that every $\langle p, h\rangle \in \mathbf{Dom}_T^2$ can be assigned to exactly one of the three relations.

5.2.3 Entailment as a containment relation

The monotonicity calculus of Sánchez Valencia (1991) carves things up differently. It interprets entailment as a *semantic containment* relation $\sqsubseteq$ analogous to the set containment relation $\subseteq$, and seeks to explain inversions of the containment relation resulting from downward-monotone operators, such as negation. It thus emphasizes the distinction between forward entailment ($p \sqsubseteq h$) and reverse entailment ($p \sqsupseteq h$). Unlike the three-way formulation, however, it lacks any way to represent contradiction (semantic exclusion).

A model of entailment based on the monotonicity calculus would thus have a different output space than either of the simpler formulations above. While the $\sqsubseteq$ and $\sqsupseteq$ relations overlap, they can be factored into three mutually-exclusive relations, which we might label $\equiv$, $\sqsubset$, and $\sqsupset$. For completeness, we should also define

a NO-CONTAINMENT relation which holds exactly when none of the other relations hold. The output space for the monotonicity calculus then consists of four relations:

$$\begin{aligned}
\equiv &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \models h \wedge h \models p\} \\
\sqsubset &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \models h \wedge h \not\models p\} \\
\sqsupset &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \not\models h \wedge h \models p\} \\
\text{NO-CONTAINMENT} &\stackrel{\text{def}}{=} \{\langle p, h\rangle \in \mathbf{Dom}_T^2 : p \not\models h \wedge h \not\models p\}
\end{aligned}$$

However, these definitions are incomplete. The monotonicity calculus differs from the simpler formulations above not only with respect to the output space, but also with respect to the input space: it defines the semantic containment relation $\sqsubseteq$ for expressions of every semantic type, including individual words and phrases as well as complete sentences. It achieves this using a recursive definition, beginning from atomic types and building up to functional types:

1. If $x, y \in \mathbf{Dom}_T$, then $x \sqsubseteq y$ iff $x =$ FALSE or $y =$ TRUE.

   (For truth values, entailment is equivalent to material implication: $x \sqsubseteq y$ iff $x \rightarrow y$. Thus FALSE $\sqsubseteq$ TRUE.)

2. If $x, y \in \mathbf{Dom}_E$, then $x \sqsubseteq y$ iff $x = y$.

   (Expressions denoting entities are mutually entailing just in case they denote the same entity; otherwise they are unrelated.)

3. If $x, y \in \mathbf{Dom}_{A \to B}$, then $x \sqsubseteq y$ iff for all $a \in \mathbf{Dom}_A$, $x(a) \sqsubseteq y(a)$.

   (One function entails another if each of its outputs entails the corresponding output of the other function.)

4. Otherwise, $x \not\sqsubseteq y$ and $y \not\sqsubseteq x$.

   (Expressions having different semantic types are unrelated.)

Defining semantic containment for expressions of every semantic type is necessary to the goal of the monotonicity calculus, which aims to explain the impact of inversions of monotonicity (even when nested) in a compositional manner. For example,

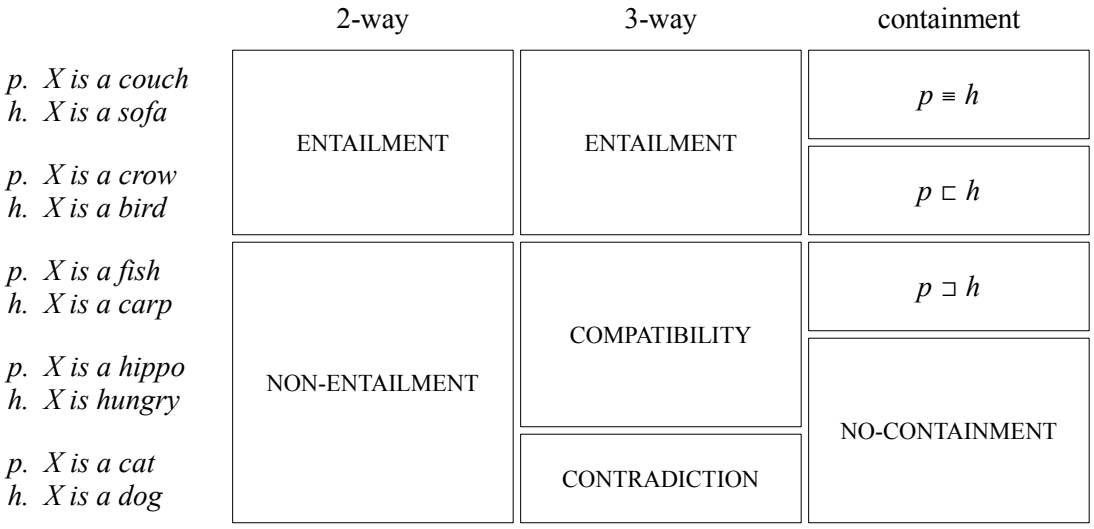

Figure 5.1: A comparison of three representations of entailment relations used in past work. The rows show five examples of simple inference problems; the columns show the entailment relation assigned to each problem by the three formulations presented in sections 5.2.1, 5.2.2, and 5.2.3.

the monotonicity calculus lets us conclude from *dancing* $\sqsupseteq$ *waltzing* that *isn't dancing* $\sqsubseteq$ *isn't waltzing* and thus *Matilda isn't dancing* $\sqsubseteq$ *Matilda isn't waltzing.* (See section 6.2.1 for a fuller exposition.)

5.2.4 The best of both worlds

Clearly, the representation of entailment used in the monotonicity calculus—with four mutually-exclusive entailment relations defined over expressions of all semantic types—is far more expressive than the two-way or three-way formulations. On the other hand, the three-way formulation makes a useful distinction between contradiction and mere non-entailment which is absent from the monotonicity calculus. (Figure 5.1 shows a comparison of these different representations.)

The inability to express semantic exclusion severely limits the deductive power of the monotonicity calculus. Consider the following sequence of propositions:

(1) a. *Garfield is a cat.*

b. *Garfield is a mammal.*

c. *Garfield is not a fish.*

d. *Garfield is not a carp.*

Clearly, the first proposition (1a) entails the last (1d). However, the monotonicity calculus lacks the machinery to recognize this. It can make the inference from (1a) to (1b) (using the semantic containment $cat \sqsubset mammal$), and from (1c) to (1d) (using the semantic containment $fish \sqsupset carp$). But it cannot make the simple inference from (1b) to (1c), which hinges on the semantic exclusion between *mammal* and *fish*.

Of course, the first-order predicate calculus renders such inferences trivial, but using formal logic requires full semantic interpretation, which is contrary to the natural logic approach. We'd like to preserve the spirit of the monotonicity calculus, while enhancing its power. Ideally, then, we'd like to have the best of both worlds, by defining an inventory of entailment relations which satisfies the following desiderata:

- It should preserve the semantic containment relations of the monotonicity calculus;
- It should augment these with relations expressing semantic exclusion;
- It should be complete, so that every pair of expressions can be assigned to some relation; and
- Indeed, it should partition the space of ordered pairs of expressions, that is, the relations it contains should be not only exhaustive but also mutually exclusive, so that we can define a function which maps any ordered pair of expressions to a single entailment relation.

We'll approach these goals somewhat indirectly. Taking inspiration from Sánchez Valencia, we'll focus first on set relations. We'll define an inventory of set relations which satisfy our desiderata, and then return to entailment relations in section 5.4.

5.3 The 16 elementary set relations

How can we define an inventory of set relations satisfying the desiderata outlined in section 5.2.4? Of course, relations between sets include familiar relations such as the (strict) subset relation ($x \subset y$) and the equivalence relation ($x = y$). We can also define an exclusion relation, which holds between non-overlapping sets ($x \cap y = \emptyset$). And we could define more arcane relations, such as the relation between two sets whose intersection is a singleton ($|x \cap y| = 1$). In fact, the space of possible set relations is very large. Recall that a relation is a set of ordered pairs. A universe U contains $2^{|U|}$ sets, $4^{|U|}$ ordered pairs of sets, and thus $2^{4^{|U|}}$ possible set relations. Assuming $|U| \gg 1$, then, the number of possible set relations is huge. However, only a small fraction of these will be of any particular interest. From the space of all possible set relations, we'd like to identify a subset which (a) include familiar and useful relations expressing equivalence, containment, and exclusion, and (b) form a partition of the space of ordered pairs of sets, so that every ordered pair of sets can be assigned to exactly one relation. One way to do this is as follows.

Each ordered pair $\langle x, y \rangle$ of subsets of U divides U into (up to) four partitions: $\overline{x} \cap \overline{y}$, $\overline{x} \cap y$, $x \cap \overline{y}$, and $x \cap y$, each of which may be empty or non-empty.[4] Let us label each of these partitions by a two-bit string encoding whether it is inside or outside each of the two sets:

label	definition	meaning
00	$\overline{x} \cap \overline{y}$	in neither x nor y
01	$\overline{x} \cap y$	in y but not x
10	$x \cap \overline{y}$	in x but not y
11	$x \cap y$	in both x and y

Every ordered pair of sets $\langle x, y \rangle$ can now be assigned to one of 16 equivalence classes, according to whether each of these four partitions is empty or not. Let us label each equivalence class with R subscripted by a 4-bit string, following the convention that the nth bit of the subscript (starting from 0) denotes the non-emptiness of the partition whose label is the bit string with binary value n. (Thus R_{1101} denotes the

[4] $\overline{x}$ denotes the *set complement* of set x, defined by the constraints $x \cap \overline{x} = \emptyset$ and $x \cup \overline{x} = U$.

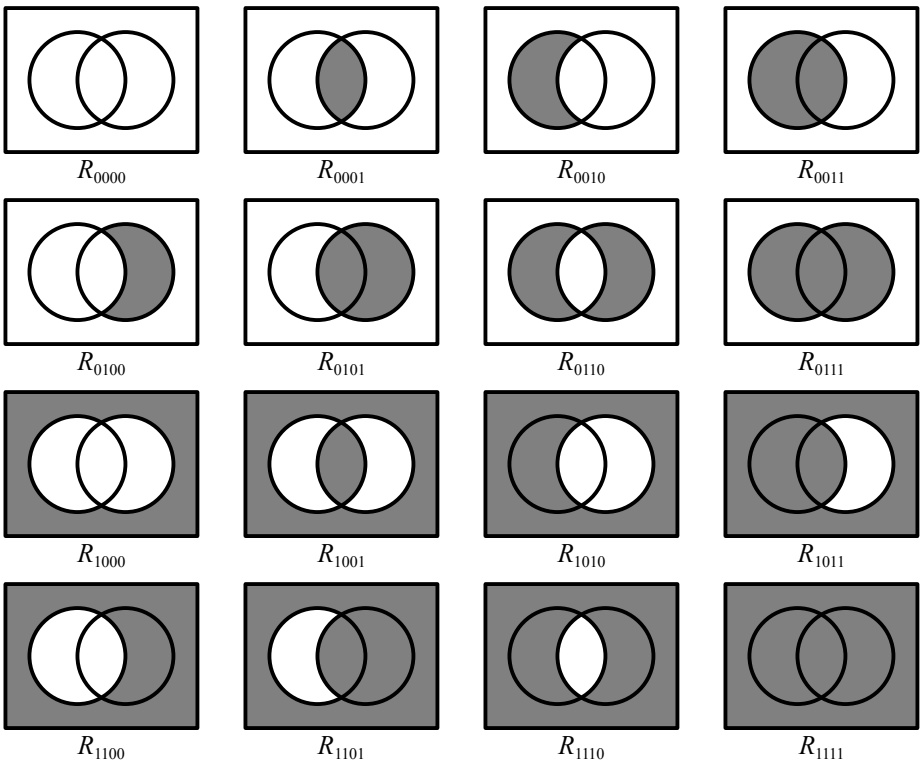

Figure 5.2: The 16 elementary set relations, represented by Johnston diagrams. Each box represents the universe U, and the two circles within the box represent the sets x and y. A region is white if it is empty, and shaded if it is non-empty. Thus in the diagram labeled R_{1101}, only the region $x \cap \overline{y}$ is empty, indicating that $\emptyset \subset x \subset y \subset U$.

equivalence class in which only partition 10 is empty.) These equivalence classes are depicted graphically in figure 5.2.

In fact, each of these equivalence classes is a set relation, that is, a set of ordered pairs of sets. We will refer to these 16 set relations as the *elementary set relations*, and we will denote this set of 16 relations by $\mathfrak{R}$. By construction, the relations in $\mathfrak{R}$ are both mutually exhaustive (every ordered pair of sets belongs to some relation in $\mathfrak{R}$) and mutually exclusive (no ordered pair of sets belongs to two different relations in $\mathfrak{R}$). Thus, every ordered pair of sets can be assigned to exactly one relation in $\mathfrak{R}$.

relation	constraint on x	constraint on y	constraint on $\langle x, y\rangle$
R_{0000}	$\emptyset = x = U$	$\emptyset = y = U$	$x = y$
R_{0001}	$\emptyset \subset x = U$	$\emptyset \subset y = U$	$x = y$
R_{0010}	$\emptyset \subset x = U$	$\emptyset = y \subset U$	$x \supset y$
R_{0011}	$\emptyset \subset x = U$	$\emptyset \subset y \subset U$	$x \supset y$
R_{0100}	$\emptyset = x \subset U$	$\emptyset \subset y = U$	$x \subset y$
R_{0101}	$\emptyset \subset x \subset U$	$\emptyset \subset y = U$	$x \subset y$
R_{0110}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \cap y = \emptyset \wedge x \cup y = U$
R_{0111}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \cap y \neq \emptyset \wedge x \cup y = U$
R_{1000}	$\emptyset = x \subset U$	$\emptyset = y \subset U$	$x = y$
R_{1001}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x = y$
R_{1010}	$\emptyset \subset x \subset U$	$\emptyset = y \subset U$	$x \supset y$
R_{1011}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \supset y$
R_{1100}	$\emptyset = x \subset U$	$\emptyset \subset y \subset U$	$x \subset y$
R_{1101}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \subset y$
R_{1110}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \cap y = \emptyset \wedge x \cup y \neq U$
R_{1111}	$\emptyset \subset x \subset U$	$\emptyset \subset y \subset U$	$x \cap y \neq \emptyset \wedge x \cup y \neq U$
			$\ldots \wedge x \not\subseteq y \wedge x \not\supseteq y$

Table 5.1: The 16 elementary set relations between sets x and y in universe U, described in terms of set-theoretic constraints. Constraints which appear in gray are less salient, but still active.

Moreover, provided that $|U| \geq 4$, the relations in $\mathfrak{R}$ are distinct.[5]

Table 5.1 describes the properties of the 16 elementary set relations in terms of set-theoretic constraints.

5.3.1 Properties of the elementary set relations

We can make a number of observations about the relations in $\mathfrak{R}$:

[5]If $|U| < 4$, some of the relations in $\mathfrak{R}$ will be extensionally equivalent. For example, if $|U| = 1$, then all relations except R_{0001}, R_{0010}, R_{0100}, and R_{1000} will have extension $\emptyset$.

The empty relation. Relation R_{0000} is an extremely degenerate case: it is the relation between two sets which are both empty ($= \emptyset$) and universal ($= U$). If the universe is non-empty (the usual case), no pairs of sets satisfy these constraints, and the relation is empty. In the unusual case of an empty universe, this relation contains exactly one pair of sets (namely $\langle \emptyset, \emptyset \rangle$), and all other relations are empty.

Singleton relations. Relations R_{0001}, R_{0010}, R_{0100}, and R_{1000} cover the cases where the universe is non-empty and both x and y are either empty or universal. Each of these relations therefore contains exactly one pair of sets.

Other edge cases. Relations R_{0011}, R_{0101}, R_{1010}, and R_{1100} cover the cases where the universe is non-empty and one (but not both) of x and y is either empty or universal. Each of these relations therefore has cardinality $2^{|U|} - 2$.

Degenerate relations. The nine relations in $\mathfrak{R}$ mentioned so far (namely, R_{0000}, R_{0001}, R_{0010}, R_{0011}, R_{0100}, R_{0101}, R_{1000}, R_{1010}, and R_{1100}) are boundary cases in which either x or y is either empty or universal. We therefore describe them as *degenerate* relations. Note that a set which is empty or universal corresponds to a predicate which is semantically vacuous (such as *round square cupola* or *exists*); consequently, we will later focus our attention on the remaining seven relations. (See section 5.5 for further discussion.)

Relations expressing equivalence. In four of the relations in $\mathfrak{R}$ (namely, R_{0000}, R_{0001}, R_{1000}, and R_{1001}), x and y are equivalent (that is, $x = y$).

Relations expressing containment. In four of the relations in $\mathfrak{R}$ (namely, R_{0010}, R_{0011}, R_{1010}, and R_{1011}), x strictly contains y (that is, $x \supset y$). In another four relations (namely, R_{0100}, R_{0101}, R_{1100}, and R_{1101}), x is strictly contained by y (that is, $x \subset y$).

Relations expressing exclusion. In eight of the relations in $\mathfrak{R}$ (namely, those in the first and third columns of figure 5.2), x and y are mutually exclusive: that is,

$x \cap y = \emptyset$.

Relations expressing exhaustion. In eight of the relations in $\mathfrak{R}$ (namely, those in the first two rows of figure 5.2), x and y are mutually exhaustive: that is, $x \cup y = U$.

The independence relation. Relation R_{1111} is unique in that it expresses non-equivalence, non-containment, non-exclusion, and non-exhaustion. We call this the "independence" relation because if sets x and y belong to this relation, then information about whether or not a given element is contained in x conveys no information about whether or not it is contained in y, and vice-versa. Intuitively, independence is the least informative relation, in a sense we'll make precise in section 5.3.2.

Converses and symmetric relations. We describe relations R and S as *converses* iff $\forall x, y : \langle x, y\rangle \in R \Leftrightarrow \langle y, x\rangle \in S$. A relation which is its own converse is described as *symmetric*. If R is a relation in $\mathfrak{R}$, then the bit-string label of the converse of R is obtained from the bit-string label of R by swapping the two middle bits. It follows that a relation in $\mathfrak{R}$ is symmetric just in case the two middle bits of its bit-string label are identical. Thus, eight relations in $\mathfrak{R}$ are symmetric: namely, R_{0000}, R_{0001}, R_{0110}, R_{0111}, R_{1000}, R_{1001}, R_{1110}, and R_{1111}. Among the remaining eight relations, R_{0010} and R_{0100} are converses, R_{0011} and R_{0101} are converses, R_{1010} and R_{1100} are converses, and R_{1011} and R_{1101} are converses.

Duals under negation. We describe set relations R and S as *duals under negation* iff $\forall x, y : \langle x, y\rangle \in R \Leftrightarrow \langle \overline{x}, \overline{y}\rangle \in S$. Two relations in $\mathfrak{R}$ are duals under negation if and only if their bit-string labels are reverses. Thus R_{1011} and R_{1101} are duals, while R_{1001} is self-dual. The significance of this duality will become apparent later.

We have defined the relations in $\mathfrak{R}$ as relations between sets, but we can easily recast them as relations between set-denoting linguistic expressions, (e.g., simple predicates). In this form, we refer to the elements of $\mathfrak{R}$ as the *elementary entailment relations*. (We consider in section 5.4 how to define entailment relations for expressions of all semantic types.)

5.3.2 Cardinalities of the elementary set relations

Intuitively, R_{1111} is the least informative set relation. If two sets are selected at random from a large universe, it's very likely that they belong to R_{1111}. In this sense, we might call R_{1111} the "default" set relation—it's the set relation you assume to hold when you don't have any information to the contrary.

We can provide some justification for this claim by a combinatoric argument. In a universe of n objects, there are 2^n possible sets, and 4^n possible ordered pairs of sets. Every such pair can be assigned to exactly one of the 16 relations in $\mathfrak{R}$, and we can use combinatorics to count how many pairs belong to each relation. Of the 4^n possible ordered pairs of sets:

- One relation in $\mathfrak{R}$ (namely R_{1111}) contains $4^n - (4 \cdot 3^n) + (6 \cdot 2^n) - 4$ pairs of sets. Note that as n approaches infinity, the proportion of pairs of sets belonging to this relation approaches 1.

- Then, four relations in $\mathfrak{R}$ (namely R_{0111}, R_{1011}, R_{1101}, and R_{1110}) contain $3^n - (3 \cdot 2^n) + 3$ pairs each. These are the non-degenerate relations expressing non-exclusive exhaustion, non-exhaustive exclusion, and containment. Note that each has a label containing one 0 and three 1s.

- Next, six relations in $\mathfrak{R}$ (namely R_{0011}, R_{0101}, R_{0110}, R_{1001}, R_{1010}, and R_{1100}) contain $2^n - 2$ pairs each. Two of these are the non-degenerate relations expressing equivalence and negation; the remainder are degenerate relations. Note that each has a label containing two 0s and two 1s.

- Then, four relations in $\mathfrak{R}$ (namely R_{0001}, R_{0010}, R_{0100}, and R_{1000}) contain just 1 pair each. These are the degenerate singleton relations. Note that each has a label containing three 0s and one 1.

- Finally, unless the universe is empty, the relation R_{0000} contains 0 pairs.

5.4 From set relations to entailment relations

In section 5.3, we defined a collection $\mathfrak{R}$ of relations between sets. Our true aim, however, is to define an inventory of *entailment relations*, which are relations between linguistic expressions. And, following Sánchez Valencia, we would like to define our entailment relations for expressions of every semantic type—not merely expressions which denote sets (e.g., predicates), but also expressions which denote truth values (e.g., propositions), entities (e.g., names), and more complex types.

A conventional account of semantic types can be briefly summarized as follows. Every linguistic expression has a semantic type, which is either an atomic type or a functional type. There are two atomic types: expressions which denote truth values have atomic type T, while expressions which denote entities have atomic type E.[6] An expression which has functional type $\phi \to \psi$ combines with an argument of type ϕ (the input type) to produce a compound expression of type ψ (the output type). For example, a predicate such as *is rich* has semantic type $E \to T$, and will combine with an expression denoting an entity (such as *Buffett*) to produce a compound expression which denotes a truth value (*Buffett is rich*). The input and output types of a functional type may themselves be functional types. For example, $(E \to T) \to (E \to T)$ is the type of an adverb (such as *partly*) which can modify an adjective of type $E \to T$ (such as *red*) to produce a compound expression of type $E \to T$ (*partly red*).

Let us define *entailment relation* to mean any set of ordered pairs of linguistic expressions, subject to the constraint that in every pair, both elements belong to the same semantic type.[7] Our goal here is to extend the definitions of the relations in $\mathfrak{R}$ from set relations to entailment relations. For semantic types which can be interpreted as denoting characteristic functions of sets, this is completely straightforward: the set-theoretic definitions shown in table 5.1 can be applied directly. This includes all functional types whose final output is type T (a truth value), and includes most of the

[6] Some authors define additional atomic types, such as numbers, but we'll keep things simple.

[7] Clearly, we're abusing terminology somewhat, since many relations which fit this definition will contain pairs of expressions in which neither element entails the other. However, in most of the entailment relations with which we will be concerned, it is at least the case that one element of every pair in the relation conveys some information, positive or negative, about the other. And, for various reasons, every alternative term for this concept seems unsatisfactory.

functional types encountered in semantic analysis: $E \to T$ (common nouns, adjectives, and intransitive verbs), $E \to E \to T$ (transitive verbs), $(E \to T) \to (E \to T)$ (adverbs), $(E \to T) \to (E \to T) \to T$ (binary generalized quantifiers), and so on. Note that it makes no difference whether T "stands alone" at the end of the type signature: for example, expressions of type $(E \to T) \to (E \to T)$ can be interpreted as denoting a subset of the Cartesian product of type $E \to T$ and type E.

The definitions can then be extended to other types by interpreting each type as if it were a type of set. For example, propositions (semantic type T) can be understood in model-theoretic terms as sets of models (or possible worlds). So if x and y are contingent propositions,[8] then x and y belong to relation R_{1001} iff they hold in exactly the same set of models; to relation R_{1101} iff y holds in every model where x holds (but not vice-versa); to relation R_{1110} iff there is no model where both x and y hold (but there is some model where neither holds); and so on. Likewise, entities (semantic type E) can be identified with singleton sets, with the result that two entity-denoting expressions belong to relation R_{1001} if they denote the same entity, or to relation R_{1110} otherwise.

5.5 The seven basic entailment relations

When interpreted as a collection of entailment relations, $\mathfrak{R}$ satisfies our desiderata for an inventory of entailment relations: first, it contains representations of both semantic containment and semantic exclusion; second, the relations in $\mathfrak{R}$ are (by design) both mutually exclusive and mutually exhaustive, so that every ordered pair of expressions can be assigned to exactly one relation in $\mathfrak{R}$. However, as an inventory of entailment relations, $\mathfrak{R}$ is somewhat unwieldy. 16 relations is rather a lot, and the symbols we've used to denote these relations (e.g. R_{1101}) are unfriendly. Can we make things more manageable?

[8] A proposition is *contingent* iff it is neither necessarily true (that is, true in all models) nor necessarily false (that is, true in no models).

5.5.1 The assumption of non-vacuity

Of the 16 relations in $\mathfrak{R}$, nine are in any case degenerate relations, in the sense defined in section 5.3.1. That is, they hold only between two expressions of which at least one has a denotation which is either empty or universal. Since expressions having empty denotations (e.g., *round square cupola*) or universal denotations (e.g., *exists*) fail to divide the world into meaningful categories, they can be regarded as semantically vacuous. Contradictions and tautologies may be common in logic textbooks, but they are rare in everyday speech. Most of the predicates we actually use (such as *red* or *heavy* or *French*) are not vacuous—they make a meaningful distinction between an in-group and an out-group. Thus, in a practical model of informal natural language inference, we will rarely go wrong by assuming the *non-vacuity* of the expressions we encounter.[9]

5.5.2 Defining the relations in $\mathfrak{B}$

Consequently, in the remainder of this chapter, and in chapters 6 and 7, we will focus our attention on the seven non-degenerate relations, which we designate as the set $\mathfrak{B}$ of *basic entailment relations*. Note that $\mathfrak{B}$ is a subset of $\mathfrak{R}$; it consists of those relations in $\mathfrak{R}$ in which the denotations of the related expressions are neither empty nor universal. As an inventory of entailment relations, $\mathfrak{B}$ preserves the key advantages of $\mathfrak{R}$: it includes representations of both semantic containment and semantic exclusion, and the relations in $\mathfrak{B}$ are both mutually exclusive and (given the assumption of non-vacuity) mutually exhaustive. But seven is a much more manageable size than 16, and we can now introduce names and symbols for the relations in $\mathfrak{B}$ that are more compact and intuitive than the symbols we have used for relations in $\mathfrak{R}$:

[9]The assumption of non-vacuity is closely related to the assumption of *existential import*, which is standard in traditional logic. For a defense of existential import in natural language semantics, see (Böttner 1988).

symbol[10]	name	example	set theoretic definition[11]	in $\mathfrak{R}$
$x \equiv y$	equivalence	*couch* $\equiv$ *sofa*	$x = y$	R_{1001}
$x \sqsubset y$	forward entailment	*crow* $\sqsubset$ *bird*	$x \subset y$	R_{1101}
$x \sqsupset y$	reverse entailment	*Asian* $\sqsupset$ *Thai*	$x \supset y$	R_{1011}
$x \wedge y$	negation	*able* $\wedge$ *unable*	$x \cap y = \emptyset \wedge x \cup y = U$	R_{0110}
$x \mid y$	alternation	*cat* $\mid$ *dog*	$x \cap y = \emptyset \wedge x \cup y \neq U$	R_{1110}
$x \smile y$	cover	*animal* $\smile$ *non-ape*	$x \cap y \neq \emptyset \wedge x \cup y = U$	R_{0111}
$x \# y$	independence	*hungry* # *hippo*	(all other cases)	R_{1111}

The relations in $\mathfrak{B}$ can be characterized as follows. First, we preserve the semantic containment relations ($\sqsubseteq$ and $\sqsupseteq$) of the monotonicity calculus, but factor them into three mutually exclusive relations: equivalence ($\equiv$), (strict) forward entailment ($\sqsubset$), and (strict) reverse entailment ($\sqsupset$). Second, we include two relations expressing semantic exclusion: negation ($\wedge$), or exhaustive exclusion, which is analogous to set complement; and alternation ($|$), or non-exhaustive exclusion. The next relation is cover ($\smile$), or non-exclusive exhaustion. Though its utility is not immediately obvious, it is the dual under negation (in the sense defined in section 5.3.1) of the alternation relation. Finally, the independence relation (#) covers all other cases: it expresses non-equivalence, non-containment, non-exclusion, and non-exhaustion. As noted in section 5.3.2, # is the least informative relation, in that it places the fewest constraints on its arguments.

[10]Selecting an appropriate symbol to represent each relation is a vexed problem. We sought symbols which (a) are easily approximated by a single ASCII character, (b) are graphically symmetric iff the relations they represent are symmetric, and (c) do not excessively abuse accepted conventions. The $\wedge$ symbol was chosen to evoke the logically similar bitwise `XOR` operator of the C programming language family; regrettably, it may also evoke the Boolean `AND` function. The $|$ symbol was chosen to evoke the Sheffer stroke commonly used to represent the logically similar Boolean `NAND` function; regrettably, it may also evoke the Boolean `OR` function. The $\sqsubset$ and $\sqsupset$ symbols were obviously chosen to resemble their set-theoretic analogs, but a potential confusion arises because some logicians use the horseshoe $\supset$ (with the *opposite* orientation) to represent material implication.

[11]Each relation in $\mathfrak{B}$ obeys the additional constraints that $\emptyset \subset x \subset U$ and $\emptyset \subset y \subset U$ (i.e., x and y are non-vacuous).

It is important to point out that our adoption of the assumption of non-vacuity, and our decision to focus on the seven non-degenerate relations in $\mathfrak{B}$, rather than the more complete set of relations in $\mathfrak{R}$, is a question of convenience, not necessity. The model of natural logic we develop in chapter 6 can easily be revised to accommodate vacuous expressions and relations between them, but then becomes somewhat unwieldy, both because more relations are required, and because the names and symbols used to describe them are less friendly.

5.6 Joining entailment relations

The model of natural language inference we wish to develop will require a general ability to combine entailment relations across a sequence of expressions. If we know that entailment relation R holds between x and y, and that entailment relation S holds between y and z, then what is the entailment relation between x and z? Recall that every entailment relation is a set of ordered pairs of expressions (of the same semantic type). The *join* of relations R and S, which we denote $R \bowtie S$,[12] can be defined formally as:

$$R \bowtie S \stackrel{\text{def}}{=} \{\langle x, z\rangle : \exists y\ (\langle x, y\rangle \in R \wedge \langle y, z\rangle \in S)\} \tag{5.1}$$

Some joins are quite intuitive. For example, it is immediately clear that:

$$\begin{aligned} & \sqsubset \bowtie \sqsubset = \sqsubset \\ & \sqsupset \bowtie \sqsupset = \sqsupset \\ & \wedge \bowtie \wedge = \equiv \\ \forall R \quad & R \bowtie \equiv = R \\ \forall R \quad & \equiv \bowtie R = R \end{aligned}$$

[12] In Tarskian relation algebra, this operation is known as *relation composition*, and is often represented by a semi-colon: $R\,;\,S$. To avoid confusion with semantic composition (to be discussed in section 6.2), we prefer to use the term *join* for this operation, by analogy to the database `JOIN` operation (also commonly represented by $\bowtie$). (To be precise, this operation is a *projected* join—but we prefer to keep the terminology simple.)

Other joins are less obvious, but still accessible to intuition. For example, the join of | and ^ is $\sqsubset$. This can be seen with the aid of Venn diagrams, or by considering simple examples: *fish* | *human* and *human* ^ *nonhuman*, thus *fish* $\sqsubset$ *nonhuman*. Other joins which can be apprehended in similar fashion include:

$$\begin{array}{ll} | \bowtie \wedge = \sqsubset & \wedge \bowtie | = \sqsupset \\ \wedge \bowtie \smile = \sqsubset & \smile \bowtie \wedge = \sqsupset \\ | \bowtie \smile = \sqsubset & \smile \bowtie | = \sqsupset \\ \wedge \bowtie \# = \# & \# \bowtie \wedge = \# \end{array}$$

(Note that the join operation is not always commutative: the order in which entailment relations are joined can make a difference to the result.)

5.6.1 "Nondeterministic" joins

Alas, we soon stumble upon an inconvenient truth: joining two relations in $\mathfrak{B}$ does not always produce a unique relation in $\mathfrak{B}$. Consider the join of | and |. If $x \mid y$ and $y \mid z$, we cannot say which relation in $\mathfrak{B}$ holds between x and z. They could be equivalent, or one might contain the other. They might be independent or alternative. All we can say for sure is that they are not exhaustive (since both are disjoint from y). The following examples illustrate the problem:

$x \mid y$	$y \mid z$	$x \ ? \ z$
gasoline | *water*	*water* | *petrol*	*gasoline* $\equiv$ *petrol*
pistol | *knife*	*knife* | *gun*	*pistol* $\sqsubset$ *gun*
dog | *cat*	*cat* | *terrier*	*dog* $\sqsupset$ *terrier*
rose | *orchid*	*orchid* | *daisy*	*rose* | *daisy*
woman | *frog*	*frog* | *Eskimo*	*woman* # *Eskimo*

The join of | and | is "nondeterministic", in the sense that $x \mid y$ and $y \mid z$ does not determine a unique relation in $\mathfrak{B}$ which holds between x and z. And there are many pairs of relations in $\mathfrak{B}$ which share this property. In fact, we'll show in section 5.6.2 that, of the 49 possible joins of two relations in $\mathfrak{B}$, 17 are "nondeterministic" in this sense.

However, to call such joins "nondeterministic" is a bit of a misnomer. The result of joining | and | is fully determined, and it is an entailment relation (in the sense defined in section 5.4), only it is not one of the seven relations in $\mathfrak{B}$. Rather, it is a *union* of such relations, specifically $\bigcup\{\equiv, \sqsubset, \sqsupset, |, \#\}$.[13] In fact, we'll soon demonstrate that joining two relations in $\mathfrak{B}$ always yields either another relation in $\mathfrak{B}$, or a union of relations in $\mathfrak{B}$.

5.6.2 Computing joins

So far, we have relied on intuition to determine the join of two entailment relations. How can we compute joins in a principled way? Equation 5.1 defines the join operation, but does not explain how to compute its result for a particular pair of inputs. We can achieve this by generalizing the approach of section 5.3 from two sets to three. In a universe U, we consider all possible ordered triples $\langle x, y, z\rangle$ of subsets of U. The sets x, y, and z divide the universe into (up to) eight partitions, each of which can be labeled by a three-bit string encoding whether it is inside or outside each of the three sets. (For example, the label 110 denotes $x \cap y \cap \overline{z}$.) Each triple $\langle x, y, z\rangle$ can now be assigned to one of 256 equivalence classes, according to whether each of these eight partitions is empty or not. Each equivalence class can be labeled with T subscripted by an 8-bit string, following the convention that the nth bit of the subscript (starting from 0) denotes the non-emptiness of the partition whose label is the bit string with binary value n. (Thus $T_{11111101}$ denotes the equivalence class in which only partition 110 is empty.) In fact, each of these equivalence classes is a *ternary set relation*, that is, a set of ordered triples of sets. Following the convention of section 5.3, we will refer to these 256 ternary set relations as the *elementary ternary set relations*, and we will use $\mathfrak{T}$ to denote this set of relations.

Each ternary set relation determines, via *projection*, three binary set relations: namely, the relation between x and y, the relation between y and z, and the relation

[13] We use this notation as shorthand for the union $\equiv \cup \sqsubset \cup \sqsupset \cup | \cup \#$. To be precise, the result of joining | and | is not identical with this union, but is a subset of it, since the union contains some pairs of sets (e.g., $\langle U \setminus a, U \setminus a\rangle$, for any $|a| = 1$) which cannot participate in the | relation. However, the approximation makes little practical difference.

between x and z. We use $\pi_{x,y}(T)$ to denote the projection of the ternary relation T onto a binary relation between x and y. The three such projections may be defined as follows:

$$\pi_{x,y}(T) \stackrel{\text{def}}{=} \{\langle x, y\rangle : \exists z\ (\langle x, y, z\rangle \in T)\}$$
$$\pi_{y,z}(T) \stackrel{\text{def}}{=} \{\langle y, z\rangle : \exists x\ (\langle x, y, z\rangle \in T)\}$$
$$\pi_{x,z}(T) \stackrel{\text{def}}{=} \{\langle x, z\rangle : \exists y\ (\langle x, y, z\rangle \in T)\}$$

By construction, each of $\pi_{x,y}(T)$, $\pi_{y,z}(T)$, and $\pi_{x,z}(T)$ must be one of the 16 elementary (binary) set relations in $\mathfrak{R}$ introduced in section 5.3.

We can now determine the join of two relations R and S in $\mathfrak{R}$ as follows:[14]

1. Find all ternary relations T in $\mathfrak{T}$ having binary relation R between x and y and binary relation S between y and z, that is, $\{T \in \mathfrak{T} : \pi_{x,y}(T) = R \wedge \pi_{y,z}(T) = S\}$.
2. For each such T, determine the binary relation between x and z, namely $\pi_{x,z}(T)$.
3. Form the union of the resulting binary relations.

We can combine these steps in a single formula:

$$\forall R, S \in \mathfrak{R} : \qquad R \bowtie S = \bigcup_{T \in \mathfrak{T} :\, \pi_{x,y}(T) = R \,\wedge\, \pi_{y,z}(T) = S} \pi_{x,z}(T)$$

Each of these three steps is easily implemented in code, and because $|\mathfrak{T}|$ is only 256, a complete join table for $\mathfrak{R}$ can be computed very quickly. Table 5.2 shows an example of computing the join of R_{1110} (which is identical with |) with itself.

Since $\mathfrak{B}$ (the set of seven basic entailment relations) is a subset of $\mathfrak{R}$ (the set of 16 elementary entailment relations), computing the joins of relations in $\mathfrak{B}$ requires no

[14] To be precise, the method described produces a (relatively tight) upper bound on the true join of R and S, not an exact result. As we observed in footnote 13, the relation which results from this method may include a few pairs $\langle x, z\rangle$ of expressions such that x cannot be an argument to R, or z to S. The exact join can be computed by exhaustively enumerating the extensions of R and S. However, the method described is far more efficient and intuitive, and the approximation involved makes little practical difference.

T	$\pi_{x,y}(T)$	$\pi_{y,z}(T)$	$\pi_{x,z}(T)$
$T_{01101000}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$
$T_{01101100}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1111} \rightarrow \#$
$T_{10100100}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1001} \rightarrow \equiv$
$T_{10101100}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1011} \rightarrow \sqsupset$
$T_{11100100}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1101} \rightarrow \sqsubset$
$T_{11101000}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$
$T_{11101100}$	$R_{1110} \rightarrow \mid$	$R_{1110} \rightarrow \mid$	$R_{1111} \rightarrow \#$

Table 5.2: An example of computing the join of two relations in $\mathfrak{R}$ using elementary ternary set relations. The first column shows a selection of relations T in $\mathfrak{T}$. The remaining columns show the projection of each T onto binary relations between x and y, y and z, and x and z. Arrows indicate the mapping from each relation in $\mathfrak{R}$ to the corresponding relation in $\mathfrak{B}$. The seven rows shown represent all and only those relations T in $\mathfrak{T}$ having $\pi_{x,y}(T) = \pi_{x,y}(T) = R_{1110}$. We can therefore conclude that $R_{1110} \bowtie R_{1110} = \bigcup\{R_{1001}, R_{1011}, R_{1101}, R_{1110}, R_{1111}\}$. Since R_{1110} is identical with $\mid$, we can further conclude that $\mid \bowtie \mid = \bigcup\{\equiv, \sqsubset, \sqsupset, \mid, \#\}$.

additional work. The complete join table for the relations in $\mathfrak{B}$ is shown in table 5.3. Note that joining two relations in $\mathfrak{B}$ always yields either a relation in $\mathfrak{B}$ or a union thereof; the "degenerate" relations in $\mathfrak{R}$ do not appear.

5.6.3 Unions of the basic entailment relations

Since entailment relations are just sets of ordered pairs of linguistic expressions having the same semantic type, the union of the relations in any subset of $\mathfrak{B}$ is itself an entailment relation, even if it is not an element of $\mathfrak{B}$. We will refer to (non-trivial) unions of relations in $\mathfrak{B}$ as *union relations*.[15] Of the 49 possible joins of relations in $\mathfrak{B}$, 32 yield a relation in $\mathfrak{B}$, while 17 yield a union relation not in $\mathfrak{B}$, with larger unions conveying less information. Union relations can be further joined, and we can establish that the smallest set of relations which contains $\mathfrak{B}$ and is closed under joining contains

[15]Some union relations hold intrinsic interest. For example, in the three-way formulation of the NLI task described in section 5.2.2, the ENTAILMENT relation can be identified with $\bigcup\{\equiv, \sqsubset\}$; the CONTRADICTION relation with $\bigcup\{^\wedge, \mid\}$; and the COMPATIBILITY relation with $\bigcup\{\sqsupset, \smile, \#\}$.

$\bowtie$	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	$\#$
$\sqsubset$	$\sqsubset$	$\sqsubset$	$\equiv\sqsubset\sqsupset\mid\#$	$\mid$	$\mid$	$\sqsubset\wedge\mid\smile\#$	$\sqsubset\mid\#$
$\sqsupset$	$\sqsupset$	$\equiv\sqsubset\sqsupset\smile\#$	$\sqsupset$	$\smile$	$\sqsupset\wedge\mid\smile\#$	$\smile$	$\sqsupset\smile\#$
$\wedge$	$\wedge$	$\smile$	$\mid$	$\equiv$	$\sqsupset$	$\sqsubset$	$\#$
$\mid$	$\mid$	$\sqsubset\wedge\mid\smile\#$	$\mid$	$\sqsubset$	$\equiv\sqsubset\sqsupset\mid\#$	$\sqsubset$	$\sqsubset\mid\#$
$\smile$	$\smile$	$\smile$	$\sqsupset\wedge\mid\smile\#$	$\sqsupset$	$\sqsupset$	$\equiv\sqsubset\sqsupset\smile\#$	$\sqsupset\smile\#$
$\#$	$\#$	$\sqsubset\smile\#$	$\sqsupset\mid\#$	$\#$	$\sqsupset\mid\#$	$\sqsubset\smile\#$	$\equiv\sqsubset\sqsupset\wedge\mid\smile\#$

Table 5.3: The join table for the seven basic entailment relations in $\mathfrak{B}$. For compactness, we omit the union notation; thus $\sqsubset\mid\#$ stands for $\bigcup\{\sqsubset, \mid, \#\}$.

just 16 relations.[16] One of these is the total relation $\bigcup\{\equiv, \sqsubset, \sqsupset, \wedge, \mid, \smile, \#\}$, to which all pairs of (non-vacuous) expressions belong. This relation, which we denote $\bullet$, is the black hole of entailment relations, in the sense that (a) it conveys zero information about pairs of expressions which belong to it, and (b) joining a chain of entailment relations will, if it contains any noise and is of sufficient length, lead inescapably to $\bullet$.[17] This tendency of joining to devolve toward less-informative entailment relations places an important limitation on the power of the inference method described in chapter 6.

In an implemented model, the complexity introduced by union relations is easily tamed. As table 5.3 shows, every one of the union relations which results from joining relations in $\mathfrak{B}$ contains #. In practice, any union relation which contains # can safely be approximated by #. After all, # is already the least informative relation in $\mathfrak{B}$—loosely speaking, it indicates ignorance of the relation between two expressions—and further joining will never serve to strengthen it. Our implemented model therefore has no need to represent union relations at all.

[16] That is, the relations in $\mathfrak{B}$ plus 9 union relations. Note that this closure fails to include most of the 120 possible union relations. Perhaps surprisingly, the unions $\bigcup\{\equiv, \sqsubset\}$ and $\bigcup\{\wedge, \mid\}$ mentioned in footnote 15 do not appear.

[17] In fact, computer experiments show that if relations are selected uniformly at random from $\mathfrak{B}$, it requires on average just five joins to reach $\bullet$.

Chapter 6

Compositional entailment

The inventory of basic entailment relations $\mathfrak{B}$ developed in chapter 5 provides a foundation on which we can now build a theory of *compositional entailment* which extends and generalizes past work in *natural logic* (introduced in section 1.5).[1] Our theory is inspired by the monotonicity calculus of Sánchez Valencia, but augments it significantly. Whereas the monotonicity calculus focuses solely on semantic containment relations, our theory adds the ability to explain inferences involving negation, antonymy, and other forms of semantic exclusion. Moreover, it incorporates elements of a model developed by Nairn et al. (2006) to explain inferences involving implicative and factive constructions. By describing all these phenomena (semantic containment, semantic exclusion, implicatives and factives) in a unified framework, the theory can also explain interactions among them.

The cornerstone of our account is the *principle of compositionality*, also known as Frege's Principle, after its best-known exponent. In its original form, this principle states that *the meaning of a compound expression is a function of the meanings of its parts*. In this chapter, we adopt an analogous principle which states that (some of) *the entailments of a compound expression are a function of the entailments of its parts*.[2] Of course, the principle of compositionality has its critics (notably Chomsky (1975)),

[1] The material in this chapter is derived in large part from (MacCartney and Manning 2009).

[2] This relies on extending entailment relations to cover expressions of all semantic types, as described in section 5.4, as opposed to the purely truth-functional conception of entailment standard in formal logic.

and it is not difficult to produce examples which seem to violate compositionality. It is nevertheless clear that the semantics of a vast range of everyday phrases (such as *two nice men who helped me carry groceries yesterday*) are best explained compositionally, and the principle of compositionality has provided the foundation for much of modern semantic analysis, from Montague onwards. Above all, our approach is pragmatic, not dogmatic. We freely concede that our theory of compositional entailment will fail to explain many entailments, even from sentences whose meaning is (arguably) purely compositional. But we hope to demonstrate that such a theory can elegantly explain a rich variety of entailments, and indeed, can do so without resorting to full semantic interpretation.

The theory we develop in this chapter can be summarized as follows. If two linguistic expressions differ by a single *atomic edit* (the deletion, insertion, or substitution of a subexpression), then the entailment relation between them depends on two factors: first, the *lexical entailment relation* generated by the edit; and second, how this lexical entailment relation is affected by semantic composition with the remainder of the expression (the context). In section 6.1, we describe the lexical entailment relations produced by various common kinds of edits. (An important set of special cases is covered in section 6.3.) Then, in section 6.2, we reach the core of the theory: an account of *projectivity*, which explains how entailment relations between simple expressions are projected, via semantic composition, to entailment relations between compound expressions. Together, these sections enable us to determine the *atomic entailment relation* between any two expressions connected by an atomic edit. Next, in section 6.4, we combine this power with the account of joining entailment relations (section 5.6) to describe a general method for establishing the entailment relation between arbitrary pairs of expressions. In section 6.5 we present a number of worked-out examples of the operation of this inference method, and in section 6.6 we consider whether the method can be shown to be inferentially sound.

6.1 Lexical entailment relations

Suppose x is a compound linguistic expression, and let $e(x)$ be the result of applying an *atomic edit* e (the deletion, insertion, or substitution of a subexpression) to x. The entailment relation $\beta(x, e(x))$ which holds between x and $e(x)$ will depend on two factors: (1) the *lexical entailment relation* generated by e, which we label $\beta(e)$, and (2) other properties of the context x in which e is applied (to be discussed in section 6.2). For example, suppose x is *red car*. If e is SUB(*car*, *convertible*), then $\beta(e)$ is $\sqsupset$ (because *convertible* is a hyponym of *car*). On the other hand, if e is DEL(*red*), then $\beta(e)$ is $\sqsubset$ (because *red* is an intersective modifier). Crucially, the value of $\beta(e)$ depends solely on the lexical items involved in e, independent of context.

Our use of the term "lexical" should not be taken to indicate that the edits in question must operate on individual words. On the contrary, it will often be convenient to consider edits involving short phrases, or even edits (especially deletions) of rather long phrases or whole clauses. What matters, for current purposes, is that the arguments to the edit will be treated as lexical units: in determining their semantics and their entailments, we will not consider their compositional structure (if any), but will treat them as semantic atoms.

How are lexical entailment relations determined? Ultimately, this is the province of lexical semantics, which lies outside the scope of this work. However, the answers are fairly intuitive in most cases, and we can make a number of useful observations.

6.1.1 Substitutions of open-class terms

The lexical entailment relation generated by a substitution edit is simply the relation between the substituted terms: $\beta(\text{SUB}(a, b)) = \beta(a, b)$. These terms need not be single words (*bachelor* $\equiv$ *unmarried man*), but they must belong to the same semantic type, or else $\beta(a, b)$ will be undefined.

For open-class terms such as common nouns, adjectives, and verbs, we can often determine the appropriate lexical entailment relation by consulting a lexical resource such as WordNet. Synonyms belong to the $\equiv$ relation (*sofa* $\equiv$ *couch*, *happy* $\equiv$ *glad*, *forbid* $\equiv$ *prohibit*); hyponym-hypernym pairs belong to the $\sqsubset$ relation (*crow* $\sqsubset$ *bird*,

frigid $\sqsubset$ *cold*, *soar* $\sqsubset$ *rise*); and antonyms generally belong to the | relation (*hot* | *cold*, *rise* | *fall*, *advocate* | *opponent*). (Note that most antonym pairs do *not* belong to the $^$ relation, since they typically do not exclude the middle.) Two common nouns which have neither a synonymy nor a hyponymy relation typically describe exclusive categories, and thus belong to the | relation, whether they are coordinate terms (*cat* | *dog*) or unrelated nouns (*battle* | *chalk*). By contrast, two unrelated adjectives usually describe qualities which are not incompatible, and therefore belong to the # relation (*weak* # *temporary*). The appropriate entailment relation between two arbitrary verbs may depend in part on their lexical aspects (*Aktionsarten*), and may also depend on world knowledge not readily available to an automatic system. For example, plausibly *skiing* | *sleeping* (because someone who is skiing is surely not asleep), but *skiing* # *talking* (someone who is skiing may or may not be talking).

Proper nouns, which denote individual entities or events, will stand in the $\equiv$ relation if they denote the same entity (*USA* $\equiv$ *United States*), or the | relation otherwise (*JFK* | *FDR*). A proper noun and a common noun should be assigned to the $\sqsubset$ relation if it can be established that the former is an instance of the latter (*Socrates* $\sqsubset$ *man*). There is an interesting question concerning geographic meronym-holonym pairs: should they be assigned to the $\sqsubset$ relation, as in *Kyoto* $\sqsubset$ *Japan*? While this may seem intuitive, it leads to non-sensical conclusions such as *Kyoto is a beautiful city* $\sqsubset$ *Japan is a beautiful city*; thus it is better to assign such pairs to the | relation. On the other hand, locative phrases formed from meronym-holonym pairs do belong to to the $\sqsubset$ relation: thus *in Kyoto* $\sqsubset$ *in Japan*.

In this work, we have adopted the (perhaps dubious) assumption that tense and aspect matter little in inference. Similarly, we neglect the distinction between singular and plural. Consequently, we routinely assign all edits involving auxiliaries and inflection to the $\equiv$ relation; thus we say that *did sleep* $\equiv$ *has slept* and *is sleeping* $\equiv$ *sleeps*. While a purist might quibble with calling such phrase pairs semantically equivalent, we find that the flexibility afforded by this approximation yields a substantial dividend in the performance of our implemented system, and we are bolstered by the fact that the official definition of the RTE task explicitly specifies that tense be ignored. Similarly, we treat punctuation as semantically vacuous, and thus assign

all edits involving punctuation marks to the $\equiv$ relation.

Very few naturally-occurring pairs of terms belong to the $\smile$ relation. When we are obliged to produce such pairs for the purpose of an illustrative example, we often resort to using a general term and the negation of one of its hyponyms, such as *mammal* $\smile$ *nonhuman* or *metallic* $\smile$ *nonferrous*.

Pairs of terms which cannot reliably be assigned to another entailment relation, according to one of the principles described in this section, should most likely be assigned to the # relation (*hungry* # *hippo*).

Of course, there are many difficult cases, where the most appropriate lexical entailment relation between two terms will depend on subjective judgments about word sense, topical context, and so on. Consider, for example, the pair *system* and *approach*. In some contexts (e.g., computer science research), they may be used as near-synonyms—but near enough to be assigned to the $\equiv$ relation? In other contexts (e.g., air traffic control), they may have quite different meanings.

6.1.2 Substitutions of closed-class terms

Closed-class terms may require special handling. For example, substitutions involving quantifiers generate a rich variety of lexical entailment relations:

$$\begin{aligned}
\textit{all} &\equiv \textit{every} \\
\textit{every} &\sqsubset \textit{some} \\
\textit{some} &\wedge \textit{no} \\
\textit{no} &\mid \textit{every} \\
\textit{four or more} &\sqsubset \textit{two or more} \\
\textit{exactly four} &\mid \textit{exactly two} \\
\textit{at most four} &\smile \textit{at least two} \\
\textit{most} &\# \textit{ten or more}
\end{aligned}$$

Some of these results may be more intuitive when seen in context. For instance, from *some* $\wedge$ *no*, we can establish that *some birds talk* $\wedge$ *no birds talk*, and from *exactly four* $\mid$ *exactly two*, we can establish that *exactly four jurors smoke* $\mid$ *exactly two jurors smoke*. Note also that some of these assertions assume the non-vacuity (section 5.5.1)

of the predicates to which the quantifiers are applied. In particular, *every x y* $\sqsubset$ *some x y* assumes that *x* is a non-empty predicate; otherwise *every x y* could be true while *some x y* is false. Similarly, *no x y* | *every x y* assumes that *x* is non-empty; otherwise *no x y* and *every x y* might be simultaneously true. Finally, note that the interpretation of bare numbers may depend on pragmatic issues: does *four* mean *at least four* or *exactly four*? If it's the former, then *I have four children* $\sqsubset$ *I have two children*; if it's the latter, then *I have four children* | *I have two children.* At stake is the well-known distinction between entailment and conversational implicature (Grice 1975); because these issues have been thoroughly explored elsewhere, we will not pursue them here.

Two pronouns, or a pronoun and a noun, should ideally be assigned to the $\equiv$ relation if it can be determined from context that they refer to the same entity, though this may be difficult for an automatic system to establish reliably.

Prepositions can be somewhat problematic. Because of their protean nature, the appropriate entailment relations among prepositions may depend strongly on context. Some pairs of prepositions are clearly best interpreted as antonyms, and thus assigned to the | relation (*above* | *below*). But many prepositions are used so flexibly in natural language that the pragmatic choice is to assign them to the $\equiv$ relation (*on* [*a plane*] $\equiv$ *in* [*a plane*] $\equiv$ *by* [*plane*]). Then again, in contexts where they clearly refer to spatial relations, the same pairs of prepositions are better assigned to the | relation (*on* [*the box*] | *in* [*the box*] | *by* [*the box*]). This context-dependence poses a significant challenge to automatic prediction of lexical entailment relations.

6.1.3 Generic deletions and insertions

For deletion edits, the default behavior is to generate the $\sqsubset$ relation (thus *red car* $\sqsubset$ *car*). Insertion edits are symmetric: by default, they generate the $\sqsupset$ relation (*sing* $\sqsupset$ *sing off-key*). This heuristic can safely be applied whenever the affected phrase is an intersective modifier, a conjunct, or an independent clause; and it can usefully be applied to phrases much longer than a single word (*car which has been parked outside since last week* $\sqsubset$ *car*). Indeed, this principle underlies most current approaches to the

RTE task, in which the premise p often contains much extraneous content not found in the hypothesis h. Most RTE systems try to determine whether p subsumes h: they penalize new content inserted into h, but do not penalize content deleted from p. The success of this strategy depends on the prevalence of upward-monotone contexts, and thus can easily be derailed by the presence of negation, certain quantifiers, restrictive verbs and adverbs, and other downward-monotone operators.

6.1.4 Special deletions and insertions

While most deletions and insertions adhere to the simple heuristic described in section 6.1.3, some lexical items exhibit special behavior upon deletion or insertion. The most obvious example is negation, which generates the $^\wedge$ relation (*didn't sleep* $^\wedge$ *did sleep*). Implicatives and factives (such as *refuse to* and *admit that*) constitute another important class of exceptions, but we postpone discussion of them until section 6.3. Then there are non-intersective adjectives such as *fake*, *former*, and *alleged*. These have various behavior: deleting *fake* or *former* seems to generate the | relation (*fake diamond* | *diamond*, *former student* | *student*), while deleting *alleged* seems to generate the # relation (*alleged spy* # *spy*). We lack a complete typology of such cases, but consider this an interesting problem for lexical semantics.

6.2 Entailments and semantic composition

How are entailment relations affected by semantic composition? That is, how do the entailment relations between compound expressions depend on the entailment relations between their parts? Suppose we have established the value of $\beta(x, y)$, and let f be an expression which can take x or y as an argument. What is the value of $\beta(f(x), f(y))$, and how does it depend on the properties of f? In other words, how is $\beta(x, y)$ *projected* through f?

6.2.1 Semantic composition in the monotonicity calculus

The monotonicity calculus of Sánchez Valencia provides a partial answer. It explains the impact of semantic composition on entailment relations $\equiv$, $\sqsubset$, $\sqsupset$, and $\#$ by assigning semantic functions to one of three monotonicity classes: UP, DOWN, and NON. If f has monotonicity UP (the default), then the entailment relation between a and b is projected through f without change: $\beta(f(x), f(y)) = \beta(x, y)$. Thus *some parrots talk* $\sqsubset$ *some birds talk*, because *parrot* $\sqsubset$ *bird*, and *some* has monotonicity UP in its first argument. If f has monotonicity DOWN, then $\sqsubset$ and $\sqsupset$ are swapped. Thus *no carp talk* $\sqsupset$ *no fish talk*, because *carp* $\sqsubset$ *fish*, and *no* has monotonicity DOWN in its first argument. Finally, if f has monotonicity NON, then $\sqsubset$ and $\sqsupset$ are projected as $\#$. Thus *most humans talk* $\#$ *most animals talk*, because while *human* $\sqsubset$ *animal*, *most* has monotonicity NON in its first argument.

The monotonicity calculus also provides an algorithm for computing the effect on entailment of multiple levels of semantic composition. Although Sánchez Valencia's presentation of this algorithm uses a complex scheme for annotating nodes in a categorial grammar parse, the central idea can be recast in simple terms: propagate a lexical entailment relation upward through a semantic composition tree, from leaf to root, while respecting the monotonicity properties of each node along the path. Consider the sentence *Nobody can enter without pants*. A plausible semantic composition tree for this sentence could be rendered as (*nobody* (*can* ((*without pants*) *enter*))). Now consider replacing *pants* with *clothes*. We begin with the lexical entailment relation: *pants* $\sqsubset$ *clothes*. The semantic function *without* has monotonicity DOWN, so *without pants* $\sqsupset$ *without clothes*. Continuing up the semantic composition tree, *can* has monotonicity UP, but *nobody* has monotonicity DOWN, so we get another reversal, and find that *nobody can enter without pants* $\sqsubset$ *nobody can enter without clothes*.

6.2.2 Projectivity signatures

While the monotonicity calculus elegantly explains the impact of semantic composition on the containment relations (chiefly, $\sqsubset$ and $\sqsupset$), it lacks any account of the exclusion relations ($^\wedge$ and $|$, and, indirectly, $\smile$). To remedy this lack, we propose to

	projectivity						
connective	≡	⊏	⊐	^	|	⌣	#
negation (*not*)	≡	⊐	⊏	^	⌣	|	#
conjunction (*and*) / intersection	≡	⊏	⊐	|	|	#	#
disjunction (*or*)	≡	⊏	⊐	⌣	#	⌣	#
conditional (*if*) (antecedent)	≡	⊐	⊏	#	#	#	#
conditional (*if*) (consequent)	≡	⊏	⊐	|	|	#	#
biconditional (*if and only if*)	≡	#	#	^	#	#	#

Table 6.1: Projectivity signatures for various constructions which correspond to logical connectives. Certain approximations have been made; see text for details.

generalize the concept of monotonicity to a concept of *projectivity*. We categorize semantic functions into a number of *projectivity signatures*, which can be seen as generalizations of both the three monotonicity classes of Sánchez Valencia and the nine implication signatures of Nairn et al. (see section 6.3). Each projectivity signature is defined by a map $\mathfrak{B} \mapsto \mathfrak{B}$ which specifies how each entailment relation is projected by the function. (Binary functions can have different signatures for each argument.) In principle, there are up to 7^7 possible signatures; in practice, probably no more than a handful are realized by natural language expressions. Though we lack a complete inventory of projectivity signatures, we can describe a few important cases.

6.2.3 Projectivity of logical connectives

We begin by considering natural language expressions and constructions which correspond to logical connectives, such as negations, conjunctions, and conditionals. The projectivity properties of such expressions can be computed algorithmically. Table 6.1 shows the results, but note that certain approximations and simplifications have been made, which will be explained below.

Negation. Like most functions, negation (represented in English by *not*, *n't*, or *never*) projects ≡ and # without change. As a downward monotone function, it swaps ⊏ and ⊐. But we can also establish that it projects ^ without change, and

swaps | and $\smile$. It thus projects every relation as its *dual under negation* (in the sense defined formally in section 5.3.1). For example:

happy	$\equiv$	*glad*	$\Rightarrow$	*not happy*	$\equiv$	*not glad*
kiss	$\sqsubset$	*touch*	$\Rightarrow$	*didn't kiss*	$\sqsupset$	*didn't touch*
human	$\wedge$	*nonhuman*	$\Rightarrow$	*not human*	$\wedge$	*not nonhuman*
French	|	*German*	$\Rightarrow$	*not French*	$\smile$	*not German*
more than 4	$\smile$	*less than 6*	$\Rightarrow$	*not more than 4*	|	*not less than 6*
swimming	#	*hungry*	$\Rightarrow$	*isn't swimming*	#	*isn't hungry*

Conjunction and intersective modification. Conjunction (expressed by *and*) is upward-monotone, but projects both $\wedge$ and | as |, and projects $\smile$ as #. The closely related phenomenon of intersective modification (achieved by adjectives, adverbs, relative clauses, and so on) has the same projectivity, for both modifier and modified. We therefore find:

dinghy	$\sqsubset$	*boat*	$\Rightarrow$	*orange dinghy*	$\sqsubset$	*orange boat*
human	$\wedge$	*nonhuman*	$\Rightarrow$	*living human*	|	*living nonhuman*
French	|	*Spanish*	$\Rightarrow$	*French wine*	|	*Spanish wine*
metallic	$\smile$	*nonferrous*	$\Rightarrow$	*metallic pipe*	#	*nonferrous pipe*

Disjunction. Like conjunction, disjunction is upward-monotone; unlike conjunction, it projects $\wedge$ and $\smile$ as $\smile$, and projects | as #:

waltzed	$\sqsubset$	*danced*	$\Rightarrow$	*waltzed or sang*	$\sqsubset$	*danced or sang*
human	$\wedge$	*nonhuman*	$\Rightarrow$	*human or equine*	$\smile$	*nonhuman or equine*
red	|	*blue*	$\Rightarrow$	*red or yellow*	#	*blue or yellow*

Conditionals. The antecedent of a conditional is downward-monotone, whereas the consequent is upward-monotone:

If he drinks tequila, he feels nauseous $\sqsupset$ *If he drinks liquor, he feels nauseous*
If he drinks tequila, he feels nauseous $\sqsubset$ *If he drinks tequila, he feels sick*

The antecedent of a conditional projects both $\wedge$ and | as #, whereas the consequent projects both $\wedge$ and | as |:

If it's sunny, we surf # *If it's not sunny, we surf*
If it's sunny, we surf # *If it's rainy, we surf*
If it's sunny, we surf | *If it's sunny, we don't surf*
If it's sunny, we surf | *If it's sunny, we ski*

Note that these results do *not* match the projectivity signatures obtained algorithmically for material implication (which include the partial signatures $\{^\wedge{:}\smile, |{:}\smile, \smile{:}\#\}$ for the antecedent, and $\{^\wedge{:}\smile, |{:}\#, \smile{:}\smile\}$ for the consequent). Clearly, material implication is not a good model for the semantics of the conditional statement as typically used in natural language; for a thorough discussion, see Stalnaker (1968; 1992).

Biconditionals. Biconditionals are rarely used in natural language, but where they are, they block projection of all entailment relations except $\equiv$ and $^\wedge$.

Some caveats are in order. While the results shown in table 6.1 are broadly correct, certain approximations have been made (except in the case of negation, for which the projectivity signature shown is exact). The reason is that, for binary functions, the projection of a given entailment relation can depend on the value of the *other* argument to the function. That is, if we are given $\beta(x, y)$, and we are trying to determine its projection $\beta(f(x, z), f(y, z))$, the answer can depend not only on the properties of f, but also on the properties of z. In particular, if z has some "special" relation to x or y (say, $\equiv$, $\sqsubset$, or $^\wedge$), a different projection may result. Therefore, the results shown in table 6.1 reflect the assumption that z is as "generic" as possible: specifically, that z stands in the (minimally informative) # relation to both x and y. Our motivation for this assumption is closely related to our motivation for ignoring vacuous expressions (those having empty or universal denotations): in natural language one rarely combines two expressions which already have some "special" relation, since the result is typically either redundant (*French European*, *sofa or couch*) or nonsensical (*human nonhuman*, *hot cold food*). This assumption causes the results shown in table 6.1 to be imprecise in the following ways:

- Certain "special" values for z can cause conjunction, disjunction, and the conditional (in both the antecedent and the consequent) to project $\sqsubset$ or $\sqsupset$ as $\equiv$. As an example, consider the projection of $\sqsubset$ by conjunction (*and*). Suppose x is *French*, y is *European*, and z is *Parisian*. Since the conjunctions *French and Parisian* and *European and Parisian* are both equivalent to *Parisian*, the entailment relation between them is $\equiv$. In table 6.1, we neglect such possibilities, following the argument that such redundant expressions rarely occur in natural language.

- Certain "special" values for z can result in *degenerate* entailment relations (in the sense defined in section 5.3.1) being projected. As an example, consider the projection of $|$ by conjunction (*and*). Suppose x is *male*, y is *female*, and z is *asexual*. Since the conjunctions *male and asexual* and *female and asexual* are both empty, the entailment relation between them is R_{1000} (defined in section 5.3.1). In table 6.1, we neglect such possibilities, in keeping with our assumption that semantically vacuous expressions will rarely be encountered in natural language.

- Except in the case of negation, the projections shown as # are in fact larger, less informative union relations (defined in section 5.6.3): either $\bigcup\{\equiv, \sqsubset, \sqsupset, |, \#\}$, $\bigcup\{\equiv, \sqsubset, \sqsupset, \smile, \#\}$, or $\bigcup\{\sqsubset, \sqsupset, |, \smile, \#\}$. As in section 5.6.3, we argue that these approximations make little practical difference, since the true projections contain #, and # is already the least informative entailment relation.

- Some of the relations shown as projections in table 6.1 are not strictly identical with the true projections, but rather are least upper bounds (in the space of relations in $\mathfrak{B}$ and unions thereof) on the true projections. In other words, the relations shown may contain certain pairs of expressions which cannot occur as a result of projection. The discrepancies in question are small, and are of no practical consequence. (Note that a similar issue was encountered in connection with computing the joins of relations in $\mathfrak{B}$ in section 5.6.2.)

	projectivity for 1st argument							projectivity for 2nd argument						
quantifier	≡	⊏	⊐	^	|	⌣	#	≡	⊏	⊐	^	|	⌣	#
some	≡	⊏	⊐	⌣†	#	⌣†	#	≡	⊏	⊐	⌣†	#	⌣†	#
no	≡	⊐	⊏	|†	#	|†	#	≡	⊐	⊏	|†	#	|†	#
every	≡	⊐	⊏	|‡	#	|‡	#	≡	⊏	⊐	|†	|†	#	#
not every	≡	⊏	⊐	⌣‡	#	⌣‡	#	≡	⊐	⊏	⌣†	⌣†	#	#
at least two	≡	⊏	⊐	#	#	#	#	≡	⊏	⊐	#	#	#	#
most	≡	#	#	#	#	#	#	≡	⊏	⊐	|	|	#	#
exactly one	≡	#	#	#	#	#	#	≡	#	#	#	#	#	#
all but one	≡	#	#	#	#	#	#	≡	#	#	#	#	#	#

Table 6.2: Projectivity signatures for various binary generalized quantifiers for each argument position. "1st argument" refers to the restrictor NP; "2nd argument" refers to the body VP. Results marked with † or ‡ depend on the assumption of non-vacuity; see text.

6.2.4 Projectivity of quantifiers

While semanticists are well acquainted with the monotonicity properties of common quantifiers, which describe how they project the containment relations (≡, ⊏, and ⊐), how they project the exclusion relations (^, |, and, indirectly, ⌣) may be less familiar. Table 6.2 summarizes the projectivity signatures of a number of binary generalized quantifiers for each argument position. This table has several noteworthy aspects.

First, all quantifiers (like most other semantic functions) project ≡ and # without change. Second, the table confirms well-known monotonicity properties of quantifiers: *no* is downward-monotone in both arguments; *every* is downward-monotone in its first argument; *most* is non-monotone in its first argument; and so on. Third, because *no* is the negation of *some*, its projectivity signature can be found by projecting the signature of *some* through the signature of *not*. Likewise for *not every* and *every*.

Note that relation | is frequently "blocked" (i.e., projected as #). Of the quantifiers shown in table 6.2, this holds for all in the first argument position, and for most in the second argument position. Thus *no fish talk # no birds talk* and *someone was early # someone was late*. Two significant exceptions are the quantifiers

every and *most* in their second arguments, where | is projected without change: thus *everyone was early* | *everyone was late* and *most people were early* | *most people were late*. (Observe the similarity of the projectivity signatures of *every* and *most* in their second arguments to that of intersective modification.)

Finally, a caveat: some of the results shown in table 6.2 depend on assuming the non-vacuity of the other argument to the quantifier: those marked with † assume it to be non-empty, while those marked with ‡ assume it to be non-universal. Without these assumptions, # is projected.

6.2.5 Projectivity of verbs

Verbs (and verb-like constructions) exhibit diverse behavior with respect to projectivity. Most verbs are upward-monotone (though not all—see section 6.3), and many verbs project ^, |, and ⌣ as #. For example:

$$\begin{aligned} \textit{humans} \mathbin{\wedge} \textit{nonhumans} &\Rightarrow \textit{eats humans} \mathbin{\#} \textit{eats nonhumans} \\ \textit{cats} \mid \textit{dogs} &\Rightarrow \textit{eats cats} \mathbin{\#} \textit{eats dogs} \\ \textit{mammals} \mathbin{\smile} \textit{nonhumans} &\Rightarrow \textit{eats mammals} \mathbin{\#} \textit{eats nonhumans} \end{aligned}$$

However, exceptions abound. Consider, for example, verbs (and verb-like constructions) which encode functional relations, such as *is married to* or *is the capital of*.[3] The hallmark of such expressions is that they impose a sort of exclusivity constraint on their objects: a given subject can have the specified relation to at most one object (though subject and object may be described in various ways, and with various degrees of specificity).[4] These constructions seem to exhibit projectivity behavior similar to that of intersective modifiers, projecting ^ and | as |, and ⌣ as #. So, for example, we find that:

$$\begin{aligned} \textit{is married to a German} &\mid \textit{is married to a non-German} \\ \textit{is married to a German} &\mid \textit{is married to an Italian} \\ \textit{is married to a European} &\mathbin{\#} \textit{is married to a non-German} \end{aligned}$$

[3]Ideal examples of this phenomenon are hard to find. Most candidates admit some exceptions: Solomon had 700 wives, and London is the capital of both England and the UK.

[4]The AuContraire system (Ritter et al. 2008) includes an intriguing approach to identifying such *functional phrases* automatically.

Categorizing verbs according to their projectivity is an interesting problem for lexical semantics, which may involve codifying some amount of world knowledge.

6.3 Implicatives and factives

6.3.1 Implication signatures

Nairn et al. (2006) offer an elegant account of inferences involving implicatives and factives[5] such as *manage to*, *refuse to*, and *admit that*. Their model classifies such operators into nine *implication signatures*, according to their implications regarding their complements—positive (+), negative (–), or null (∘)—in both positive and negative contexts. Thus *manage to* has implication signature +/–, because it carries a positive implication in a positive context (*managed to escape* implies *escaped*), and a negative implication in a negative context (*didn't manage to escape* implies *didn't escape*). In contrast, *refuse to* has implication signature –/∘, because it carries a negative implication in a positive context (*refused to dance* implies *didn't dance*), and no implication in a negative context (*didn't refuse to dance* implies neither *danced* nor *didn't dance*). Table 6.3 shows canonical examples of each of the nine possible implication signatures.

An important difference between factive and implicative operators is that the factives (but not the implicatives) carry the same implication in both positive and negative contexts. This difference is related to the nature of the primary implication carried by the operator. A commonly held view (Karttunen 1971) is that factive constructions *presuppose*, rather than *entail*, the truth (or falsity, in the case of counterfactives) of their complements: thus, *he admitted that he knew* presupposes that *he knew*. One of the hallmarks of presuppositions (as opposed to entailments) is that they are not affected by negation, which is why the factives carry the same implication in negative contexts as in positive contexts. The implicatives, on the other hand, are commonly thought to entail, rather than presuppose, the truth (or falsity)

[5]We use "factives" as an umbrella term embracing counterfactives (e.g., *pretend that*) and non-factives (e.g., *believe that*) along with factives proper (e.g., *admit that*).

	signature	example
implicatives	$+/-$	*manage to*
	$+/\circ$	*force to*
	$\circ/-$	*permit to*
	$-/+$	*fail to*
	$-/\circ$	*refuse to*
	$\circ/+$	*hesitate to*
factives	$+/+$	*admit that*
	$-/-$	*pretend that*
	$\circ/\circ$	*believe that*

Table 6.3: The nine implication signatures of Nairn et al.

of their complements: thus *he managed to escape* entails that *he escaped*. Of course, implicatives may carry presuppositions as well (*he managed to escape* might be argued to presuppose that *he tried to escape*, or perhaps that *it was hard to escape*), but such implications are secondary to the relation between the implicative and its complement.

6.3.2 Deletions and insertions of implicatives

The model developed by Nairn et al. explains inferences involving implicatives by means of an *implication projection algorithm*, which bears some resemblance to the marking algorithm of Sánchez Valencia's monotonicity calculus, in that it involves propagating information about the effects of semantic functions on inferability through a semantic composition tree.

We can take a big step toward unifying the two accounts within our framework by specifying, for each implication signature, the lexical entailment relation generated when an operator of that signature is deleted from (or inserted into) a compound expression. For example, deleting an operator with implication signature $+/\circ$ generates the $\sqsubset$ relation (*he was forced to sell* $\sqsubset$ *he sold*), whereas deleting an operator with implication signature $-/\circ$ generates the $|$ relation (*he refused to fight* $|$ *he fought*).

signature	$\beta(\textsc{del}(\cdot))$	$\beta(\textsc{ins}(\cdot))$	example
$+/-$	$\equiv$	$\equiv$	*he managed to escape* $\equiv$ *he escaped*
$+/\circ$	$\sqsubset$	$\sqsupset$	*he was forced to sell* $\sqsubset$ *he sold*
$\circ/-$	$\sqsupset$	$\sqsubset$	*he was permitted to live* $\sqsupset$ *he lived*
$-/+$	$\wedge$	$\wedge$	*he failed to pay* $\wedge$ *he paid*
$-/\circ$	$\mid$	$\mid$	*he refused to fight* $\mid$ *he fought*
$\circ/+$	$\smile$	$\smile$	*he hesitated to ask* $\smile$ *he asked*
$\circ/\circ$	$\#$	$\#$	*he believed he had won* $\#$ *he had won*

Table 6.4: The lexical entailment relations generated by deletions and insertions of implicatives (and nonfactives), by implication signature.

Table 6.4 depicts the lexical entailment relations generated by deletions and insertions of implicative operators according to their implication signatures. This table invites several observations. First, as the examples make clear, there is room for variation regarding passivization and morphology. An implemented model must tolerate such diversity. Second, the entailment relations generated by deletion and insertion are converses of each other, as we'd expect. Third, we get the right predictions concerning the implications of implicatives occurring in negative polarity contexts. Indeed, some of the examples may seem more intuitive when one considers their negations. For example, deleting an operator with implication signature $\circ/-$ generates lexical entailment relation $\sqsupset$, but as we saw in section 6.2, under negation $\sqsupset$ is projected as $\sqsubset$ (*he wasn't permitted to live* $\sqsubset$ *he didn't live*). Likewise, deleting an operator with implication signature $\circ/+$ generates lexical entailment relation $\smile$; under negation, this is projected as $\mid$ (*he didn't hesitate to ask* $\mid$ *he didn't ask*).

6.3.3 Deletions and insertions of factives

Note that table 6.4 includes neither the factives proper (implication signature $+/+$) nor the counterfactives (implication signature $-/-$). It is tempting to try to handle the factives as we did the implicatives. For example, deleting implication signature $+/+$ seems to generate the lexical entailment relation $\sqsubset$, since *he admitted that he*

knew implies *he knew*, but not vice-versa. Likewise, deleting implication signature –/– seems to generate the lexical entailment relation |, since *he pretended he was sick* excludes *he was sick*, but is not equivalent to its negation.

However, following this path does not lead to the right predictions in negative polarity contexts. If deletions of signature +/+ yield $\sqsubset$, then since negation projects $\sqsubset$ as $\sqsupset$, we would expect to find that *he didn't admit that he knew* $\sqsupset$ *he didn't know*. But this is clearly incorrect. In the first place, *he didn't know* does not imply *he didn't admit that he knew*. Moreover, if $x \sqsupset y$, then the truth of x must be compatible with the truth of y, but one cannot consistently assert both *he didn't admit that he knew* and *he didn't know*, since the former presupposes the negation of the latter. Similarly, if deletions of signature –/– yield |, then since negation projects | as $\smile$, we would expect to find that *he didn't pretend he was sick* $\smile$ *he wasn't sick*. But this too is incorrect. If $x \smile y$, then the truth of x must be compatible with the falsity of y, but one cannot consistently assert both *he didn't pretend he was sick* and *he was sick*, since the former presupposes the negation of the latter.

What has gone wrong? Recall from section 6.3.1 that the implication between a factive and its complement is not an entailment, but a presupposition. The problem arises principally because, as is well known, the projection behavior of presuppositions differs from that of entailments (van der Sandt 1992). Consequently, the model of compositional entailment developed in this chapter does not suffice to explain inferences involving presupposition.[6] While the model could perhaps be elaborated to handle presupposition, we have chosen not to pursue this path.

The problems described here do not affect the nonfactives (implication signature ∘/∘), since the nonfactives do not carry any presupposition (or, indeed, any entailment) regarding their complements. Thus, we can accurately say that deleting a nonfactive generates lexical entailment relation # (*he believed he had won # he had won*). And, since negation projects # as #, we obtain the correct predictions in negative contexts (*he didn't believe he had won # he hadn't won*).

[6]Nevertheless, the NatLog system described in chapter 7 does handle factives (proper) and counterfactives within this framework, assigning lexical entailment relations $\sqsubset$ and | to deletions of implication signatures +/+ and –/– (respectively). Indeed, NatLog benefits from doing so, since in practice, this leads to correct predictions more often than not.

			projectivity						
signature	example	monotonicity	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	#
$+/-$	*manage to*	UP	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\mid$	$\smile$	#
$+/\circ$	*force to*	UP	$\equiv$	$\sqsubset$	$\sqsupset$	$\mid$	$\mid$	#	#
$\circ/-$	*permit to*	UP	$\equiv$	$\sqsubset$	$\sqsupset$	$\smile$	#	$\smile$	#
$-/+$	*fail to*	DOWN	$\equiv$	$\sqsupset$	$\sqsubset$	$\wedge$	$\smile$	$\mid$	#
$-/\circ$	*refuse to*	DOWN	$\equiv$	$\sqsupset$	$\sqsubset$	$\mid$	#	$\mid$	#
$\circ/+$	*hesitate to*	DOWN	$\equiv$	$\sqsupset$	$\sqsubset$	$\smile$	$\smile$	#	#
$+/+$	*admit that*	UP	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	$\wedge$	#	#
$-/-$	*pretend that*	UP	$\equiv$	$\sqsubset$	$\sqsupset$	$\wedge$	#	$\wedge$	#
$\circ/\circ$	*believe that*	NON	#	#	#	#	#	#	#

Table 6.5: The monotonicity and projectivity properties of implicatives and factives, by implication signature. Some results may depend on whether one assumes a *de dicto* or *de re* reading; see text.

6.3.4 Projectivity signatures of implicatives

We can further cement implicatives and factives within our model by specifying the projectivity properties of each implication signature. For an implicative or factive operator i having implication signature I, the projection of entailment relation R can be computed automatically, by exhaustively considering the four possible combinations of truth or falsity for $i(x)$ and $i(y)$, where $\langle x, y\rangle \in R$, given the implication relations between $i(x)$ and x, and between $i(y)$ and y, as specified by I; and the constraints on combinations of truth or falsity for x and y, as specified by R. However, a more detailed description of the algorithm would be more tedious than enlightening.

Table 6.5 summarizes the results. It may be helpful to illustrate some of the findings with concrete examples. Implicatives having signature $+/\circ$ are upward-monotone (*forced to tango* $\sqsubset$ *forced to dance*); and project both $\wedge$ and $\mid$ as $\mid$ (*forced to stay* $\mid$ *forced to go* and *forced to tango* $\mid$ *forced to waltz*).[7] By contrast, implicatives

[7]It might be objected that *forced to tango* does not exclude *forced to waltz*, since someone could be forced to perform both dances during the course of an evening. However, these expressions cannot describe the same event: a single forcing event cannot be simultaneously a forcing-to-tango event and a forcing-to-waltz event.

having signature $-/\circ$ are downward-monotone (*refused to tango* $\sqsupset$ *refused to dance*); they project $\wedge$ as $|$ (*refused to stay* $|$ *refused to go*) but project $|$ as $\#$ (*refused to tango* $\#$ *refused to waltz*).

Note that some of the results in table 6.5 may depend on whether one assumes a *de dicto* or *de re* reading, and that the *de dicto* reading may be more or less available, depending on the specific implicative or factive operator (and perhaps on context). For the two-way implicatives (signatures $+/-$ and $-/+$), a *de dicto* reading is arguably completely unavailable. For the remaining implicatives, the *de dicto* reading seems to be quite marked. (One could perhaps accept, "When he refused to tango, he wasn't refusing to dance—he thought the tango was a sex position.") For certain factives (e.g., *know*) and counterfactives, the *de dicto* reading may be more plausible, and for many nonfactives (e.g., *believe* or *want*), the *de dicto* reading seems to dominate. In general, a *de dicto* reading has the effect of "blocking" projection, that is, projecting every entailment relation as $\#$. This is reflected in the last row of table 6.5, which describes the monotonicity class for implication signature $\circ/\circ$ as NON, and assigns $\#$ to all its projections.

6.4 Putting it all together

We now have the building blocks of a general method to establish the entailment relation between a premise p and a hypothesis h. The steps are as follows:

1. Find a sequence of atomic edits $\langle e_1, \ldots, e_n \rangle$ which transforms p into h: thus $h = (e_n \circ \ldots \circ e_1)(p)$. For convenience, let us define $x_0 = p$, $x_n = h$, and $x_i = e_i(x_{i-1})$ for $i \in [1, n]$.
2. For each atomic edit e_i:
   (a) Determine the lexical entailment relation $\beta(e_i)$ generated by edit e_i, as described in section 6.1.
   (b) Project $\beta(e_i)$ upward through the semantic composition tree of expression x_{i-1} to find the entailment relation $\beta(x_{i-1}, x_i)$ which holds between expressions x_{i-1} and x_i, as described in section 6.2. We call this the *atomic*

entailment relation for edit e_i, and sometimes refer to it using the alternate notation $\beta(x_{i-1}, e_i)$.

3. Join atomic entailment relations across the sequence of edits, as in section 5.6: $\beta(p, h) = \beta(x_0, x_n) = \beta(x_0, e_1) \bowtie \ldots \bowtie \beta(x_{i-1}, e_i) \bowtie \ldots \bowtie \beta(x_{n-1}, e_n)$

While this inference method effectively explains many common patterns of inference (see section 6.5 for examples), it faces several important limitations:

- First, it depends on finding an appropriate edit sequence connecting p and h. By itself, the theory presented in this chapter offers no insight as to how such a sequence should be established, or when one candidate edit sequence should be preferred to another. (This question has been addressed separately in chapter 3.) The method does assume that none of the edits span multiple monotonicity/projectivity domains, and can break down if this assumption is not met. Also, as the examples in section 6.5 will illustrate, the ordering of edits can (though usually does not) affect the outcome of this inference method.

- Second, the usefulness of the result is sometimes limited by the tendency of the join operation toward less informative entailment relations, as described in section 5.6. Joining atomic entailment relations across a sequence of edits may yield a union of relations in $\mathfrak{B}$, rather than an individual relation in $\mathfrak{B}$; in the worst case, it may yield the total relation $\bullet$, which is completely uninformative. (See section 6.5.5 for an example.)

- Third, the inference method provides no general mechanism for combining information from multiple premises. It thus fails to capture many inference rules of classical logic, including modus ponens, modus tollens, disjunction elimination, and so on. (However, some inferences can be enabled by treating auxiliary premises as encoding lexical entailment relations. For example, if we encode *All men are mortal* as the lexical entailment *men* $\sqsubset$ *mortal*, then we can enable the classic syllogism *Socrates is a man* $\sqsubset$ *Socrates is mortal*.)

Because of these limitations, the inference method we describe here has less deductive power than first-order logic. Indeed, it fails to sanction some fairly simple

inferences, including de Morgan's laws for quantifiers (see section 6.5.4 for details). Yet despite its limitations, this inference method neatly explains a broad range of inferences, including not only those which involve semantic containment (which are also explained by the monotonicity calculus) but also those which involve semantic exclusion and implicativity (which are not). The next section presents a variety of examples.

6.5 Examples

In the following pages, we illustrate the operation of the inference method described in section 6.4, and elucidate some related issues, by working through a series of concrete examples of its application.

6.5.1 An example involving exclusion

While the monotonicity calculus notably fails to explain even the simplest inferences involving semantic exclusion, such examples are easily accommodated in our framework. For instance:

p: *Stimpy is a cat.*

h: *Stimpy is not a poodle.*

Clearly, this is a valid inference. To establish this using our inference method, we must begin by selecting a sequence of atomic edits which transforms the premise p into the hypothesis h. While there are several possibilities, one obvious choice is first to replace *cat* with *dog*, then to insert *not*, and finally to replace *dog* with *poodle*. An analysis of this edit sequence is shown in table 6.6. In this representation (of which we will see several more examples in the following pages), we show three entailment relations associated with each edit e_i, namely:

- $\beta(e_i)$, the lexical entailment relation generated by e_i,
- $\beta(x_{i-1}, e_i)$, the atomic entailment relation which holds across e_i, and

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Stimpy is a cat			
1	SUB(*cat*, *dog*)		|	|	|
		Stimpy is a dog			
2	INS(*not*)		^	^	$\sqsubset$
		Stimpy is not a dog			
3	SUB(*dog*, *poodle*)		$\sqsupset$	$\sqsubset$	$\sqsubset$
		Stimpy is not a poodle			

Table 6.6: Analysis of an inference involving semantic exclusion.

- $\beta(x_0, x_i)$, the cumulative join of all atomic entailment relations up through e_i. This can be calculated in the table as $\beta(x_0, x_{i-1}) \bowtie \beta(x_{i-1}, e_i)$.

We also show (in gray) the intermediate forms through which p progresses as it is transformed into h by the sequence of edits.

The analysis proceeds as follows. First, replacing *cat* with its coordinate term *dog* generates the | relation. Next, inserting *not* generates the ^ relation, and | joined with ^ yields $\sqsubset$. Finally, replacing *dog* with its hyponym *poodle* generates the $\sqsupset$ relation. Because of the downward-monotone context created by *not*, this is projected as $\sqsubset$, and $\sqsubset$ joined with $\sqsubset$ yields $\sqsubset$. Therefore, p entails h.

6.5.2 Examples involving implicatives

The inference method described also explains many examples involving implicatives, including this one:

p: *The doctor didn't hesitate to recommend Prozac.*

h: *The doctor recommended medication.*

This is a valid inference. Premise p can be transformed into hypothesis h by a sequence of three edits: the deletion of the implicative *hesitate to*, the deletion of the negation, and the substitution of *Prozac* with its hypernym *medication*. (Here and in subsequent examples, we neglect edits involving auxiliaries and morphology, which simply yield the $\equiv$ relation.) Table 6.7 shows one possible ordering for these three edits; in section 6.5.3 we use the same example to explore the impact of using different edit orderings.

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		The doctor didn't hesitate to recommend Prozac			
1	DEL(*hesitate to*)		$\smile$	|	|
		The doctor didn't recommend Prozac			
2	DEL(*didn't*)		$^$	$^$	$\sqsubset$
		The doctor recommended Prozac			
3	SUB(*Prozac*, *medication*)		$\sqsubset$	$\sqsubset$	$\sqsubset$
		The doctor recommended medication			

Table 6.7: Analysis of an inference involving an implicative.

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		We were not permitted to smoke			
1	DEL(*permitted to*)		$\sqsupset$	$\sqsubset$	$\sqsubset$
		We did not smoke			
2	DEL(*not*)		$^$	$^$	|
		We smoked			
3	INS(*Cuban cigars*)		$\sqsupset$	$\sqsupset$	|
		We smoked Cuban cigars			

Table 6.8: Analysis of another inference involving an implicative.

We can analyze these three edits as follows. First, the deletion of the implicative *hesitate to* generates lexical entailment relation $\smile$, according to its implication signature (namely $\circ/+$); but because of the downward-monotone context created by *didn't*, this is projected as |. Next, the deletion of *didn't* generates lexical entailment relation $^$, in an upward-monotone context, and | joined with $^$ yields $\sqsubset$. Finally, the substitution of *Prozac* with its hypernym *medication* generates lexical entailment relation $\sqsubset$, in an upward-monotone context, and $\sqsubset$ joined with $\sqsubset$ yields $\sqsubset$. Therefore, p entails h.

As another example, consider:

p: *We were not permitted to smoke.*

h: *We smoked Cuban cigars.*

This is *not* a valid inference: in fact, p and h are incompatible. But again, p can be transformed into h by a sequence of three edits (neglecting auxiliaries and morphology, as before), which we analyze as follows. First, deleting the implicative

i	e_i	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
1	DEL(*hesitate to*)	$\smile$	|	|
2	DEL(*didn't*)	^	^	$\sqsubset$
3	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsubset$	$\sqsubset$
1	DEL(*hesitate to*)	$\smile$	|	|
2	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsupset$	|
3	DEL(*didn't*)	^	^	$\sqsubset$
1	DEL(*didn't*)	^	^	^
2	DEL(*hesitate to*)	$\smile$	$\smile$	$\sqsubset$
3	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsubset$	$\sqsubset$
1	DEL(*didn't*)	^	^	^
2	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsupset$	|
3	DEL(*hesitate to*)	$\smile$	$\smile$	$\sqsubset$
1	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsubset$	$\sqsubset$
2	DEL(*hesitate to*)	$\smile$	|	|
3	DEL(*didn't*)	^	^	$\sqsubset$
1	SUB(*Prozac*, *medication*)	$\sqsubset$	$\sqsubset$	$\sqsubset$
2	DEL(*didn't*)	^	^	|
3	DEL(*hesitate to*)	$\smile$	$\smile$	$\sqsubset$

Table 6.9: Six analyses of the same inference, using different edit orders.

permitted to generates lexical entailment relation $\sqsupset$, according to its implication signature (namely $\circ/-$), but because of the downward-monotone context created by *not*, this is projected as $\sqsubset$. Next, deleting *not* generates lexical entailment relation ^, in an upward-monotone context, and $\sqsubset$ joined with ^ yields |. Finally, inserting *Cuban cigars* restricts the meaning of *smoked*, generating lexical entailment relation $\sqsupset$, in an upward-monotone context, and | joined with $\sqsupset$ yields |. Therefore, h contradicts p.

6.5.3 Different edit orders

Since the inference method described in section 6.4 requires selecting an ordered sequence of edits, but does not establish any criteria for preferring one edit order to another, a critical question is: does edit order make any difference to the outcome, and if so, how? As we'll see in section 6.5.5, different edit orders can indeed yield different outcomes: some edit orders can lead to an entailment relation which is less informative than (though still compatible with) the desired result. However, we very commonly find in real-world inference problems that the result obtained from our inference method is insensitive to the order in which the edits are applied. An illustration is shown in table 6.9. Here, we consider all six possible orderings of the edits involved in the Prozac example from section 6.5.2. In each case, the lexical entailment relations generated by each edit are the same, but how they're projected into atomic entailment relations can depend on the relative ordering of deletions of downward monotone operators. Likewise, the intermediate results of joining can vary, but in each case, the final result is the same.

Consider, for example, the second-to-last block in table 6.9. We begin by replacing *Prozac* with its hypernym *medication*. This generates lexical entailment relation $\sqsubset$, which is then projected through the downward-monotone implicative *hesitate to* as $\sqsupset$, which is in turn projected through negation as $\sqsubset$. Next, the deletion of *hesitate to* generates lexical entailment relation $\smile$, which is projected through negation as $|$, and $\sqsubset$ joined with $|$ yields $|$. Finally, the deletion of negation generates lexical entailment relation $\wedge$, in an upward-monotone context, and $|$ joined with $\wedge$ yields $\sqsubset$: the same result as before.

6.5.4 Inability to handle de Morgan's laws for quantifiers

As noted earlier (section 6.4), the inference method we describe has less deductive power than first-order logic, and consequently fails to explain some fairly simple inferences which are staples of introductory logic courses. Consider, for example, de

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Not all birds fly			
1	DEL(*not*)		$^$	$^$	$^$
		All birds fly			
2	SUB(*all*, *some*)		$\sqsubset$	$\sqsubset$	$\smile$
		Some birds fly			
3	INS(*not*)		$^$	$\smile$	$\equiv\sqsubset\sqsupset\smile\#$
		Some birds do not fly			

Table 6.10: Analysis of an example of de Morgan's laws for quantifiers, first try.

Morgan's laws for quantifiers:

$$\neg(\forall x\ P(x)) \ \Leftrightarrow\ \exists x\ (\neg P(x)) \tag{6.1}$$

$$\neg(\exists x\ P(x)) \ \Leftrightarrow\ \forall x\ (\neg P(x)) \tag{6.2}$$

The following example provides a natural-language instantiation of this pattern of inference:

p: *Not all birds fly.*

h: *Some birds do not fly.*

This inference is trivially valid: the entailment relation between p and h is $\equiv$. However, there does not appear to be any way to demonstrate this using the inference method we describe. Transforming p into h will require three atomic edits: (a) the deletion of the sentential (wide-scope) negation, (b) the substitution of *all* with *some*, and (c) the insertion of the predicate (narrow-scope) negation. While these three edits could be applied in any of $3! = 6$ orderings, it will turn out that every ordering leads to the same (unsatisfactory) result.

To begin with, consider the ordering shown in table 6.10. First, the deletion of the sentential negation generates the $^$ relation. Next, the quantifier substitution yields the $\sqsubset$ relation, and $^$ joined with $\sqsubset$ yields $\smile$. Finally, the insertion of the predicate negation yields the $^$ relation, but because the insertion occurs in the second argument (the body) of the existential quantifier *some*, $^$ is projected as $\smile$, and $\smile$ joined with $\smile$ yields, as a final result, the union relation $\bigcup\{\equiv, \sqsubset, \sqsupset, \smile, \#\}$. This relation might be dubbed the NON-EXCLUSION relation, since it omits only the exclusion relations $^$

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Not all birds fly			
1	DEL(*not*)		^	^	^
		All birds fly			
2	INS(*not*)		^	|	⊐
		All birds do not fly			
3	SUB(*all*, *some*)		⊏	⊏	≡⊏⊐⌣#
		Some birds do not fly			

Table 6.11: Analysis of an example of de Morgan's laws for quantifiers, second try.

and |. Because this result contains the desired result (namely ≡), it is not incorrect, but it is far less informative than we would hope.

Different orderings of the three required edits lead to the same outcome. For example, if we insert the predicate negation before we perform the quantifier substitution, then we obtain the analysis shown in table 6.11. As before, the deletion of the sentential negation generates the ^ relation. But this time, the insertion of the predicate negation occurs in the second argument (the body) of the universal quantifier *all*, so that the lexical entailment relation ^ is projected as | (see section 6.2.4), and ^ joined with | yields ⊐. Finally, the quantifier substitution generates ⊏, and ⊐ joined with ⊏ again yields the NON-EXCLUSION relation: not incorrect, but under-specific.

While we won't bother to work through the details here, it can be shown that each of the six possible orderings of the three required edits leads to the same result, albeit via different intermediate steps. The problem could perhaps be solved by allowing more complex edit operations to count as atomic, and by specifying the appropriate lexical entailment relations generated by these richer edits. In our example, it would suffice to treat the quantifier substitution and the insertion of predicate negation as a single atomic edit which generates the ^ relation. However, such an ad-hoc solution has little to recommend it.

6.5.5 A more complex example

Let's now look at a more complex example that demonstrates the interaction of a number of aspects of the model we've presented. The inference is:

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Jimmy Dean refused to move without blue jeans			
1	SUB(*Jimmy Dean*, *James Dean*)		≡	≡	≡
		James Dean refused to move without blue jeans			
2	DEL(*refuse to*)		|	|	|
		James Dean moved without blue jeans			
3	INS(*did*)		≡	≡	|
		James Dean did move without blue jeans			
4	INS(*n't*)		^	^	⊏
		James Dean didn't move without blue jeans			
5	SUB(*move*, *dance*)		⊐	⊏	⊏
		James Dean didn't dance without blue jeans			
6	DEL(*blue*)		⊏	⊏	⊏
		James Dean didn't dance without jeans			
7	SUB(*jeans*, *pants*)		⊏	⊏	⊏
		James Dean didn't dance without pants			

Table 6.12: Analysis of a more complex inference, first try.

p: *Jimmy Dean refused to move without blue jeans.*

h: *James Dean didn't dance without pants.*

Of course, the example is quite contrived, but it has the advantage that it compactly exhibits several phenomena of interest: semantic containment (between *move* and *dance*, and between *pants* and *jeans*); semantic exclusion (in the form of negation); an implicative (namely, *refuse to*); and nested inversions of monotonicity (created by *refuse to* and *without*).

In this example, the premise p can be transformed into the hypothesis h by a sequence of seven edits, as shown in table 6.12. (This time we include even "light" edits, for completeness.) We analyze these edits as follows. The first edit simply substitutes one variant of a name for another; since both substituends denote the same entity, the edit generates the ≡ relation. The second edit deletes an implicative (*refuse to*) with implication signature −/∘. As described in section 6.3.2, deletions of this signature generate the | relation, and ≡ joined with | yields |. The third edit inserts an auxiliary verb (*did*); since auxiliaries are more or less semantically vacuous, this generates the ≡ relation, and | joined with ≡ yields | again. The fourth edit inserts a negation, generating the ^ relation. Here we encounter the first interesting

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Jimmy Dean refused to move without blue jeans			
1	INS(*did*)	*Jimmy Dean did refuse to move without blue jeans*	$\equiv$	$\equiv$	$\equiv$
2	INS(*n't*)	*Jimmy Dean didn't refuse to move without blue jeans*	$\wedge$	$\wedge$	$\wedge$
3	DEL(*blue*)	*Jimmy Dean didn't refuse to move without jeans*	$\sqsubset$	$\sqsupset$	$\mid$
4	SUB(*jeans*, *pants*)	*Jimmy Dean didn't refuse to move without pants*	$\sqsubset$	$\sqsupset$	$\mid$
5	SUB(*move*, *dance*)	*Jimmy Dean didn't refuse to dance without pants*	$\sqsupset$	$\sqsupset$	$\mid$
6	DEL(*refuse to*)	*Jimmy Dean didn't dance without pants*	$\mid$	$\smile$	$\sqsubset$
7	SUB(*Jimmy Dean*, *James Dean*)	*James Dean didn't dance without pants*	$\equiv$	$\equiv$	$\sqsubset$

Table 6.13: Analysis of a more complex inference, second try.

join: as explained in section 5.6, $\mid$ joined with $\wedge$ yields $\sqsubset$. The fifth edit substitutes *move* with its hyponym *dance*, generating the $\sqsupset$ relation. However, because the edit occurs within the scope of the newly-introduced negation, $\sqsupset$ is projected as $\sqsubset$, and $\sqsubset$ joined with $\sqsubset$ yields $\sqsubset$. The sixth edit deletes a generic modifier (*blue*), which generates the $\sqsubset$ relation by default. This time the edit occurs within the scope of *two* downward-monotone operators (*without* and negation), so we have two inversions of monotonocity, and $\sqsubset$ is projected as $\sqsubset$. Again, $\sqsubset$ joined with $\sqsubset$ yields $\sqsubset$. Finally, the seventh edit substitutes *jeans* with its hypernym *pants*, generating the $\sqsubset$ relation. Again, the edit occurs within the scope of two downward-monotone operators, so $\sqsubset$ is projected as $\sqsubset$, and $\sqsubset$ joined with $\sqsubset$ yields $\sqsubset$. Thus p entails h.

Of course, the edit sequence shown is not the only sequence which can transform p into h. A different edit sequence might yield a different sequence of intermediate steps, but the same final result. Consider, for example, the edit sequence shown in table 6.13. Note that the lexical entailment relation $\beta(e_i)$ generated by each edit is the same as before. But because the edits involving downward-monotone operators (namely, INS(*n't*) and DEL(*refused to*)) now occur at different points in the edit sequence, many of the atomic entailment relations $\beta(x_{i-1}, e_i)$ have changed, and thus the sequence of

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
		Jimmy Dean refused to move without blue jeans			
1	INS(*did*)		≡	≡	≡
		Jimmy Dean did refuse to move without blue jeans			
2	INS(*not*)		^	|	|
		Jimmy Dean did refuse not to move without blue jeans			
3	DEL(*refuse to*)		|	|	≡⊏⊐|#
		Jimmy Dean didn't move without blue jeans			
4	DEL(*blue*)		⊏	⊏	•
		Jimmy Dean didn't move without jeans			
5	SUB(*jeans*, *pants*)		⊏	⊏	•
		Jimmy Dean didn't move without pants			
6	SUB(*move*, *dance*)		⊐	⊐	•
		Jimmy Dean didn't dance without pants			
7	SUB(*Jimmy Dean*, *James Dean*)		≡	≡	•
		James Dean didn't dance without pants			

Table 6.14: Analysis of a more complex inference, third try.

joins has changed as well. In particular, edits 3 and 4 occur within the scope of *three* downward-monotone operators (negation, *refuse*, and *without*), with the consequence that the ⊏ relation generated by each of these lexical edits is projected as ⊐. Likewise, edit 5 occurs within the scope of two downward-monotone operators (negation and *refuse*), and edit 6 occurs within the scope of one downward-monotone operator (negation), so that | is projected as ⌣. Nevertheless, the ultimate result is still ⊏.

However, it turns out not to be the case that every edit sequence which transforms p into h will yield equally satisfactory results. Consider the sequence shown in table 6.14. The crucial difference in this edit sequence is that the insertion of *not*, which generates lexical entailment relation ^, occurs within the scope of *refuse*, so that ^ is projected as atomic entailment relation | (see section 6.3.4). But the deletion of *refuse to* also produces atomic entailment relation | (see section 6.3.2), and | joined with | yields a relatively uninformative union relation, namely $\bigcup\{\equiv, \sqsubset, \sqsupset, |, \#\}$ (which could also be described as the NON-EXHAUSTION relation). The damage has been done: further joining leads directly to the "black hole" relation •, from which there is no escape.

Note, however, that even for this infelicitous edit sequence, our inference method

has not produced an *incorrect* answer (because the • relation includes the ⊏ relation), only an *uninformative* answer (because it includes all other relations in $\mathfrak{B}$ as well). In fact, through all the examples we have considered, we have encountered no evidence that our inference method is unsound, i.e., that it can lead to an incorrect (as opposed to merely uninformative) result. We'll consider whether this is true in general in the final section of this chapter.

6.6 Is the inference method sound?

We have seen that the inference method described in section 6.4 is able to produce the desired entailment relation (that is, one which is not only correct but as informative as possible) on a broad variety of example problems, including inferences involving semantic containment, semantic exclusion, implicativity, and nested inversions of monotonicity. We have also seen that the method fails to achieve inferential *completeness*: that is, there exist valid inferences (such as instantiations of de Morgan's laws for quantifiers, section 6.5.4) for which the method is not able to produce the desired entailment relation, but only a less informative (though compatible) relation. What about the converse property of inferential *soundness*? When the inference method produces an informative result, can we be confident that the result is correct?

Unfortunately, we cannot provide a *formal* proof of soundness. Indeed, to seek such a proof would be to misconstrue the very purpose of natural logic, which is, at its heart, a model of *informal* reasoning. It is true, of course, that in order to make the model precise and to implement it in code, we have erected a certain amount of formal scaffolding: the definition of the set $\mathfrak{B}$ of basic entailment relations and the algebra of their joins (chapter 5), and the notion of projectivity signatures (this chapter). This formal framework, if divorced from natural language and viewed purely as a calculus of set relations, would no doubt support a formal proof of soundness. But such a calculus would not be a model of natural logic, nor would it provide an account of natural language inference; thus, it is a topic for another dissertation.

In a model of natural language inference, the informality of natural language necessarily intrudes. The correctness of results produced by the inference method

described in section 6.4 depends on many factors which defy formalization. First, as noted above, we must find a suitable sequence of edits which transforms p into h (and the inference method *per se* offers no guidance on how to find such a sequence, or indeed, whether one exists). Second, we must predict the correct entailment relation for each edit. Cases like SUB(*jeans*, *pants*) are straightforward enough, but (as we noted in section 6.1) other cases are not so clear-cut. Lexical ambiguity is one issue: whether *bank* $\sqsubset$ *building* depends on which sense of *bank* is intended. Vagueness is another: does *three* $\sqsubset$ *several*? Does *method* $\equiv$ *strategy*? Finally, we must correctly determine the projectivity properties of semantic functions in our sentences, and properly identify the extent of their arguments. Classic scope ambiguities (as in *Every man loves some woman*) may be one issue, but there are other potential sources of difficulty. For example, we noted in section 6.2.5 that while most verbs project $|$ as $\#$, those which encode functional relations project $|$ as $|$. But whether a particular verb encodes a functional relation may not be entirely clear-cut.

Many of the difficulties just described involve properly identifying the semantic properties of lexical items involved in the inference. A further consideration—though it is purely theoretical—is that lexical items exhibiting certain combinations of semantic properties could, in principle, render our inference method unequivocally unsound.[8] Suppose there were an English adverb *X-ly* having the following (admittedly bizarre) combination of properties: first, it is semantically vacuous (so that $\beta(\text{DEL}(X\text{-}ly)) = \equiv$); and second, it is downward monotone (so that its projectivity signature includes $\{\sqsubset{:}\sqsupset\}$). Now consider the following inference:

p: *John runs X-ly.*

h: *John moves.*

Here, p is transformed into h by two edits, which can occur in either order: SUB(*runs, moves*) and DEL(*X-ly*). Of course, $\beta(\text{SUB}(runs, moves)) = \sqsubset$. Now, the result produced by the inference method will depend upon the order in which the edits are applied. If we first apply the DEL and then the SUB, the result is $\sqsubset$, because the DEL is semantically vacuous, and after it has occurred, *X-ly* is no longer present

[8]The following example is due to an anonymous reviewer for the Eighth International Conference on Computational Semantics, to whom we are indebted.

to affect the SUB, which generates $\sqsubset$. However, if we first apply the SUB and then the DEL, then the result is $\sqsupset$, because the presence of the downward-monotone *X-ly* causes the lexical entailment relation $\sqsubset$ produced by the SUB to be projected as $\sqsupset$. Since it is not possible that $p \sqsubset h$ and $p \sqsupset h$, at least one of these two results must be incorrect. Therefore, if expressions like *X-ly* can occur in natural language, then at least some edit orders can cause our inference method to produce incorrect results.

We are not aware of any actual natural language expression having the unusual combination of properties exemplified by *X-ly*. Indeed, it is difficult to imagine that such an expression could be coined—what, exactly, might it mean? It therefore seems a plausible hypothesis that expressions like *X-ly* do not occur in natural languages, and consequently that, in practice if not in principle, such expressions pose no threat to the soundness of our inference method. This claim resembles the typological claims that have been made about certain monotonicity patterns of quantifiers existing in natural languages. But because we can offer no concrete evidence to support this claim, it remains a hypothesis only.

Whatever the theoretical limitations of our model of natural logic, it has been shown in practice successfully to address a wide range of inference problems. In section 6.5, we illustrated this using a variety of toy examples; in chapter 7, we will describe a computational implementation of this model and a large-scale evaluation of its effectiveness on real-world inference problems.

Chapter 7

The NatLog system

The model of natural logic developed in chapters 5 and 6 may be an elegant theory, and we have seen that it successfully explains a handful of contrived examples. But is the model just a toy, or can it be effectively implemented and applied to real-world examples of natural language inference? This chapter aims to answer those questions.[1]

The NatLog system is a Java implementation of our model of natural logic. In the following pages, we present the high-level architecture of the system (section 7.1) and provide detailed descriptions of each of its five key components (sections 7.2 through 7.6). Next, we illustrate the operation of the system (section 7.7) using a concrete example first introduced in the previous chapter. We then discuss evaluations of the effectiveness of the system on two well-known collections of NLI problems: the FraCaS test suite (section 7.8) and the RTE3 test suite (section 7.9).

7.1 System architecture

The NatLog system uses a multi-stage pipelined architecture in which the task of *alignment* (that is, identifying corresponding entities and predicates in p and h) is distinguished from the task of *entailment classification* (determining whether p entails h). While this division of labor may seem obvious, it is by no means a given: in many

[1] The material in this chapter is derived in large part from (MacCartney and Manning 2008).

RTE systems, alignment is not explicitly formulated as a separate task, but is achieved implicitly as part of entailment classification. In adopting a pipelined approach, we follow a precedent established by Marsi and Krahmer (2005), and developed further in the Stanford RTE system, as described in (MacCartney et al. 2006) and in chapter 4 of this dissertation.

As we saw in chapter 4, the Stanford RTE system uses a pipeline of three stages, comprising linguistic analysis, alignment, and entailment classification. The NatLog pipeline has a similar structure, but places far great emphasis on entailment classification, which it divides into three separate stages. Altogether, then, the NatLog pipeline includes five stages of processing:

1. **Linguistic analysis** (section 7.2) involves applying standard linguistic analysis tools (tokenization, lemmatization, parsing, etc.) to the input sentences p and h in order to generate a variety of linguistic annotations which prepare the ground for further processing.

2. **Alignment** (section 7.3) means establishing a sequence of atomic edits which transforms p into h. The NatLog system does not itself have the capacity to perform alignment, but relies instead on alignments derived from outside sources (such as the MANLI system introduced in chapter 3, or the Stanford RTE aligner described in chapter 4).

3. **Lexical entailment classification** (section 7.4) is the core of the NatLog system: it uses machine learning and a variety of lexical resources to predict an entailment relation for each atomic edit, based solely on features of the lexical items involved, independent of context.

4. **Entailment projection** (section 7.5) aims to predict the entailment relation which holds across each atomic edit by considering the impact of the context in which the edit occurs on the lexical entailment relation produced in the previous stage.

5. **Entailment joining** (section 7.6) combines the atomic entailment relations

predicted by the preceding stage across the chain of edits in order to produce the entailment relation which holds between p and h.

The following sections describe each of these stages in greater detail.

7.2 Linguistic analysis

The NatLog pipeline begins by applying a number of off-the-shelf linguistic analysis tools to the input sentences p and h, in order to generate linguistic annotations which will be useful later in processing. We rely on the Stanford parser (Klein and Manning 2003), a Penn Treebank-trained statistical parser, for tokenization, part-of-speech tagging, and syntactic phrase-structure parsing. We also generate a lemma for each word token using a Java port of the finite-state analyzer of Minnen et al. (2001). Lemmas and part-of-speech tags are used in lexical entailment classification (section 7.4), while the syntactic parses are used in entailment projection (section 7.5). However, relative to some other NLI systems, the linguistic pre-processing performed by NatLog is relatively light. In particular, we make no attempt to perform named entity recognition, coreference resolution, dependency parsing, or semantic role labeling. While all these kinds of information could potentially be useful in later stages of processing, we have chosen to focus our development efforts on our core model of natural logic, rather than on incremental improvements to linguistic annotation.

This stage includes one other important type of linguistic analysis: *monotonicity marking* of p and h, in which we compute the effective monotonicity for each token span in each input sentence. While this computation is actually performed as part of the linguistic analysis stage, logically it forms part of the fourth processing stage (entailment projection); thus we postpone further discussion until section 7.5.

7.3 Alignment

In the second stage of processing, we establish an *alignment* between the premise p and hypothesis h, represented by a sequence of *atomic edits*, operating over phrases,

which transforms p into h. (Following the usage common in the machine translation literature, we use the term "phrase" to refer to a contiguous span of word tokens, whether or not the tokens constitute a syntactic phrase.) This alignment representation is symmetric and many-to-many, and is general enough to include various other alignment representations as special cases. We define four edit types:

- an EQ edit connects a phrase in p with an equal (by word lemmas) phrase in h
- a SUB edit connects a phrase in p with an unequal phrase in h
- a DEL edit covers an unaligned phrase in p
- an INS edit covers an unaligned phrase in h

Each edit is parameterized by the token indices at which it operates, and edit indices may "cross", permitting representation of movement. (This is the same representation of alignment used by the MANLI system, described in chapter 3.)

The NatLog system does not itself include any facility for finding good alignments; rather, its purview is the identification of entailment relations between aligned sentence pairs. NatLog therefore relies on alignments derived from other sources: either human annotators, or automatic alignment systems such as the MANLI system (chapter 3) or the Stanford RTE aligner (chapter 4).

Although NatLog does not itself generate alignments, but merely reads them in from an outside source, it does perform one important bit of processing as part of the alignment stage: name, heuristic ordering of edits. The outside sources from which NatLog obtains alignments typically do not specify any particular ordering in which the edits contained in an alignment should be applied. However, the inference method implemented by NatLog requires that such an ordering be established (see section 6.4), since this ordering defines a path from p to h through intermediate forms, and thereby decomposes the inference problem into a sequence of *atomic inference problems*, one for each atomic edit. Since NatLog is free to choose whatever edit order it prefers, it uses heuristic rules to select an order which maximizes the benefit of the monotonicity marking performed in the first stage of processing (but described in section 7.5).

In particular, NatLog prefers to put DEL edits at the beginning of the edit sequence, SUB edits in the middle, and INS edits at the end. There is one complication: the relative ordering of edits involving downward-monotone (or non-monotone) operators (e.g., the deletion of *not*) may influence the effective monotonicity applicable for other edits. Therefore, in choosing an edit order, NatLog places any edits involving downward-monotone (or non-monotone) operators in a separate group, after all other DEL and SUB edits, and before all other INS edits. This has the benefit that when NatLog needs to compute the effective monotonicity for a particular edit (in the fourth stage), it is able to use the monotonicity markings on p for all ordinary DEL and SUB edits, and the monotonicity markings on h for all ordinary INS edits. (By "ordinary", we simply mean "not involving downward-monotone or non-monotone operators".)

7.4 Predicting lexical entailment relations

Much of the heavy lifting in the NatLog system is done in the third stage of processing, by the *lexical entailment model*, which is a concrete implementation of the principles described in section 6.1. This model has the job of predicting a lexical entailment relation for each atomic edit in the alignment determined in the previous stage. Its prediction is based solely on features of the lexical items involved in the edit, independent of the projective properties of the context in which the edit occurs. (For example, this model should assign the entailment relation $\sqsupset$ to the edit SUB(*weapon*, *rifle*), regardless of whether the context at the locus of the edit is upward-monotone, downward-monotone, or something else.)

To makes its predictions, the lexical entailment model uses machine learning techniques, and exploits a plethora of external lexical resources containing information about relationships between word meanings. For each edit, the model generates a feature representation (section 7.4.1) which encapsulates all available information about the type of the edit and the words involved in the edit, and then applies a decision tree classifier (section 7.4.2) which has been trained on a large set of manually-annotated lexical entailment problems.

7.4.1 Feature representation

The process of predicting a lexical entailment relation for a given atomic edit begins with generating a feature representation of the edit, which is a real-valued vector encoding a variety of information about the edit: its type, its size, characteristics of the words involved the edit, and (in the case of SUB edits) information about the semantic relationship between the substituends. This feature representation then serves as an input to the classifier described in section 7.4.2.

We begin by describing the features active for SUB edits, in which we are particularly concerned with the semantic relationship between the substituends, that is, the phrases (often single words) from p and h involved in the edit.

WordNet-derived features. Perhaps the most important features for SUB edits are those which exploit WordNet to measure the semantic relatedness of the substituends, using various relation types, including synonymy, antonymy, and hypernymy. Each of these features takes values on the interval [0..1].

- The `WNSyn` feature indicates synonymy: it takes value 1 iff the substituends are synonyms per WordNet (i.e., belong to the same synset), and 0 otherwise.
- The `WNAnt` feature indicates antonymy: it takes value 1 iff the substituends are antonyms per WordNet, and 0 otherwise.
- The `WNHyper` feature indicates hypernymy. If the h phrase is a (direct or indirect) hypernym of the p phrase in WordNet, then `WNHyper` takes the value $1 - \frac{n}{8}$, where n if the number of links which must be traversed in the WordNet hypernym hierarchy to get from the p phrase to the h phrase. Otherwise, `WNHyper` takes value 0. For example, `WNHyper`$(owl, bird) = 0.75$, while `WNHyper`$(bird, owl) = 0.0$.
- The `WNHypo` feature indicates hyponymy, and is simply the inverse of the `WNHyper` feature. Thus `WNHypo`$(owl, bird) = 0.0$, while `WNHypo`$(bird, owl) = 0.75$.

- The `JiCo` feature is a measure of semantic relatedness based on the well-known Jiang-Conrath WordNet distance metric (Jiang and Conrath 1997), which measures the distance between two WordNet synsets s_1 and s_2 as

$$d_{JC}(s_1, s_2) \stackrel{\text{def}}{=} IC(s_1) + IC(s_2) - 2 \cdot IC(lso(s_1, s_2))$$

  where $IC(s)$ is the *information content* of synset s (that is, the negative of the logarithm of its frequency of occurrence in a large corpus), and $lso(s_1, s_2)$ is the *least superordinate* (most specific hypernym) of synsets s_1 and s_2. Possible values for d_{JC} range from a minimum of 0 (for synonyms) to a theoretical maximum of $2 \cdot \max_s(IC(s))$. Thus, to convert d_{JC} to a measure of semantic relatedness on [0..1], we use

$$\texttt{JiCo}(s_1, s_2) \stackrel{\text{def}}{=} 1 - \frac{d_{JC}(s_1, s_2)}{2 \cdot \max_s(IC(s))}$$

  This formula gives results that are intuitively pleasing, although it has the consequence that comparatively few pairs of terms receive a score near 0.

Features based on other lexical resources. In addition to the WordNet-based features, we use a few other features based on external lexical resources which are in some ways complementary to WordNet. Like the WordNet-based features, these features have values on [0..1].

- The `NomB` feature uses information from the NomBank lexical resource (Meyers et al. 2004) to indicate derivationally related words having different parts of speech, particularly nominalizations. We extracted from the NOMLEX-PLUS.0.8 file about 4,000 related noun/verb pairs (such as *accusation*/*accuse*) and about 2,500 related adjective/adverb pairs (such as *angry*/*angrily*). The `NomB` feature is set to 0.75 if the substituends are among these pairs, or to 0 otherwise.

- The `DLin` feature is a measure of semantic relatedness based on a thesaurus compiled automatically by Dekang Lin (Lin 1998). This resource, which covers

nouns, verbs, and adjectives, estimates the relatedness between a pair of words or phrases by measuring their *distributional similarity*, that is, the similarity between the distributions, in a very large corpus, of the grammatical relationships in which they participate. It is thus a useful complement to the information obtained from manually-compiled resources like WordNet and NomBank. Because the scores obtained from the thesaurus are not normalized, we map them onto [0..1] by dividing every score by the maximum score observed in the thesaurus (separately for each part of speech).

String similarity features. Since no static lexical resource can possibly contain every pair of words which might be encountered, the robustness of the system can be greatly enhanced by including features which measure only string similarity. Such features are particularly helpful for identifying the equivalence between different versions of proper names, which are commonly missing from static lexical resources. For example, string similarity can help us to identify alternate or erroneous spellings of a name (*Ahmadinejad*, *Ahmadi-Nejad*, or *Ahmadinejab*), or shorter and longer versions of the same name (*Lincoln* vs. *Abraham Lincoln*).

- The `LemStrSim` feature estimates the similarity between words w_1 and w_2 using a function based on the string edit distance between the word lemmas:

$$\texttt{LemStrSim}(w_1, w_2) = \max\left[0, 1 - \frac{dist(lemma(w_1), lemma(w_2))}{\max(|lemma(w_1)|, |lemma(w_2)|) - k}\right]$$

  Here $lemma(w)$ denotes the lemma of word w, as generated by the finite state transducer of Minnen et al. (2001); $dist()$ denotes Levenshtein string edit distance (Levenshtein 1966); and $|\cdot|$ denotes string length. k is a penalty parameter whose purpose is to prevent assigning high similarity to very short pairs of unrelated words with similar spelling, such as *or* and *for*, or *she* and *the*. For example, if $k = 0$, then $\texttt{LemStrSim}(she, the) = 0.67$, but if $k = 2$ (the valued we used in practice), then $\texttt{LemStrSim}(she, the) = 0.0$. The similarity assigned to longer word pairs is less affected by the penalty parameter. For example, with $k = 2$, $\texttt{LemStrSim}(Ahmadinejad, Ahmadinejab) = 0.89$.

- The `LemSubSeq` feature is designed to identify the case where one of the substituends is a multi-word phrase which contains the other as a sub-phrase, as in SUB(*Lincoln*, *Abraham Lincoln*) or SUB(*Dick and Jane*, *Jane*). Unlike the other features presented so far, the `LemSubSeq` feature takes values which are entailment relations (and which are subsequently encoded by a bundle of boolean features). In particular:
  - `LemSubSeq`$(p, h) = \sqsubset$ iff h is a subsequence of p
  - `LemSubSeq`$(p, h) = \sqsupset$ iff p is a subsequence of h
  - `LemSubSeq`$(p, h) = \equiv$ iff p and h are equal
  - `LemSubSeq`$(p, h) = \#$ otherwise

  As the name suggests, the `LemSubSeq` feature works by comparing word lemmas, not merely surface forms.

Lexical category features. We also include a number of features which describe the lexical categories to which the substituends belong.

- The `Light` feature is a boolean feature which is activated just in case both substituends contain only semantically "light" words, that is, words which have a limited impact on the semantics of the sentences in which they appear. The intent is that substitutions involving light words should be predicted to generate lexical entailment relation $\equiv$. We define light words to include punctuation, prepositions, possessive markers, articles, auxiliary verbs, and expletives. Of course, we do not mean to make a strong claim that all such words are semantically vacuous—rather, this definition reflects a pragmatic choice which seems to work well in practice.

- The `Preps` feature is a boolean feature which is activated just in case both substituends are prepositions. The presence of this feature encourages NatLog to be somewhat liberal (i.e., biased toward predicting $\equiv$) regarding substitutions involving prepositions. As we argued in section 6.1.2, this is often—though not always—an appropriate choice.

- The `Pronoun` feature is a boolean feature which is activated just in case both substituends are pronouns. The presence of this feature encourages NatLog to be somewhat liberal regarding substitutions involving pronouns.

- The `NNNN` feature is a boolean feature which is activated just in case either both substituends are common nouns, or both substituends are proper nouns. The motivation for this feature is to encode the background assumption that most such pairs of nouns should (in the absence of evidence to the contrary) be predicted to stand in the | relation, as described in section section 6.1.1. For example, we want *apple* | *banana* and *Atlanta* | *Boston*. Of course, for specific pairs of nouns, the $\equiv$, $\sqsubset$, or $\sqsupset$ relation might be more appropriate, but we hope that in such cases some other featurizer will provide the evidence we need to make that prediction.

- The `Quantifier` feature is designed to enable predicting appropriate lexical entailment relations for SUB edits involving quantifier phrases, as outlined in section 6.1.2. Its output space is the set $\mathfrak{B}$ of basic entailment relations, and its value is intended to serve as the predicted lexical entailment relation for the given substitution. This feature function contains quite a bit of manually encoded knowledge. It defines a number of *quantifier categories*, such as:

  - `ALL`: including *all*, *every*, *each*, *both*, *everybody*, etc.
  - `SOME`: including *some*, *something*, *several*, *one or more*, etc.
  - `NONE`: including *no*, *nothing*, *nobody*, *no one*, *neither*, etc.
  - `NUM`: specific cardinal numbers

  In all, we define about a dozen quantifier categories, along with patterns used to identify occurrences of each category. Finally, for many category pairs, we define relations in $\mathfrak{B}$ which should be predicted to result from a substitution of quantifiers in those categories. So, for example, we have:

$$\begin{aligned} \text{SUB}(\texttt{ALL}, \texttt{SOME}) &\rightarrow \sqsubset \\ \text{SUB}(\texttt{ALL}, \texttt{NONE}) &\rightarrow | \\ \text{SUB}(\texttt{ALL}, \texttt{NUM}) &\rightarrow \# \end{aligned}$$

and so on. These definitions are driven both by theory and by pragmatism. If either substituend fails to be recognized as a quantifier phrase, or if we have not defined a predicted entailment relation for the given pair of quantifier categories, then the feature takes value #.

Miscellaneous additional features for SUB edits. Our feature representation includes a couple of additional features which do not fit neatly into any of the preceding feature categories.

- The `NeqNum` feature is a boolean feature designed to identify number mismatches. It is activated just in case the substituends are both numbers and are not equal.
- The `MiscSub` feature is used to encode knowledge about specific pairs of expressions which do not fit neatly into any of the other features. Its output space is the set $\mathfrak{B}$ of basic entailment relations, and it includes hand-coded mappings from specific pairs of expressions to lexical entailment relations which should be predicted for SUB edits involving those pairs. In particular, the `MiscSub` feature encodes the knowledge that:
  - SUB(*and*, *or*) should yield $\sqsubset$.
  - SUB edits involving different ways of expressing the genitive (e.g., *'s*, *of*, *his*, *her*, *its*) should yield $\equiv$.
  - SUB edits involving certain pairs of prepositions should yield $|$ (e.g., *after* $|$ *before*, *inside* $|$ *outside*), rather than the default $\equiv$.

What about DEL and INS edits? These are handled as one: in fact, when asked to predict a lexical entailment relation for an INS edit, NatLog first converts it to a DEL edit (by swapping the roles of p and h), then applies the lexical entailment model (which is designed for DEL edits), and then returns the converse of the resulting prediction (for example, swapping $\sqsubset$ and $\sqsupset$).

The feature representation used for DEL edits is much smaller than the one used for SUB edits. (As noted in section 6.1.3, most DEL edits just have $\sqsubset$ as the target

lexical entailment relation, so this is an easier prediction problem.) In fact, for DEL edits, we use only the `Light` and `Pronoun` features, plus an additional feature which applies only to DELs:

- The `MiscDel` feature is used to encode knowledge about specific expressions which do not fit neatly into any of the other features. Its output space is the set $\mathfrak{B}$ of basic entailment relations, and it includes hand-coded mappings from specific expressions to lexical entailment relations which should be predicted for DEL edits involving those pairs. In particular, the `MiscDel` feature encodes the knowledge that:
  - DEL(*not*) and DEL(*false*) should yield $\wedge$, while DEL(*true*) should yield $\equiv$.
  - Certain non-intersective adjectives have non-default behavior. DEL(*fake*) and DEL(*former*) should yield $|$, while DEL(*alleged*) should yield $\#$.
  - Deletions of modal verbs often yield $\#$. Thus DEL(*could*) and DEL(*should*) should yield $|$.

  Finally, the `MiscDel` feature is also where we encode knowledge about lexical entailment relations generated by deletions of implicative and factive operators, as described in section 6.3.2 and section 6.3.3. To do this, we use a custom-built resource which maps a few dozen common implicative and factive operators to their implication signatures (see section 6.3.1), based on a list obtained from researchers at PARC. For example, this map tells us that *force* has implication signature $+/\circ$, and thus DEL(*force*) should yield $\sqsubset$; while *fail* has implication signature $-/+$, and thus DEL(*fail*) should yield $\wedge$.

7.4.2 Classifier training

The feature representation described in section 7.4.1 serves as input to a classifier whose job is to predict lexical entailment relations. For this purpose, we use a J48 decision tree classifier from the Weka machine learning library (Witten and Frank 2005). Our motivation for choosing a decision tree classifier—as opposed to a linear model such as a maximum entropy classifier or a support vector machine—is that we expect the decision space to be richly structured, with most classification decisions

problems involving SUB edits (1,525 examples)

reln	count	examples
$\equiv$	1,042	SUB(*blast*, *explosion*), SUB(*had*, *has*), SUB(*Alfred Nobel*, *Nobel*)
$\sqsubset$	95	SUB(*jet*, *plane*), SUB(*Italy*, *nation*), SUB(*deafening*, *loud*)
$\sqsupset$	86	SUB(*robots*, *mini-robots*), SUB(*acquired*, *bought*), SUB(*some*, *one*)
$\wedge$	8	SUB(*no*, *some*), SUB(*confirmed*, *unconfirmed*)
$\mid$	191	SUB(*1972*, *1975*), SUB(*Rwanda*, *Uganda*), SUB(*salt*, *vinegar*)
$\smile$	0	
#	103	SUB(*fetal*, *human*), SUB(*gatherers*, *pygmies*), SUB(*baby*, *girl*)

problems involving DEL edits (924 examples)

reln	count	examples
$\equiv$	191	DEL(*has*), DEL(*manage to*), DEL(*genuine*)
$\sqsubset$	694	DEL(*last week*), DEL(*predictably*), DEL(*single*)
$\sqsupset$	3	DEL(*tried to*)
$\wedge$	6	DEL(*not*), DEL(*forgot to*), DEL(*it is false that*)
$\mid$	7	DEL(*former*), DEL(*refused to*), DEL(*prevented*)
$\smile$	2	DEL(*hesitate to*)
#	21	DEL(*may*), DEL(*needed to*), DEL(*concluded*)

Table 7.1: Examples of training problems for the lexical entailment model.

hinging on just one or two features. While linear models excel at combining evidence from a large number of features, in our problem setting we anticipate comparatively little need to combine evidence.

In order to train the classifier, we constructed a training set containing 2,449 *lexical entailment problems*, derived from alignments of NLI problems from various sources. Of these problems, 1,525 represented SUB edits, and consisted of a pair of words or phrases. The other 924 represented DEL edits, and consisted of just a single word or phrase. Each problem was manually annotated with one of the seven relations in $\mathfrak{B}$, indicating the lexical entailment relation generated by the given edit. As table 7.1 makes clear, the distribution of data was far from even. Because the problems were

derived from actual alignments, most SUB problems consisted of pairs of equal (or nearly equal) expressions, and thus were labeled with relation ≡. The second most common label for SUB problems was |; many of these problems contained pairs of unequal proper nouns or mutually exclusive common nouns. Among the DEL problems, the overwhelming majority were labeled with relation ⊏, reflecting the prevalence of intersective modification, as described in section 6.1.3. The second most common label for DEL problems was ≡; most such problems involved expressions treated as semantically vacuous, such as auxiliary verbs, though a few involved implicatives.

During training, the decision tree was minimally pruned, and the resulting tree contains about 180 leaves. When tested on the training data, the classifier achieves >99% accuracy, indicating that our feature representation successfully captures nearly all relevant distinctions between examples.

7.5 Entailment projection

Whereas the third stage predicts a lexical entailment relation for each edit, it is the job of the fourth stage of processing to determine the atomic entailment relation which holds across each edit: that is, the entailment relation which holds between successive intermediate forms in the path from p to h defined by the alignment. As we described in section 6.2, this atomic entailment relation will depend not only on the lexical entailment relation produced by the edit (which is independent of context), but also on the projective properties of the context in which the edit occurs. Thus, predicting atomic entailment relations involves two tasks: first, establishing the projective properties which apply in different regions of the sentences involved in the problem; and second, determining how these properties affect the projection of lexical entailment relations into atomic entailment relations for specific edits. We describe the first of these tasks as *monotonicity marking* and the second as *predicting projections*.

unary operator: *without*
  pattern: `IN < /^[Ww]ithout$/`
  argument 1: monotonicity DOWN on dominating PP
    pattern: `__ > PP=proj`

binary operator: *most*
  pattern: `JJS < /^[Mm]ost$/ !> QP`
  argument 1: monotonicity NON on dominating NP
    pattern: `__ >+(NP) (NP=proj !> NP)`
  argument 2: monotonicity UP on dominating S
    pattern: `__ >> (S=proj !> S)`

Figure 7.1: Some monotonicity operator type definitions.

7.5.1 Monotonicity marking

The goal of monotonicity marking is to identify within the sentences p and h any operators which have monotonicity DOWN or NON, and to predict the likely extent of their arguments.

(In an ideal world, we would identify not only operators having monotonicity DOWN or NON, but operators having any projectivity signature other than the one which maps each entailment relation to itself. In that case, we would describe the task as "projectivity marking", as opposed to simply "monotonicity marking". The fact that NatLog does only monotonicity marking, rather than full projectivity marking, is one of the limitations of the current system.)

For example, in the sentence *Jimmy Dean refused to move without blue jeans*, we want to recognize that *refused* is a downward-monotone operator whose argument is the embedded sentence *to move without blue jeans*, and that *without* is a downward-monotone operator whose argument is the noun phrase *blue jeans*. Similarly, in the sentence *The best player doesn't always win*, we want to recognize that *best* is a non-monotone operator whose argument is the noun *player*, and that *doesn't* is a downward-monotone operator whose argument is the entire sentence.

Our choice of a Treebank-trained parser (driven by the goal of broad coverage) somewhat complicates this effort, because the nesting of constituents in phrase-structure parses does not always correspond to the structure of idealized semantic

composition trees. Our solution is imperfect, but largely effective. We define a list of *monotonicity operator types* (that is, categories of operators with monotonicity DOWN or NON, such as implicatives like *refuse*, or prepositions like *without*), and for each such operator type we specify its arity and a Tregex tree pattern (Levy and Andrew 2006) which permits us to identify its occurrences in our Treebank parses. We also specify, for each argument position of each operator type, both the monotonicity class (DOWN or NON) and another Tregex pattern which helps us to determine the likely extent of the argument in a phrase-structure tree. (Figure 7.1 shows some example definitions.)

Although we have presented monotonicity marking as belonging (logically) to the fourth stage of NatLog processing, in fact it happens during the first stage, as part of linguistic analysis. After parsing, we match the tree patterns in each of the monotonicity operator type definitions against the phrase-structure parse trees for p and h, and add annotations to parse tree nodes to indicate constituents which are either monotonicity operators or arguments thereto.

Note that we make no attempt to resolve scope ambiguities, such as that exhibited by the sentence *Every man loves some woman*, but we find that in practice this rarely causes problems: the default behavior usually leads to the right predictions. There are at least three reasons for this. First, inferences which hinge on scope ambiguities are rare. Second, the assumption that the first quantifier takes wide scope turns out to be a fairly reliable heuristic. Third, in many cases the effective monotonicity at each point in the sentence will be same no matter how the scope ambiguity is resolved. This is the case with *Every man loves some woman*: whether *some* is interpreted as having wide or narrow scope, the effective monotonicity at *man* will be DOWN, while the effective monotonicity in the rest of the sentence will be UP.

7.5.2 Predicting projections

Having computed a lexical entailment relation for each edit in the third stage of processing, NatLog uses the monotonicity markings on the parses of p and h to determine the projection of each lexical entailment relation into an atomic entailment

relation. As we described in section 7.3, NatLog facilitates this process by choosing an edit ordering in which DEL and SUB edits not involving monotonicity operators are applied first, and INS edits not involving monotonicity operators are applied last. Consequently, in computing the atomic entailment relation for a particular edit, we use monotonicity markings from p for DEL and SUB edits, and monotonicity markings from h for INS edits. (This strategy can go wrong if there are multiple edits involving monotonicity operators, but this rarely occurs in practice. A more principled approach would be separately to compute monotonicity markings on every intermediate form on the path from p to h defined by the alignment.)

Given a parsed sentence with monotonicity markings, how do we actually determine the projection of a given entailment relation? Our first task is to establish the effective monotonicity at the locus of the edit. First, we find the smallest parse constituent containing the tokens involved in the edit. Next, we trace a path upward through the parse tree from this constituent to the root, collecting any monotonicity markings we find along the way. (We exclude any monotonicity markings which were generated by an operator which is itself an argument to the edit.) If any of these markings is NON, we conclude that the effective monotonicity is NON. Otherwise, we count the number of markings which are DOWN, and if this number is odd, we conclude that the effective monotonicity is DOWN. Otherwise, we conclude that the effective monotonicity is UP.

If the effective monotonicity is NON, then we assume that every lexical entailment relation is projected as atomic entailment relation #. If the effective monotonicity is DOWN, then we assume that every lexical entailment relation is projected as its *dual under negation* (in the sense of section 5.3.1): thus $\sqsubset$ is projected as $\sqsupset$, and vice-versa. Finally, if the effective monotonicity is UP, then we assume that every lexical entailment relation is projected without change.

7.6 Joining entailment relations

In the final stage of processing, the atomic entailment relations predicted for each edit (which result from the projectivity computations in the previous stage) are combined,

one by one, across the chain of atomic edits, to produce a final, overall prediction for the inference problem. The combination is performed using the join operation $\bowtie$ defined in section 5.6. This is a deterministic operation, and is simple to compute. (In fact, the 7×7 join table for relations in $\mathfrak{B}$ is precomputed, so this stage of processing involves only lookups.)

As we alluded to in section 5.6.3, the NatLog system has no need to represent unions of relations in $\mathfrak{B}$ which may result from joining two relations in $\mathfrak{B}$. Rather, those joins which would result in a union relation are approximated instead by #. Note that if a chain of joins ever reaches #, then it is "stuck" there: no further joining can lead to any other result. Consequently, prediction errors earlier in the processing pipeline are likely to cause the final prediction to be #, and this tendency becomes stronger as the number of atomic edits grows. Because # is the least informative relation in $\mathfrak{B}$ (permitting neither the truth nor falsity of one of its arguments to be inferred from the truth or falsity of the other), we might say that NatLog has an innate tendency to be conservative in its predictions, in the sense that when it runs into problems, it tends to predict neither entailment nor contradiction.

This chaining of entailments across edits can be compared to the method presented in (Harmeling 2007); however, that approach assigns to each edit merely a probability of preserving truth, not an entailment relation.

7.7 An example of NatLog in action

Because our presentation of the five stages of the NatLog system in sections 7.2 through 7.6 has perhaps been rather abstract, we now describe a concrete example which illustrates the operation of the NatLog system. For this purpose, we use an inference problem first introduced in section 6.5.5, namely:

p: *Jimmy Dean refused to move without blue jeans.*

h: *James Dean didn't dance without pants.*

During the linguistic analysis stage, these sentences are parsed using the Stanford parser; the results are shown in figure 7.2. The operation of the remaining four stages is depicted in table 7.2.

```
(S
  (NP (NNP Jimmy) (NNP Dean))
  (VP (VBD refused)
    (S
      (VP (TO to)
        (VP (VB move)
          (PP (IN without)
            (NP (JJ blue) (NNS jeans)))))))
  (. .))
```

parse for p

```
(S
  (NP (NNP James) (NNP Dean))
  (VP (VBD did) (RB n't)
    (VP (VB dance)
      (PP (IN without)
        (NP (NNS pants)))))
  (. .))
```

parse for h

Figure 7.2: Syntactic parses for the p and h sentences of the James Dean example.

	edit	lexical features	entrel	atomic mono	atomic entrel	cum. join
1	DEL(*blue*)		$\sqsubset$	UP	$\sqsubset$	$\sqsubset$
2	SUB(*Jimmy Dean*, *James Dean*)	strsim:0.67	$\equiv$	UP	$\equiv$	$\sqsubset$
3	SUB(*move*, *dance*)	hyponym	$\sqsupset$	DOWN	$\sqsubset$	$\sqsubset$
4	EQ(*without*, *without*)		$\equiv$	DOWN	$\equiv$	$\sqsubset$
5	SUB(*jeans*, *pants*)	hypernym	$\sqsubset$	UP	$\sqsubset$	$\sqsubset$
6	DEL(*refused to*)	implic:+/∘	|	UP	|	|
7	INS(*n't*)	cat:neg	^	UP	^	$\sqsubset$
8	INS(*did*)	cat:aux	$\equiv$	DOWN	$\equiv$	$\sqsubset$

Table 7.2: An example of the operation of the NatLog model.

The best alignment for this example is fairly easy to identify, and the edits it contains appear as the first column table 7.2. Note, however, that these edits have been ordered according to the heuristics described in section 7.3: DELs first, then SUBs, then edits which involve operators with non-default projectivity, and INSs last.

The second column of table 7.2 shows features generated by the lexical entailment model for each edit. For compactness, we display at most one (especially salient) feature per edit. *Jimmy Dean* and *James Dean* have comparatively high string similarity. We use WordNet to identify the other two SUB edits as hyponym and hypernym substitutions. The `miscDeln` feature identifies *refuse to* as an implicative with signature +/∘, and the final INS edits are distinguished by their lexical categories.

The third column of table 7.2 shows the lexical entailment relation predicted for each edit. A generic deletion (edit 1) generates the $\sqsubset$ relation. A substitution with high string similarity (edit 2) leads to the $\equiv$ relation, as do a match (edit 4), and the insertion of a semantically vacuous auxiliary (edit 8). The hyponym (edit 3) and hypernym (edit 5) substitutions generate $\sqsupset$ and $\sqsubset$, respectively. The implicative deletion (edit 6) generates $|$, according to its implication signature, and the insertion of negation (edit 7) generates the $\wedge$ relation.

The fourth column of table 7.2 shows the effective monotonicity at the locus of each edit. (We present monotonicity classes, rather than the more general projectivity signatures, for simplicity and compactness.) The effective monotonicity for edits 1 through 6 is computed from the syntactic parse for p, while the effective monotonicity for edits 7 and 8 is computed from the syntactic parse for h. In p, *refuse to* is identified as a downward-monotone operator whose scope is its sentential complement, and *without* is identified as a downward-monotone operator whose scope is its argument noun phrase. Consequently, edits 1 and 5 occur in an upward-monotone context (since they fall within the scope of two downward-monotone operators), while edits 3 and 4 occur in a downward-monotone context (since they fall within the scope of just one downward-monotone operator), and edit 2 occurs in an upward-monotone context (since it doesn't fall within the scope of any downward-monotone operators). Meanwhile, in h, *not* is identified as a downward-monotone operator whose scope is the sentence in which it appears. Consequently, edit 8 occurs in a downward-monotone context (since it falls within the scope of a downward-monotone operator), while edit 7 occurs in an upward-monotone context (since it doesn't fall within the scope of any downward-monotone operators).

The fifth column of table 7.2 shows how the lexical entailment relations (from the third column) are projected into atomic entailment relations, according to the effective monotonicity shown in the fourth column. The only noteworthy case is edit 3, where the lexical entailment relation $\sqsupset$ is projected into atomic entailment relation $\sqsubset$ by a downward-monotone context.

The sixth and final column of table 7.2 shows the cumulative join of the atomic entailment relations in the fifth column. Initially, these are unremarkable: $\sqsubset$ joined

with either $\equiv$ or $\sqsubset$ yields $\sqsubset$. But at edit 6, we find that $\sqsubset$ joined with | yields |, and at edit 7, we find that | joined with $^\wedge$ yields $\sqsubset$ again. The final entailment relation in this line, $\sqsubset$, is NatLog's final (and correct) answer for our example problem.

7.8 Evaluation on the FraCaS test suite

The FraCaS test suite (Cooper et al. 1996) of NLI problems was one product of the FraCaS Consortium, a large collaboration in the mid-1990s aimed at developing a range of resources related to computational semantics. The FraCaS problems contain comparatively simple sentences, and the premise and hypothesis sentences are usually quite similar, so that just a few edits suffice to transform p into h. Despite this simplicity, the problems are designed to reflect a broad diversity of semantic and inferential phenomena. For this reason, the FraCaS test suite has proven to be invaluable as a developmental test bed for the NatLog system and as a yardstick for evaluating its effectiveness. Indeed, the test suite was created with just such an application as its primary goal. As the authors write:

> In light of the view expressed elsewhere in this and other FraCaS deliverables ... that inferential ability is not only a central manifestation of semantic competence but is in fact centrally constitutive of it, it shouldn't be a surprise that we regard inferencing tasks as the best way of testing an NLP system's semantic capacity.[2]

Despite having been designed expressly for the purpose of evaluating NLI systems, the FraCaS test suite went largely unexploited for more than a decade. In fact, to our knowledge, we are the first to have undertaken a quantitative system evaluation using the complete FraCaS data set.[3]

[2] Cooper et al. (1996), p. 63.

[3] Sukkarieh (2003) uses a handful of FraCaS examples in a manual evaluation of her McLogic formalism for knowledge representation and inference, but does not undertake a complete, quantitative, or automatic evaluation.

7.8.1 Characteristics of the FraCaS test suite

The FraCaS test suite contains 346 NLI problems, each consisting of one or more premise sentences, (usually) followed by a question and an answer. Figures 1.1 and 1.2 show a representative selection of FraCaS problems.

Questions and hypotheses. While the original FraCaS test suite expresses the "goal" of each problem as a question, the standard formulation of the NLI task involves determining the entailment relation between a premise and a declarative hypothesis. Thus, for the purpose of this work, we converted each FraCaS question into a declarative hypothesis, using an automatic tool which first generates a syntactic parse of the question and then applies a few simple tree-transformation rules. The results were then manually reviewed for grammaticality. Very few corrections were needed, mostly involving replacing negative-polarity items such as *any*. (For example, the question *Did any accountant attend the meeting?* was automatically transformed into the hypothesis *Any accountant attended the meeting*. *Any* was then manually changed to *Some*.)

Answer types. Most FraCaS problems are labeled with one of three answers: YES, NO, or UNK.[4] This three-way formulation of the NLI task was introduced in section 5.2.2, where we identified the YES label with the ENTAILMENT relation, the NO label with the CONTRADICTION relation, and the UNK label with the COMPATIBILITY relation. We can also identify each of these three labels with a union of relations in $\mathfrak{B}$, as described in section 5.6.3: YES corresponds to $\bigcup\{\equiv, \sqsubset\}$, NO corresponds to $\bigcup\{^\wedge, |\}$, and UNK corresponds to $\bigcup\{\sqsupset, \smile, \#\}$. The distribution of answers is *not* balanced: about 59% of the problems have answer YES, while 28% have answer UNK,

[4] Actually, the answers are not completely standardized, and quite a few contain qualifiers, such as, "Yes, on one possible reading". However, in most cases, the given answer can be straightforwardly identified with one of the values YES, NO, or UNK. But 12 of the 346 problems (about 3%) do not fit neatly into this scheme. Four problems lack questions and answers altogether, and in the remaining eight problems, the answer is too ambiguous or complex to be clearly identified with one of the three labels. Examples of such answers include "Yes on one scoping; unknown on another scoping", "Not many", and "At most two". These 12 problems were omitted from the evaluation described in section 7.8.2.

and 10% have answer NO. Consequently, a most-common-class classifier can achieve respectable accuracy simply by guessing YES on every problem.

Multiple-premise problems. Of the 346 FraCaS problems, 154 (about 45%) contain multiple premises: 122 problems contain two premises, 29 problems contain three premises, two problems contain four premises, and one problem contains five premises. These multiple-premise problems were excluded from the evaluation described in section 7.8.2. As we noted in section 6.4, an important limitation of the inference method implemented by NatLog is that, unlike first-order logic, it provides no general mechanism for combining information from multiple premises. However, this limitation is not unique to NatLog: it is faced by all other NLI systems of which we are aware. To be sure, most NLI systems—including NatLog—can accept a premise which contains multiple *sentences*. Indeed, recent RTE data sets have included an ever-greater proportion of problems with multi-sentence premises. But, for the most part, solving such problems requires nothing more than extracting information from a single sentence. By contrast, the inference problems shown in figure 1.2 cannot be solved without combining information from multiple sentences. Because NatLog relies on analyzing entailment relations across a single chain of atomic edits, it cannot effectively handle such problems. But nor can other systems developed for the RTE challenge.

Sections. The FraCaS test suite is divided into nine sections, each focused on a specific category of semantic phenomena: (1) quantifiers, (2) plurals, (3) anaphora, (4) ellipsis, (5) adjectives, (6) comparatives, (7) temporal reference, (8) verbs, and (9) attitudes. Of course, some of these sections will be more amenable than others to the natural logic approach: we hope to do well with quantifiers, but expect to make little headway with ellipsis, anaphora, or temporal reference.

7.8.2 Experiments and results

We evaluated the NatLog system on a subset of the FraCaS data containing 183 single-premise inference problems (representing about 53% of the complete FraCaS data set). The test set excluded multiple-premise problems (for reasons discussed in

System	P %	R %	Acc %
baseline: most common class	55.7	100.0	55.7
bag of words	59.7	87.2	57.4
NatLog 2007	68.9	60.8	59.6
NatLog 2008	89.3	65.7	**70.5**

Table 7.3: Performance of various systems on 183 single-premise FraCaS problems (three-way classification). The columns show precision and recall for the YES class, and accuracy.

section 7.8.1) and a handful of other problems which lack a well-defined answer (see footnote 4).

The NatLog system depends on alignments (that is, edit sequences connecting p and h) from an outside source. For these experiments, we generated alignments automatically using a very simple dynamic programming algorithm similar to the Levenshtein string edit distance algorithm (Levenshtein 1966), applied at the token level. The results from this automatic alignment were then manually reviewed and corrected.

Table 7.3 shows the results of the evaluation, along with some comparisons. As a baseline, we show results for a most-common-class classifier, which achieves perfect recall for the YES class (because it always guesses YES), but mediocre precision and accuracy (equal to the proportion of YES answers in the test set). A slightly stronger comparison is provided by a bag-of-words model like the one described in chapter 2. Relative to the most-common-class classifier, this model achieves lower recall, but slightly better precision and accuracy. Table 7.3 also shows results from an evaluation of an earlier version of the NatLog system, reported in (MacCartney and Manning 2007).

The current (2008) version of the NatLog system achieves overall accuracy of over 70%, representing a 27% reduction in error from the 2007 version and a 33% reduction in error from the baseline. A particularly noteworthy and gratifying result is the high figure for precision: over 89% for the current system. Errors in which NatLog wrongly predicts YES have fallen 66% from the 2007 version and 76% from the baseline.

§	Section	#	P %	R %	Acc %
1	Quantifiers	44	95.2	100.0	**97.7**
2	Plurals	24	90.0	64.3	75.0
3	Anaphora	6	100.0	60.0	50.0
4	Ellipsis	25	100.0	5.3	24.0
5	Adjectives	15	71.4	83.3	80.0
6	Comparatives	16	88.9	88.9	81.3
7	Temporal reference	36	85.7	70.6	58.3
8	Verbs	8	80.0	66.7	62.5
9	Attitudes	9	100.0	83.3	88.9
Sections 1, 2, 5, 6, 9		108	90.4	85.5	**87.0**
All sections		183	89.3	65.7	70.5

Table 7.4: Performance of NatLog on 183 single-premise FraCaS problems, broken out by section. The columns show the number of problems, precision and recall for the YES class, and accuracy.

Not surprisingly, NatLog's performance varies considerably over the different sections of the FraCaS test set. Table 7.4 shows results broken down by section. In the section concerning quantifiers, which is both the largest and the most amenable to natural logic, all problems but one are answered correctly.[5] We also answer all problems but one correctly in the (admittedly small) section on attitudes, which involves implicatives and factives. As we anticipated, performance is mediocre in four sections concerning semantic phenomena (anaphora, ellipsis, temporal reference, and verbs) not relevant to natural logic and not modeled by the system. But in the other five sections (covering about 60% of the problems), we achieve accuracy of 87%, a reduction in error of 61% from the 2007 version of the system. What's more, precision is high in nearly every section: even outside its areas of expertise, the system rarely predicts entailment when none exists.

[5]In fact, the sole exception is disputable, since it hinges on whether *many* refers to proportion (apparently, the view held by the FraCaS authors) or absolute quantity.

		guess			
		YES	NO	UNK	total
gold	YES	67	4	31	102
	NO	1	16	4	21
	UNK	7	7	46	60
	total	75	27	81	183

Table 7.5: Confusion matrix for NatLog on the FraCaS test suite (all sections).

7.8.3 Discussion

The confusion matrix shown in table 7.5 reveals an interesting property of the NatLog system. By far the largest category of confusions are those where the answer is YES but we guess UNK. This reflects both the bias toward YES in the FraCaS data, and the system's tendency to predict UNK (entailment relation #) when confused: given the rules for joining entailment relations, the system can predict YES only if all atomic-level predictions are either $\sqsubset$ or $\equiv$.

In fact, almost two-thirds of all errors made by NatLog are cases where NatLog wrongly predicted UNK. As we noted in section 5.6.3, joining a chain of predicted atomic entailment relations will tend toward a less-informative result (namely, #) if the chain contains any "noise" in the form of mistaken predictions. Consequently, NatLog tends to predict # whenever it runs into trouble, and in this sense, UNK is NatLog's default response.

It may be instructive to examine the remaining categories of errors more closely. The examples referred to in the following text are shown in figure 7.3.

Gold UNK, guess NO. There were seven errors in this category, and all involved cases where a lexical substitution generated the | relation, which was (wrongly) projected to the top level without change (and any other edits generated $\equiv$ or $\sqsubset$). Problem 277 is representative of these mistakes. (Three other cases are highly similar to this one, while the remainder are broadly similarly.) The edit SUB(*1991*, *1992*) is correctly predicted to generate lexical entailment relation |; however, NatLog acts as

§1: Quantifiers			
56	p	Many British delegates obtained interesting results from the survey.	
	h	Many delegates obtained interesting results from the survey.	UNK
§2: Plurals			
109	p	Just one accountant attended the meeting.	
	h	Some accountants attended the meeting.	NO
§4: Ellipsis			
176	p	John said Mary wrote a report, and Bill did too.	
	h	John said Bill wrote a report.	YES
§5: Adjectives			
217	p	John is a cleverer politician than Bill.	
	h	John is cleverer than Bill.	UNK
§7: Temporal reference			
277	p	Smith lived in Birmingham in 1991.	
	h	Smith lived in Birmingham in 1992.	UNK
304	p	Smith wrote a report for two hours.	
	h	Smith wrote a report.	UNK

Figure 7.3: Examples of errors made by NatLog on the FraCaS test suite.

if the verb *lived* projects | (from a temporal modifier) without change, when in fact it should project | as #. Note the discrepancy between implementation and theory: in section 6.2.5, we noted explicitly that most verbs project | as #. True, we also observed there that verbal constructions which encode functional relations, such as *was born in*, do project | as |; however, *lived* is not such a verb. This type of error could be corrected without much difficulty.

Gold UNK, guess YES. Each of the seven errors in this category involved a deletion edit for which NatLog rightly predicted lexical entailment relation $\sqsubset$, but wrongly assumed that this relation would be projected without change to the top level. The sole error made in the section on quantifiers (problem 56) is disputable, since it hinges on whether the quantifier *many* refers to proportion (and thus is non-monotone in its first argument—apparently the view held by the FraCaS authors), or to absolute quantity (and thus is upward-monotone in its first argument). In this author's view, the latter reading is more intuitive, and NatLog's answer of YES is in fact correct. The remaining errors in this category involved adjectives, comparatives, and temporal reference, and are typified by problems 217 and 304.

Gold YES, guess NO. All four errors in this category involve ellipsis, and all are more-or-less hopeless cases for NatLog. Problem 176 is a typical example. The alignment used by NatLog in this problem includes the deletion of the second (elliptical) clause in p, and straightforward linear alignment of the first clause in p to h, including the edit SUB(*Mary*, *Bill*). For this edit, NatLog (rightly) predicts the lexical entailment relation |, and consequently produces answer NO for the problem. While there is an alignment which could have led NatLog to the correct answer (namely, the one which matches each token in h to the equal token in p, and deletes all other tokens in p), it is unreasonable to expect an aligner to produce it, since that alignment appears preferable only to one who grasps the phenomenon of ellipsis.

Gold NO, guess YES. The sole error in this category was problem 109. NatLog predicts YES because it (rightly) assigns lexical entailment relation $\sqsubset$ to the edit

SUB(*one*, *Some*), and because it does not model the restrictive semantics of *Just*. Note that YES would be the correct answer if not for the *s* on *accountants* in *h*; thus, a further reason for NatLog's mistake is that it systematically (and intentionally) ignores morphology.

Since the NatLog system was developed with FraCaS problems in mind, these results do not constitute a proper evaluation on unseen test data. On the other hand, the system does no training on FraCaS data, and has had no opportunity to learn its biases. (Otherwise, accuracy on §4 could not fall so far below the baseline.) The system not only answers most problems correctly, but usually does so for valid reasons, particular within its areas of expertise. All in all, the results fulfill our main goal in testing on FraCaS: to demonstrate the representational and inferential adequacy of our model of natural logic.

7.9 Evaluation on the RTE test suite

Because the FraCaS data set is comparatively small, and because it has not been used to evaluate other NLI systems, making comparisons difficult, we chose also to evaluate NatLog using a larger, better known collection of NLI problems from the PASCAL RTE Challenge, which was introduced in section 1.3.2.

7.9.1 Characteristics of the RTE test suite

The RTE test suites differ from the FraCaS test suite in several important and relevant ways. (See figure 7.4 for some example problems.) First, the RTE data sets are much larger. Second, the goal of RTE is binary classification (as in section 5.2.1), rather than three-way classification. Third, instead of textbook examples of semantic phenomena, RTE problems are more natural-seeming, and the premises are much longer than in FraCaS.

Due to the character of RTE problems, we do not expect NatLog to be a good general-purpose solution to solving all RTE problems. One reason is that most RTE problems depend on forms of inference, such as paraphrase, temporal reasoning, or

71	p	As leaders gather in Argentina ahead of this weekends regional talks, Hugo Chávez, Venezuela's populist president is using an energy windfall to win friends and promote his vision of 21st-century socialism.	
	h	Hugo Chávez acts as Venezuela's president.	YES
85	p	Mr. Fitzgerald revealed he was one of several top officials who told Mr. Libby in June 2003 that Valerie Plame, wife of the former ambassador Joseph Wilson, worked for the CIA.	
	h	Joseph Wilson worked for CIA.	NO
788	p	Democrat members of the Ways and Means Committee, where tax bills are written and advanced, do not have strong small business voting records.	
	h	Democrat members had strong small business voting records.	NO

Figure 7.4: Illustrative examples from the RTE3 development set.

relation extraction, which NatLog is not designed to address. Another reason is that in most RTE problems, the edit distance between p and h is relatively large. More atomic edits means a greater chance that errors made in lexical entailment classification or entailment projection will propagate, via entailment joining, to the system's final output. (Given the rules for entailment joining, the system can answer YES to an RTE problem only if all atomic-level predictions are either $\sqsubset$ or $\equiv$.) Rather, in applying NatLog to RTE, we hope to make reliable predictions on a subset of RTE problems, trading recall for precision. If we succeed, then we may be able to hybridize with a broad-coverage RTE system to obtain better results than either system individually—the same strategy that was adopted by Bos and Markert (2006) for their FOL-based system.

7.9.2 Experiments and results

In order to evaluate NatLog's effectiveness on RTE problems, we used data from the RTE3 test suite. In these experiments, we used alignments generated by the alignment component of the Stanford RTE system (chapter 4) as input to our entailment model.

System	Data	% Yes	P %	R %	Acc %
NatLog	dev	22.5	**73.9**	32.4	59.3
	test	26.4	**70.1**	36.1	59.4
Stanford	dev	50.3	68.7	67.0	67.3
	test	50.0	61.8	60.2	60.5
Hybrid, balanced	dev	50.0	70.3	68.2	68.8
	test	50.0	65.5	63.9	64.3
Hybrid, optimized	dev	56.0	69.2	75.2	**70.0**
	test	54.5	64.5	68.5	**64.5**

Table 7.6: Performance of various systems on the RTE3 test suite (two-way classification). The columns show the data set used (development or test, 800 problems each), the proportion of YES predictions, precision and recall for the YES class, and accuracy.

Recall that in the Stanford system, an alignment is a map from h words to p words. An h word can be aligned to any p word, regardless of position; multiple h words can be aligned to the same p word; and there is no notion of phrase alignments. When we translate such alignments into the NatLog representation described in section 7.3, each pair of aligned words generates a SUB edit (or, if the words are identical, an EQ edit). Unaligned p words yield DEL edits, while unaligned h words yield INS edits. Where possible, contiguous sequences of word-level edits are then collected into equivalent phrase edits. While the result of this translation method cannot be interpreted as a conventional edit script (there is no well-defined ordering of edits, and multiple edits can operate on the same input phrases), we find that this poses no great impediment to subsequent processing by the entailment model.

The first rows of table 7.6 show the performance of NatLog on RTE3 development and test sets, as well as the performance of the Stanford RTE system. The overall accuracy of the NatLog system is not particularly impressive (below 60%), mainly due to the fact that most RTE problems involve types of inference where natural logic is of little help. However, relative to the Stanford system, NatLog achieves high precision on its YES predictions—above 70%—suggesting that combining the NatLog

and Stanford systems using a hybridization strategy may be effective. For comparison, the FOL-based system of Bos and Markert (2006) attained a similarly high precision of 76% on RTE2 problems, but was able to make a positive prediction in only about 4% of cases. NatLog makes positive predictions far more often—in about 25% of cases. Thus, NatLog achieves six times the coverage of the Bos & Markert system, with nearly the same precision.

The Stanford system makes YES/NO predictions by thresholding a real-valued *inference score*. To construct a hybrid system, we adjust the Stanford inference scores by $+\Delta$ or $-\Delta$, depending on whether or not NatLog predicts YES. We choose Δ by optimizing development set accuracy, while adjusting the threshold to generate balanced predictions (that is, equal numbers of YES and NO predictions). As an additional experiment, we fix Δ at this value and then adjust the threshold to optimize development set accuracy, resulting in an excess of YES predictions. (Since this optimization is based solely on development data, its use on test data is fully legitimate.) Results for these two cases are shown in the last rows of table 7.6. The parameters tuned on development data were found to yield good performance on test data. The optimized hybrid system attained an absolute accuracy gain of 4% over the Stanford system, corresponding to an extra 32 problems answered correctly. This result is statistically significant ($p < 0.05$, McNemar's test, 2-tailed).

7.9.3 Discussion

The gains attributable to NatLog are exemplified by problem 788 (figure 7.4). NatLog sanctions the deletion of a restrictive modifier and an appositive from the premise, and recognizes that deleting a negation generates a contradiction; thus it correctly answers NO. On the other hand, there are many RTE problems where NatLog's precision works against it. For example, NatLog answers NO to problem 71 because it cannot account for the insertion of *acts as* in the hypothesis. Fortunately, both the Stanford system and the hybrid system answer this problem correctly.

Chapter 8

Conclusions

In this dissertation, we have explored a range of approaches to the problem of natural language inference (NLI): first, the robust, but imprecise, bag-of-words model (chapter 2), which predicts inferential validity via approximate matching of lexical content; next, the Stanford RTE system (chapter 4), which preserves the approximate matching strategy, but seeks to add a bit more precision; and finally, natural logic (chapters 5 and 6) and the NatLog system (chapter 7), which aims to address the subset of NLI problems requiring substantially greater semantic precision. We have also examined the problem of alignment for NLI (chapter 3), which plays a role in each of these approaches. What have we learned about the problem of NLI, and what are the most promising directions for future research in this area? This final chapter seeks to address these questions.

8.1 Contributions of the dissertation

This dissertation has examined the problem of natural language inference from a number of angles, and we hope to have made important contributions in several different areas.

In chapter 2, we developed a bag-of-words model for NLI and evaluated it on several RTE problem sets. While the basic form of the model is not novel, the specific details of its construction are original. Perhaps our most important finding

was the wide variation in the intrinsic difficulty of the various RTE problem sets, as measured by the yardstick of this very simple model.

In chapter 3, we undertook the first systematic investigation of the problem of alignment for NLI. We examined the relation between alignment in NLI and in machine translation (MT), and presented a number of arguments for the unsuitability of MT aligners to the NLI alignment task. We made the first comparative evaluation of bag-of-words, MT, and NLI aligners on an NLI alignment task. And, we proposed a new model of alignment for NLI—the MANLI system—and showed that it significantly outperforms rival approaches, by exploiting external lexical resources in a supervised learning paradigm, by including contextual features to aid in aligning function words, and by allowing multi-word phrases to be aligned as units.

In chapter 4, we introduced the Stanford RTE system, which was among the first NLI systems to make a clear separation between alignment and entailment determination. This crucial innovation enables the Stanford RTE system to make predictions of inferential validity which are based, not merely on the degree of alignment, but also on global features of the aligned $\langle p, h \rangle$ pair motivated by semantic theory. The Stanford RTE system is thereby able to attain substantially greater precision than simple overlap models such as the bag-of-words model presented in chapter 2.

Seeking still greater precision, we then presented the most important contribution of the dissertation: a new model of natural logic which extends the monotonicity calculus of van Benthem and Sánchez-Valencia to incorporate semantic exclusion and implicativity. We began in chapter 5 by defining an expressive inventory of entailment relations—the set $\mathfrak{B}$ of seven basic entailment relations—which includes representations of both semantic containment and semantic exclusion. We showed that the relations in $\mathfrak{B}$ are both mutually exclusive and (if vacuous expressions are excluded) mutually exhaustive, so that every pair of (non-vacuous) expressions can be assigned to some relation in $\mathfrak{B}$. We also described the algebra of joins for relations in $\mathfrak{B}$, and explained how to compute such joins automatically.

In chapter 6, we introduced the concept of projectivity signatures, which generalizes the concept of monotonicity classes to cover the exclusion relations. We then

described a general method for determining the entailment relation between two sentences p and h, by (1) finding a sequence of atomic edits which transforms p into h; (2) predicting a lexical entailment relation for each edit; (3) propagating the lexical entailment relation generated by each edit upward through the semantic composition tree of the sentence to which the edit is applied, according to the projectivity signatures of intermediate nodes; and (4) joining the resulting entailment relations across the edit sequence. And, we showed how to provide a unified account of inferences involving implicatives and non-factives under the same framework.

Finally, in chapter 7, we described the NatLog system, the first robust, general-purpose system for natural logic inference over real English sentences. NatLog contains two key innovations: (1) it learns to predict lexical entailment relations by using machine learning techniques and exploiting a variety of manually and automatically constructed sources of information on lexical relations; and (2) it identifies expressions with non-default projectivity and computes the likely extent of their arguments in a syntactic parse using hand-crafted tree patterns. We demonstrated the practical value of the NatLog system in evaluations on the FraCaS and RTE test suites. On the FraCaS test suite, we showed NatLog attained high accuracy on most sections (excluding sections not relevant to natural logic, such as ellipsis), and high precision on all sections. On the RTE test suite, we showed that adding NatLog as a component in the broad-coverage Stanford RTE system led to substantial accuracy gains.

In addition to these specific contributions just outlined, we hope to have achieved something greater through this program of research: namely, to have shown that natural logic, far from being merely a formal plaything of logicians and semanticists, has relevance and applicability to the real-world challenges of open-domain NLI, and consequently constitutes a topic ripe for the attention of NLP researchers. And indeed, there are indications that our work in this area has begun to catalyze related work by other investigators. For example, Schoenmackers et al. (2008) built on our work on inferences involving monotonicity in their exploration of scaling textual inference to the web, while Danescu-Niculescu-Mizil et al. (2009) were motivated by our model of natural logic to seek methods for unsupervised discovery of downward-entailing operators.

8.2 The future of natural language inference

What does the future hold for research on natural language inference? One of the most important learnings to emerge from work in this area is that the kinds of inference covered by the NLI task are too diverse for any single approach to provide a complete solution. Consider, for example, the following inference:

> p *The First Family's flight to Paris left Washington at 11am and lasted six hours.*
>
> h *The First Lady was not in France at 2pm.*

While h is not a strict logical consequence of p, most people would not hesitate to accept that h follows from p, and thus the inference is valid by the definition of the NLI task. And the reasoning involved is straightforward—one need not possess the deductive genius of Sherlock Holmes to arrive at h. And yet recognizing the validity of the inference depends on several different kinds of expertise: first, the knowledge that the First Lady is a member of the First Family, and that Paris is in France; next, the capacity to handle inferences involving containment and negation; moreover, the ability to perform simple clock arithmetic; and finally, an aptitude for basic spatial reasoning. In short, there is no silver bullet.

So, going forward, we'll need to look at ways to combine different kinds of reasoners, with different areas of expertise—including not only models based on lexical overlap (such as the bag-of-words model presented in chapter 2) and models of natural logic (like NatLog), but also reasoners with specialized expertise in temporal, spatial, and mathematical reasoning; systems which excel in relation extraction; and systems with a facility for commonsense reasoning. Now, some of these components are closer to reality than others, but even if we were satisfied with each of the individual components, the key question is, how can we combine them to best advantage? Should we apply each one independently, and then aggregate their results? What sort of aggregation function should we use? Or, can we devise an architecture in which these heterogeneous reasoners collaborate in a more fine-grained way, each contributing individual steps to an overall proof? These difficult questions will undoubtedly keep the NLI research community busy for many years.

Bibliography

Adams, Rod. 2006. Textual Entailment Through Extended Lexical Overlap. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Akhmatova, Elena. 2005. Textual entailment resolution via atomic propositions. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Bar-Haim, Roy, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pp. 1–9.

Bar-Haim, Roy, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic Inference at the Lexical-Syntactic Level. In *Proceedings of AAAI-07*.

Bos, Johan, and Katja Markert. 2005a. Combining shallow and deep NLP methods for recognizing textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, pp. 65–68.

Bos, Johan, and Katja Markert. 2005b. Recognising Textual Entailment with Logical Inference. In *Proceedings of EMNLP-05*.

Bos, Johan, and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Böttner, Michael. 1988. A note on existential import. *Studia Logica* 47(1):35–40.

Brockett, Chris. 2007. Aligning the RTE 2006 Corpus. Technical Report MSR-TR-2007-77, Microsoft Research. URL: `ftp://ftp.research.microsoft.com/pub/tr/TR-2007-77.pdf`.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics* 19(2):263–311.

Callison-Burch, Chris, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of EACL-06*, pp. 249–256.

Chambers, Nathanael, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning Alignments and Leveraging Natural Logic. In *ACL-07 Workshop on Textual Entailment and Paraphrasing*.

Chomsky, Noam. 1975. Questions of form and interpretation. *Linguistic Analysis* 1: 75–109.

Collins, Michael. 2002. Discriminative titleing methods for hidden Markov models. In *Proceedings of EMNLP-02*, pp. 1–8.

Collins, Michael. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics* 29(4):589–637.

Condoravdi, Cleo, Dick Crouch, Valeria de Paiva, Reinhard Stolle, and Daniel G. Bobrow. 2003. Entailment, Intensionality and Text Understanding. *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning* pp. 38–45.

Cooper, Robin, et al. 1996. Using the framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium. URL: `http://www.cogsci.ed.ac.uk/~fracas/`.

Crouch, Dick, Roser Saurí, and Abraham Fowler. 2005. AQUAINT Pilot Knowledge-Based Evaluation: Annotation Guidelines. Technical report, Palo Alto Research Center. URL: `http://www2.parc.com/istl/groups/nltt/papers/aquaint_kb_pilot_evaluation_guide.pdf`.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Danescu-Niculescu-Mizil, Cristian, Lillian Lee, and Richard Ducott. 2009. Without a 'doubt'? Unsupervised discovery of downward-entailing operators. In *Proceedings of NAACL-HL-09*.

Dang, Hoa Trang, and Danilo Giampiccolo. 2008. The TAC 2008 Recognizing Textual Entailment (RTE) Track. URL: `http://www.nist.gov/tac/tracks/2008/rte/`.

Das, Dipanjan, and André F. T. Martins. 2007. A Survey on Automatic Text Summarization. URL: `http://www.cs.cmu.edu/~nasmith/LS2/das-martins.07.pdf`.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-2006*.

de Marneffe, Marie-Catherine, Sebastian Padó, and Christopher D. Manning. 2009. Multi-word expressions in textual entailment: Much ado about nothing? In *Proceedings of the ACL/IJCNLP 2009 Workshop on Applied Textual Inference (TextInfer 2009)*.

de Marneffe, Marie-Catherine, Anna Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08*.

de Salvo Braz, Rodrigo, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment and question-answering. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pp. 1678–1679.

DeNero, John, Dan Gillick, James Zhang, and Dan Klein. 2006. Why Generative Phrase Models Underperform Surface Heuristics. In *Proceedings of the ACL-06 Workshop on Statistical Machine Translation*, pp. 31–38.

Dolan, Bill, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora. In *Proceedings of COLING-04*.

Fellbaum, Christiane, et al. 1998. *WordNet: an electronic lexical database*. Cambridge, Mass: MIT Press.

Finkel, Jenny Rose, Trond Grenager, and Christopher D. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of ACL-05*, pp. 363–370.

Fowler, Abraham, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. 2005. Applying COGEX to recognize textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Fraser, Alexander, and Daniel Marcu. 2007. Measuring Word Alignment Quality for Statistical Machine Translation. *Computational Linguistics* 33(3):293–303.

Fyodorov, Yaroslav, Yoad Winter, and Nissim Francez. 2000. A Natural Logic Inference System. In *Proceedings of the 2nd Workshop on Inference in Computational Semantics (ICoS-2)*.

Giampiccolo, Danilo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2008. The Fourth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the TAC-08 Text Analysis Conference*.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-07 Workshop on Textual Entailment and Paraphrasing*.

Glickman, Oren, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Grice, H. Paul. 1975. Logic and conversation. *Syntax and Semantics* 3:41–58.

Haghighi, Aria, Andrew Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-05)*.

Harabagiu, Sanda, and Andrew Hickl. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of ACL-06*, volume 44, p. 905.

Harabagiu, Sanda, Andrew Hickl, and Finley Lacatusu. 2006. Negation, Contrast, and Contradiction in Text Processing. *Proceedings of AAAI-06*.

Harmeling, Stefan. 2007. An extensible probabilistic transformation-based approach to the Third Recognizing Textual Entailment Challenge. In *ACL-07 Workshop on Textual Entailment and Paraphrasing*.

Hickl, Andrew, and Jeremy Bensley. 2007. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In *ACL-07 Workshop on Textual Entailment and Paraphrasing*.

Hickl, Andrew, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCC's GROUNDHOG System. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Hobbs, Jerry R., Mark Stickel, Paul Martin, and Douglas D. Edwards. 1988. Interpretation as abduction. In *Proceedings of ACL-88*, pp. 95–103.

Jiang, Jay J., and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*.

Jijkoun, Valentin, and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, pp. 73–76.

Karttunen, Lauri. 1971. Implicative verbs. *Language* 47(2):340–358.

Klein, Dan, and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-03*.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL-07, demonstration session*.

Lacatusu, Finley, Andrew Hickl, Kirk Roberts, Ying Shi, Jeremy Bensley, Bryan Rink, Patrick Wang, and Lara Taylor. 2006. LCC's GISTexter at DUC 2006: Multi-Strategy Multi-Document Summarization. In *Proceedings of DUC-06*.

Lakoff, George. 1970. Linguistics and natural logic. *Synthese* 22:151–271.

Landauer, Thomas K., and Susan T. Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction and representation of knowledge. *Psychological Review* 104(2):211–240.

Levenshtein, Vladimir I. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10:707.

Levy, Roger, and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC-06*. URL: `http://nlp.stanford.edu/software/tregex.shtml`.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of NAACL-06*. URL: `http://www.cs.berkeley.edu/~pliang/papers/alignment-naacl2006.pdf`.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pp. 768–774.

MacCartney, Bill, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP-08*.

MacCartney, Bill, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to Recognize Features of Valid Textual Entailments. In *Proceedings of NAACL-06*.

MacCartney, Bill, and Christopher D. Manning. 2007. Natural logic for textual inference. In *ACL-07 Workshop on Textual Entailment and Paraphrasing*.

MacCartney, Bill, and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling-08)*.

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

Malakasiotis, Prodromos, and Ion Androutsopoulos. 2007. Learning Textual Entailment using SVMs and String Similarity Measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 42–47.

Marcu, Daniel, and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP-02*, pp. 133–139.

Marsi, Erwin, and Emiel Krahmer. 2005. Classification of semantic relations by humans and machines. In *ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

McCallum, Andrew, and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL-2003*.

Meyers, Adam, et al. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pp. 24–31.

Minnen, Guido, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering* 7(3):207–223.

Moldovan, Dan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. COGEX: A logic prover for question answering. In *Proceedings of NAACL-2003*.

Nairn, Rowan, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of ICoS-5 (Inference in Computational Semantics)*.

Och, Franz Josef, and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1):19–51.

Padó, Sebastian, Marie-Catherine de Marneffe, Bill MacCartney, Anna Rafferty, Eric Yeh, and Christopher D. Manning. 2008. Deciding entailment and contradiction with stochastic and edit distance-based alignment. In *Proceedings of the Text Analysis Conference (TAC-2008)*.

Padó, Sebastian, Michel Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Robust machine translation evaluation with entailment features. In *Proceedings of ACL-09*.

Raina, Rajat, Andrew Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*.

Ritter, Alan, Doug Downey, Stephen Soderland, and Oren Etzioni. 2008. It's a Contradiction—No, it's Not: A Case Study using Functional Relations. In *Proceedings of EMNLP-08*.

Romano, Lorenza, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL 2006*.

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

Sánchez Valencia, Victor. 1995. Parsing-driven inference: Natural logic. *Linguistic Analysis* 25:258–285.

Schoenmackers, Stefan, Oren Etzioni, and Daniel S. Weld. 2008. Scaling Textual Inference to the Web. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pp. 79–88.

Stalnaker, Robert. 1968. A theory of conditionals. *Studies in Logical Theory* 2: 98–122.

Stalnaker, Robert. 1992. Notes on conditional semantics. In *Proceedings of the 4th Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 316–327.

Sukkarieh, Jana Z. 2001. Quasi-NL Knowledge Representation for Structurally-Based Inferences. In *Proceedings of the 3rd Workshop on Inference in Computational Semantics (ICoS-3)*.

Sukkarieh, Jana Z. 2003. An expressive efficient representation: Bridging a gap between NLP and KR. In *Proc. of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, pp. 800–815.

Tatu, Marta, and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of HLT/EMNLP 2005*, pp. 371–378.

Tatu, Marta, and Dan Moldovan. 2007. COGEX at RTE3. In *Proceedings of ACL-07*.

van Benthem, Johan. 1988. The semantics of variety in categorial grammars. In J.V.B.W. Buszkowski and W. Marciszewski (eds.), *Categorial grammar*, pp. 33–55. John Benjamins.

van Benthem, Johan. 1991. Language in action: categories, lambdas and dynamic logic. *Studies in Logic* 130.

van Benthem, Johan. 2008. A brief history of natural logic. Technical Report PP-2008-05, Institute for Logic, Language & Computation. URL: `http://www.illc.uva.nl/Publications/ResearchReports/PP-2008-05.text.pdf`.

van der Sandt, Rob A. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics* 9(4):333–377. URL: `http://jos.oxfordjournals.org/cgi/content/abstract/9/4/333`.

van Eijck, Jan. 2005. Natural logic for natural language. URL: `http://homepages.cwi.nl/~jve/papers/05/nlnl/NLNL.pdf`.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING-96*, pp. 836–841.

Voorhees, Ellen. 2008. Contradictions and justifications: Extensions to the textual entailment task. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Witten, Ian H., and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*, 2nd edition. Morgan Kaufmann.

Zanzotto, F.M., A. Moschitti, M. Pennacchiotti, and M.T. Pazienza. 2006. Learning textual entailment from examples. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.