LEARNING OPEN DOMAIN KNOWLEDGE FROM TEXT

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

Gábor György Angeli
May 2016

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.

(Percy Liang)

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)

Approved for the Stanford University Committee on Graduate Studies

Preface

The increasing availability of large text corpora holds the promise of acquiring an unprecedented amount of knowledge from this text. However, current techniques are either specialized to particular domains or do not scale to large corpora. This dissertation develops a new technique for learning open-domain knowledge from unstructured web-scale text corpora.

A first application aims to capture common sense facts: given a candidate statement about the world and a large corpus of known facts, is the statement likely to be true? We appeal to a probabilistic relaxation of natural logic – a logic which uses the syntax of natural language as its logical formalism – to define a search problem from the query statement to its appropriate support in the knowledge base over valid (or approximately valid) logical inference steps. We show a 4x improvement in recall over lemmatized lookup for querying common sense facts, while maintaining above 90% precision.

This approach is extended to handle longer, more complex premises by segmenting these utterances into a set of atomic statements entailed through natural logic. We evaluate this system in isolation by using it as the main component in an Open Information Extraction system, and show that it achieves a 3% absolute improvement in F_1 compared to prior work on a competitive knowledge base population task.

A remaining challenge is elegantly handling cases where we could not find a supporting premise for our query. To address this, we create an analogue of an evaluation function in gameplaying search: a shallow lexical classifier is folded into the search program to serve as a heuristic function to assess how likely we would have been to find a premise. Results on answering 4^{th}grade science questions show that this method improves over both the classifier in isolation and a strong IR baseline, and outperforms prior work on the task.

Acknowledgements

Contents

List of Tables

List of Figures

Chapter 1

Introduction

At its core, machine-learning-driven natural language processing aims to imitate human intelligence by observing a human perform a given task repeatedly and training from this data. For example, in order to train a system to recognize whether a given word is the name of a person, we would first collect a large set of words, labeled as either people or not. A system would then take this data, and learn a *model* that can then predict, on unseen words, whether it is a person or not. The great advantage of this framework is that it frees us from having to have a deep understanding of the underlying process by which humans perform the target task, instead allowing us to observe examples and use this to learn to replicate the task. In fact, this has been responsible for much of the progress in natural language processing in the first two decades of the new millennium. Many of the core NLP tasks (named entity recognition, part of speech tagging, parsing, etc.) can now be done with high accuracy, and many of the higher-level tasks (relation extraction, sentiment analysis, question-answering, etc.) have matured to the point of being useful as off-the-shelf components both for academia and industry.

With these advances, I believe that we should turn back to a relatively neglected topic: how do we begin to create programs that exhibit *general purpose* intelligence? In many ways, data-driven natural language processing systems are idiot savants: these systems perform at impressive accuracies at very narrow tasks – the tasks they were trained to replicate – but are incapable of either generalizing across tasks, or performing complex common-sense inferences. For example, the following is a list of some questions which are

trivial for humans to answer, but are very difficult for a trained system without either (1) very specific, narrow, and deep training data, or (2) a very large amount of general-purpose background knowledge:

> ***I ate a bowl of soup, and put the bowl in the bathtub. Did it float?***
> Answering this question correctly requires not only a fairly complex bit of inference, but also a large amount of varied background knowledge: a bowl is concave, empty concave things float, if you eat soup the bowl becomes empty, bathtubs are full of water, etc.
>
> ***I left water in the freezer; what happened to it?***
> Here again, we need to know that freezers are cold (below freezing), that water turns to ice when it's below freezing, and that water turning to ice is more informative than the other things that also "happen" to it, such as it getting cold, or getting dark, or no longer sloshing, etc.
>
> ***The Congressman resigned to go back to governing his hometown. What is his new title?***
> To correctly answer "mayor," we would have to know that if someone resigns from a title, he no longer holds it, and that a mayor governs a town. Also, that a hometown is a city or other entity with a mayor – unlike, say, homework or downtown.

A central tenet of this dissertation is that, if we're in pursuit of general intelligence, we should be aiming to answer these sorts of questions not by collecting narrow deep training sets, but rather by developing techniques to collect large amounts of common-sense information at scale. This is, after all, the same common-sense information we leverage to solve these tasks ourselves. An additional axiom of the dissertation is that the most promising way to collect this sort of common-sense knowledge is from text. This is not not an uncontroversial axiom: much of what we know about the world we know from vision, or interactions with physical objects. Colors, shapes, and sizes of things for example are far more naturally extracted from visual data; the laws of physics are much more easily learned through interaction with the world.

Nonetheless, natural language is the de-facto standard for storing and transmitting information, and has the key advantages of being plentiful and convenient to work with. Textual data stores information about people and places (e.g., *Obama was born in Hawaii*), facts about science and engineering (e.g., *Ice is frozen water*), or simply common-sense facts (e.g., *Some mushrooms are poisonous*). It's also important to remember that with the internet, we have unprecedented access to a huge – and growing – amount of text.

The most natural solution to this problem of common-sense knowledge is to collect large manually-curated (or semi-curated) databases of these general purpose facts. Freebase [13] is a popular example of one such database; Cyc [63] is another famous hand-curated knowledge base. However, these manually created knowledge bases are both woefully incomplete and quickly become outdated. In medicine, half of the medical school curriculum becomes obsolete within 5 years of graduation,[1] requiring constant updating. MEDLINE counts 800,000 biomedical articles published in 2013 alone.[2] In academia, studies show that up to 90% of papers are never cited, suggesting that many are never read.

This dissertation presents an alternative: instead of storing knowledge in knowledge bases, let's continue to store knowledge directly in text. This allows us to make use of our large corpus of unstructured text directly, with perhaps minimal syntactic annotation, to reason about true and false facts. Storing information becomes as easy as storing text. But, this comes at a cost. The key challenge in this approach is the ability to use a large corpus of plain text to query facts and answer questions which are not verbatim expressed in the text. For example, a statement *the cat ate a mouse* should support even lexically dissimilar queries like *carnivores eat animals* and reject logically contradicted queries (like *no carnivores eat animals*). Or, it may be difficult to find the relevant nugget of information from a long and syntactically complex sentence from, e.g., Wikipedia. Therefore, the technical focus of this dissertation will be on how we run soft logical inference from a very large set of candidate premises (our source corpus) to either prove or disprove a candidate fact whose truth we do not know.

A natural formalism for addressing this challenge is *natural logic* – a proof theory

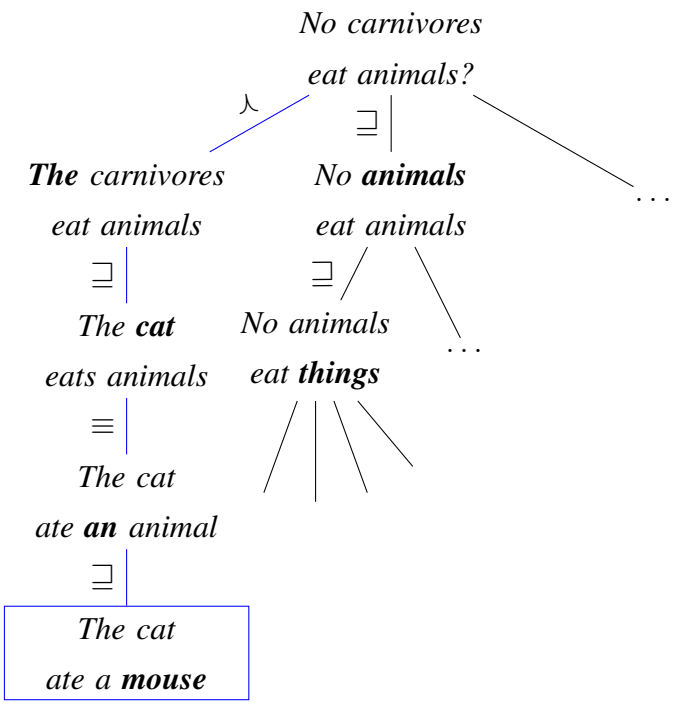
[1] `http://uvamagazine.org/articles/adjusting\_the\_prescription/`
[2] `https://www.nlm.nih.gov/bsd/medline\_cit\_counts\_yr\_pub.html`

over the syntax of natural language. The logic offers computational efficiency and eliminates the need for semantic parsing and domain-specific meaning representations, while still warranting most common language inferences (e.g., negation). Furthermore, the inferences warranted by the logic tend to be the same inferences that are cognitively easy for humans – that is, the inferences humans assume a reader will effortlessly make.

This dissertation explores how to leverage natural logic as a formalism for extracting knowledge not only when it is verbatim written in text, but also when it is only *implied* by some statement in the text and we must perform a large scale inference over a large set of candidate premises to find the right one (if any). In the subsequent chapters, we will review the theory behind natural logic (Chapter 3), and then describe a system to (1) extract common-sense knowledge from a large corpus of unannotated text via a search procedure over a soft relaxation of natural logic; (2) simplify complex syntactic structures into maximally informative atomic statements, and (3) incorporate an entailment classifier into this search to serve as an informed backoff.

In Chapter 4 we introduce our general framework for inferring the truth or falsehood of common-sense facts from a very large knowledge base of statements about the world. For example, if a premise *the cat ate a mouse* is present in the knowledge base, we should conclude that a hypothesis *no carnivores eat animals* is false. The system constructs a search problem for each queried hypothesis over relaxed natural logic inferences: the surface form of the hypothesis is allowed to mutate until it matches one of the facts in the knowledge base. These mutations correspond to steps in a natural logic proof; a learned cost for each mutation corresponds to the system's confidence that the mutation is indeed logically valid (e.g., mutating to a hypernym has low cost, whereas nearest neighbors in vector space has high cost). This amounts to a high-performance fuzzy theorem prover over an arbitrarily large premise set, where the extent to which particular logical mutations are correct or incorrect can be learned from a training set of known correct and incorrect facts.

An illustration of a search from the query *no carnivores eat animals* is given below, with the appropriate natural logic relation annotated along the edges. The mutation from *no* to *the* negates the sentence ($\curlywedge$); the mutations from *carnivore* to *cat*, the introduction of *an*, and the mutation from *animal* to *mouse* all preserve negation ($\sqsupseteq$). Therefore, the premise negates the query fact:



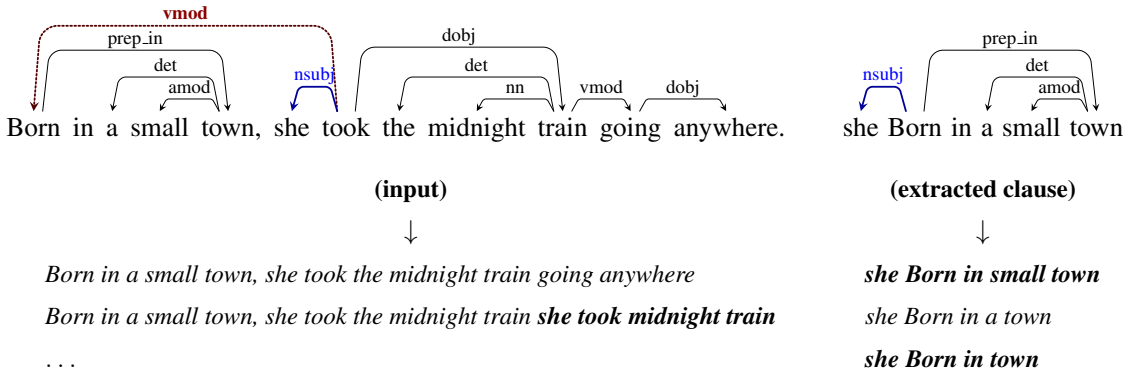
This framing of the problem has a number of advantages. Prior work in *textual entailment* – the task of determining if a premise sentence logically entails the truth of a hypothesis – has traditionally only dealt with very small (1 or 2 sentence) premise sets. In contrast, this approach scales to arbitrarily large premise sets. Prior work in large-scale inference – for example approaches making use of large graphical models (e.g., Markov Logic Networks [87]) – tend to be computationally inefficient as the size of the inference problem grows. This approach becomes more efficient the larger the premise set grows, since we can run a shallower search in expectation. From the other direction, unlike information retrieval approaches, we remain sensitive to a notion of entailment rather than simply similarity – for example, we can detect false facts in addition to true ones. In an empirical evaluation, we show that we can recover 50% of common sense facts at 90% precision – 4x the recall of directly querying a very large source corpus of 270 million premises.

Although this approach works well in cases where the the source corpus is composed of simple atomic facts, most useful facts in the wild are embedded as small parts of more

complex sentences. This is, in fact, relevant not only as a subcomponent in our reasoning engine (it will allow us to digest complex sentences from real-world data sources and segment them into atomic facts) but also as a standalone application. A common motif in information extraction in general is the value in converting such complete sentences into a set of atomic propositions.

Chapter 5 describes our system to extract atomic propositions (e.g., *Obama was born in Hawaii*) from longer, more syntactically difficult sentences (e.g., *Born in Hawaii, Obama attended Columbia*) by recursively segmenting a dependency tree into a set of self-contained clauses expressing atomic propositions. This segmentation is done by defining a breadth-first search on a dependency tree, where at each arc a decision is made among a set of actions determining whether to split off the subordinate clause, and if so how that split should occur. These clauses are then maximally shortened to yield propositions which are logically entailed by the original sentence, and also maximally concise. For instance, the statement *anchovies were an ideal topping for Italian sailors* yields *anchovies are a topping.*

For example, the figure below shows the system segmenting the sentence *born in a small town, she took the midnight train going anywhere* into two "clauses," and then shortening each clause to obtain maximally informative short propositions. The left "clause" is the entire sentence. In that context, the main message we are trying to convey is that a girl ("*she*") took the midnight train. Her birthplace, the destination of the train, etc. are supplemental bits of information. Therefore, we are justified in stripping off these additional modifiers, and arriving at a maximally concise utterance (i.e., dependency tree fragment) *she took midnight train*. However, we also extract *she [was] born in a small town* as a separate clause. Now, her birthplace is the central theme of the clause, and the most we can strip off is the qualifier *small* on *town*:



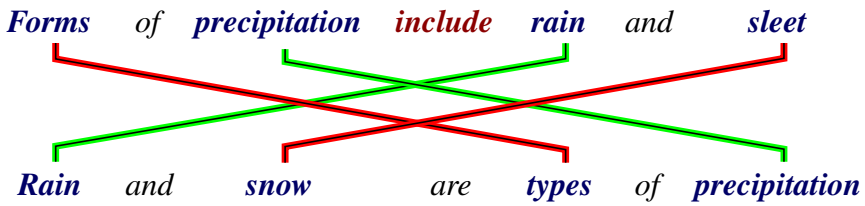
In addition to being a component in our reasoning engine, we can directly use this method for Open Information Extraction (Open IE; see Section 2.2) – a flavor of relation extraction where the relation, subject, and object are all allowed to be *open domain* plain-text strings. On a NIST-run knowledge base population task, we show that this system achieves good results, outperforming all prior work applying OpenIE to structured relation extraction by 3 F_1. Despite not being developed for this task, our system achieves a score halfway between the median and top performing system, outperforming multiple purpose-built systems.

Lastly, a key property of natural logic is its ability to interface nicely with statistical models which featurize the surface form of a sentence. This allows us to construct models which capture the broad coverage and high recall of shallow featurized models, while nonetheless maintaining much of the nuanced logical reasoning that natural logic provides. This is in contrast to most earlier work using structured logic for question answering tasks, which struggle to get reasonable recall. For example, Bos and Markert [16] observe that, in the RTE task of determining whether a sentence entails another sentence, strict logical methods obtain only 5.8% recall.

Chapter 6 outlines a method for combining these two signals – a shallow featurized classifier and our natural logic engine – elegantly in a single unified model. We continue to view our problem as a search problem, where we mutate the query fact in ways that are warranted (or approximately warranted) by natural logic until we hit a premise fact. However, since each of the intermediate states in this search is in itself a natural language sentence, we can featurize these intermediate states and run a classifier to predict whether there is a premise which is *likely* to entail (or contradict) that state. Since the intermediate state

entails (or contradicts) the query fact, we can infer from transitivity that the same premise entails / contradicts the query fact. This can be thought of as an evaluation function in the same way that a gameplaying agent uses an evaluation function to assess the game state. If the search problem reaches a terminal state (a known premise in our case, analogous to, e.g., a checkmate in Chess) then the evaluation function is not needed. However, if no terminal state is found, it is nonetheless useful to have an estimate for how close we are to a terminal state.

The evaluation function is carefully designed to be easy to update during the search process. It makes use of a set of alignment features – e.g., as in the figure below – such that as we mutate the premise (*rain and snow are types of precipitation*) we can match fewer or more alignments. Weights over these features are learned, presumably such that more valid alignments is positively correlated with entailment. Note that even in the below example, despite the premise and the hypothesis being substantially similar they are not technically an entailment pair, and therefore the evaluation function is crucial to produce an entailment relation. Moreover, note that as the search progresses (e.g., *types* is replaced by its synonym *forms*) the confidence of entailment will improve.



We evaluate this complete system on 4th grade science exam questions, and show that we outperform prior work, a strong information retrieval baseline, and a standalone version of the evaluation function.

Together, these contributions form a powerful reasoning engine for inferring open-domain knowledge from very large premise sets. Unlike traditional IR approaches, or shallow classification methods, the system maintains a notion of logical validity (e.g., proper handling of negation); unlike structured logical methods, the system is high-recall and robust to real-world language and fuzzy inferences. From here, the foundation is laid to

leverage this sort of knowledge in a general way, in pursuit of systems that exhibit broad-domain intelligence. Returning to the examples from the beginning of the section, if we are faced with a question like:

I ate a bowl of soup, and put the bowl in the bathtub. Did it float?

We have a method for determining from a large unlabeled corpora of text that a bowl is concave, empty concave things float, and so forth. We do this by exploiting three key insights: first, that we can run statistical natural logic proofs over a very large natural language premise set by casting the task as a search problem, producing the most efficient open-domain reasoning engine over such a large number of premises. Second, we can segment long, syntactically complex sentences into simple atomic utterances that are convenient for this natural logic search problem. Third, we can augment the search with an evaluation function, which ensures high recall in retreiving the truth of these facts . The remainder of this disseration will review natural logic as a formalism, and then describe in depth the components of this reasoning system.

Chapter 2

Related Work

In this chapter, we review the body of work most related to this dissertation, and to the larger goal of extracting knowledge from text. We cover related work in extracting structures knowledge bases from text (Knowledge Base Population) and open domain knowledge base population as two methods for extracting structured and semi-structured knowledge from text. We then review work on common-sense reasoning – one of the main goals of this dissertation. Lastly, we review work on textual entailment, related to the underlying natural logic proof system in this dissertation, and conclude with reviewing work on question answering.

2.1 Knowledge Base Population

Knowledge base population is the task of taking a large body of unstructured text, and extracting from it a structured knowledge base. Importantly, the knowledge base has a fixed schema of relations (e.g., *born in*, *spouse of*), usually with associated type signatures. These knowledge bases can then be used as a repository of knowledge for downstream applications – albeit restricted to the schema of the knowledge base itself. In fact, many downstream NLP applications do query large knowledge bases. Prominent examples include question answering systems [107], and semantic parsers [9, 61, 116, 118].

Prior work in this area can be categorized into a number of approaches. The most common of these are *supervised* relation extractors [37, 48, 101], *distantly supervised* relation

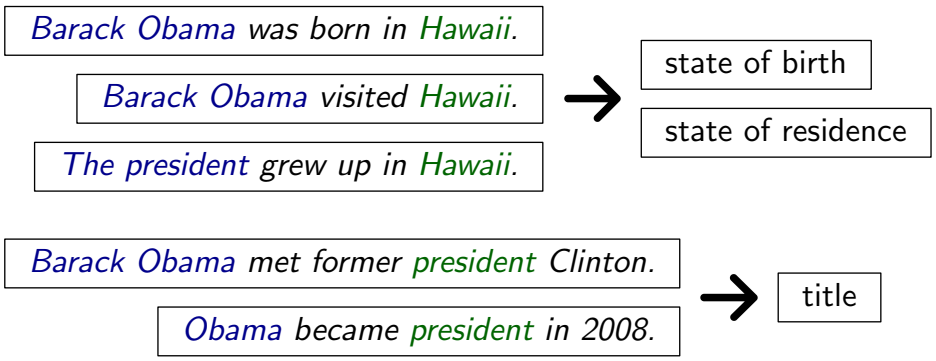


Figure 2.1: The relation extraction setup. For a pair of entities, we collect sentences which mention both entities. These sentences are then used to predict one or more relations between those entities. For instance, the sentences containing both *Barack Obama* and *Hawaii* should support the state of birth and state of residence relation.

extractors [33, 78, 99, 111], and rule based systems [26, 47, 97].

Relation extraction can be naturally cast as a supervised classification problem. A corpus of relation mentions is collected, and each mention x is annotated with the relation y, if any, it expresses. The classifier's output is then aggregated to decide the relations between the two entities.

However, annotating supervised training data is generally expensive to perform at large scale. Although resources such as Freebase or the TAC KBP knowledge base have on the order of millions of training tuples over entities it is not feasible to manually annotate the corresponding mentions in the text. This has led to the rise of *distantly supervised* methods, which make use of this indirect supervision, but do not necessitate mention-level supervision.

Traditional distant supervision makes the assumption that for every triple (e_1, y, e_2) in a knowledge base between a subject e_1, a relation y, and an object e_2, every sentence containing mentions for e_1 and e_2 express the relation y. For instance, taking Figure 2.1, we would create a datum for each of the three sentences containing BARACK OBAMA and HAWAII labeled with state of birth, and likewise with state of residence, creating 6 training examples overall. Similarly, both sentences involving *Barack Obama* and *president* would be marked as expressing the title relation.

While this allows us to leverage a large database effectively, it nonetheless makes a number of naïve assumptions. First – explicit in the formulation of the approach – it assumes that every mention expresses some relation, and furthermore expresses the known relation(s). For instance, the sentence *Obama visited Hawaii* would be erroneously treated as a positive example of the born in relation. Second, it implicitly assumes that our knowledge base is complete: entity mentions with no known relation are treated as negative examples.

The first of these assumptions is addressed by multi-instance multi-label (MIML) learning, which puts an intermediate latent variable for each sentence-level prediction, that then has to predict the correct knowledge base triples [102]. Min et al. [77] address the second assumption by extending the MIML model with additional latent variables, while Xu et al. [113] allow feedback from a coarse relation extractor to augment labels from the knowledge base. These latter two approaches are compatible with but are not implemented in this work.

Lastly, there are approaches to inferring new facts in a knowledge base that do not make use of text at all, but rather aim to complete missing facts from the knowledge base based on patterns found from known facts. For example, in the simplest case, if we know that a person usually lives in the state of his employer, we can with high confidence predict a missing state of residence if we know a person's employer, and the employer's state of headquarters.

Chen et al. [24] and Socher et al. [94] use Neural Tensor Networks for this task, predicting unseen relation triples in WordNet and Freebase. This builds on a line of work by Bordes et al. [14] and Jenatton et al. [55] on knowledge base completion based on vector-space approaches. Yao et al. [114] and Riedel et al. [88] present a related line of work, inferring new relations between Freebase entities via inference over both Freebase and OpenIE relations (see Section 2.2). This work appeals to low-rank matrix factorization methods: in the simplest case, entity pairs form the rows of a matrix, and possible relations form the columns. The entries of the matrix are 1 is a given relation holds between the given entity pair, and 0 otherwise. This matrix is then factored into a low-rank approximation, in effect trying to compress the information in the knowledge base into a more concise representation. Since this is an approximate factorization, some entries in the matrix which

were previously 0's now have a non-zero value – these are taken to be new facts in the knowledge base.

2.2 Open Information Extraction

One approach to broad-domain knowledge extraction is *open information extraction* (Open IE). Traditional relation extraction settings usually specify a domain of relations they're interested in (e.g., place of birth, spouse, etc.), and usually place restrictions on the types of arguments extracted (e.g., only people are born places). Open IE systems generalize this setting to the case where both the relation and the arguments are represented as plain text, and therefore can be entirely open-domain. For example, in the sentence *the president spoke to the Senate on Monday*, we might extract the following triples:

(*president*; *spoke to*; *Senate*)
(*president*; *spoke on*; *Monday*)

One line of work in this area are the early UW OpenIE systems: for example, TextRunner [115] and ReVerb [40]. In both of these cases, an emphasis is placed on *speed* – token-based surface patterns are extracted that would correspond to open domain triples. ReVerb, for instance, heuristically extracts relations centered around maximally expanded relations centered around a main verb, and attempts to find the best arguments for that relation subjected to certain constraints. A confidence function is then trained over a set of features on the extractions, so that the system can provide calibrated confidence values.

With the introduction of fast dependency parsers, [112] extracts triples from learned dependency patterns (in one mode), or POS-tagged surface features (for additional speed). Building upon this, Ollie [74] also learns patterns from dependency parses, but with the additional contributions of (1) allowing for extractions which are mediated by nouns or adjectives, not just verbs; and (2) considering context more carefully when extracting these triples. Exemplar [76] adapts the open IE framework to n-ary relationships similar to semantic role labeling, but without the associated expensive machinery of SRL systems.

In another line of work, The Never Ending Language Learning project (NELL) [22] iteratively learns more facts from the internet from a seed set of examples. In the case of

NELL, the ontology is open-domain but fixed, and the goal becomes to learn all entries in the domain. For example, learning an extended hypernymy tree (*Post University* is a University); but also more general relations (*has acquired*, *publication writes about*, etc).

Open IE has been shown to be useful in a number of downstream NLP tasks. One line of work makes use of these open-domain triples to perform the fixed-schema relation extraction we discussed in Section 2.1. For example, [96] constructs a mapping from open-domain to structured triples in only 3 hours of human effort, and achieves results which are competitive with (and higher precision than) custom-trained extractors. Soderland et al. [95] use ReVerb extractions to enrich a domain-specific ontology. Another line of work makes use of these triples in the context of knowledge base completion, as alluded to earlier [88, 114]. In this case, we are learning a sort of soft mapping from Open IE triples to structured triples implicitly in the matrix factorization. In a related vein, OpenIE has been used directly for learning entailment patterns. For example, Schoenmackers et al. [92] and Berant et al. [10] learn valid open-domain entailment patterns from OpenIE extractions. Berant et al. [10], for instance, presents a method for learning that the relation *be epidemic in* should entail *common in* should entail *occur in*.

Another prominent use-case for open domain triples is question answering and search [38, 41]. In Fader et al. [41], a set of question paraphrases is mined from a large question answering corpus; these paraphrases are then applied to a new question until it matches a known Open IE relation. In each of these cases, the concise extractions provided by open IE allow for efficient symbolic methods for entailment, such as Markov logic networks or matrix factorization.

2.3 Common Sense Reasoning

The goal of tackling common-sense reasoning is by no means novel in itself either. Among others, work by Reiter [86] and McCarthy [75] attempts to reason about the truth of a consequent in the absence of strict logical entailment. Reiter introduces a notion of *default reasoning*, where facts and entailments can be taken as true unless there is direct evidence to the contrary. For example, *birds fly* ($\text{bird}(x) \Rightarrow \text{flies}(x)$) is a statement which is clearly

false in the strictest sense – penguins and ostriches don't fly – but nonetheless a statement that we would consider true. Reiter argues that common-sense statements of this sort should be considered true by default in a formal reasoning engine, unless there is explicit evidence against them. McCarthy argues a related point on circumscription: that in the absence of evidence to the contrary, we should assume that all prerequisites for an action are met. For instance, if someone claims to have a rowboat, we should assume that they have oars, the oars fit into the rowlocks, the boat is without holes, etc.; unless explicitly told otherwise. In another line of work, Pearl [83] presents a framework for assigning confidences to inferences which can be reasonably assumed.

More recently, work by Schubert [93] and Van Durme et al. [106] approach common sense reasoning with *episodic logic*. Episodic logic is a first-order logic which focuses primarily on time-bounded situations (events and states), rather than the time-insensitive propositions of conventional first-order logic. For example, a sentence *John kicked Pluto* could be interpreted as:

$$\exists e1 : [e1 \text{ before Now1}]\,[[\text{John kick Pluto}] * * e1]$$

That is, there is an event $e1$ that happened before now (the reference time), and the various the sentence *John kick Pluto* characterizes or fully describes $e1$. The operator $**$ has an analogous operator $*$ for cases where the statement (or formula) is *true* in the event, but does not fully characterize it For example:

$$\exists e1 : [e1 \text{ before Now1}]\,[[\text{John has a leg}] * * e1]$$

Lastly, efforts like MIT's ConceptNet [103] and Cyc [63] aim to create a comprehensive, unified knowledge base of common-sense facts. For instance, ConceptNet has facts like (*learn*; *motivated by goal*; *knowledge*). These facts are collected from a number of sources, both automatic (e.g., Open IE) and hand-curated (e.g., WordNet). Cyc catalogs in a structured way facts like: There exists a female animal for every Chordate which is described by the predicate *mother*. All of these facts are hand-curated.

2.4 Textual Entailment

This dissertation is in many ways to work on recognizing textual entailment (RTE) – e.g., Schoenmackers et al. [92], Berant et al. [10]. Textual Entailment is the task of determining if a given premise sentence entails a given hypothesis. That is, if without additional context, a human would infer that the hypothesis is true if the premise is true. For instance:

I drove up to San Francisco yesterday
I was in a car yesterday

Although the definition of entailment is always a bit fuzzy – what if I drove a train up to SF, or perhaps a boat? – nonetheless a reasonable person would assume that if you drove somewhere you were in a car. This sort of reasoning is similar to the goal of this dissertation: given premises, infer valid hypothesis to claim as true. However, in RTE the premise set tends to be very small (1 or 2 premises), and the domain tends to have less of a focus on common-sense or broad domain facts.

Work by Lewis and Steedman [64] approach entailment by constructing a CCG parse of the query, while mapping questions which are paraphrases of each other to the same logical form using distributional relation clustering. Hickl [49] approaches the problem by segmenting the premise and the hypothesis into a set of *discourse commitments* that a reader would accept upon reading the sentence. This can then be used to determine whether the commitments in the premise match those in the hypothesis. For example, the following premise, taken from the RTE-3 challenge:

> "The Extra Girl" (1923) is a story of a smalltown girl, Sue Graham (played by Mabel Normand) who comes to Hollywood to be in the pictures. This Mabel Normand vehicle, produced by Mack Sennett, followed earlier films about the film industry and also paved the way for later films about Hollywood, such as King Vidors "Show People" (1928).

would yield the commitments:

T1. "The Extra Girl" [took place in] 1923.
T2. "The Extra Girl" is a story of a smalltown girl.

T3. ”The Extra Girl” is a story of Sue Graham.

T4. Sue Graham is a smalltown girl.

T5. Sue Graham [was] played by Mabel Normand.

. . .

T10. “The Extra Girl” [was] produced by Mack Sennett.

T11. Mack Sennett is a producer.

This could then be used to easily justify the hypothesis: *“The Extra Girl” was produced by Sennett*.

This dissertation focuses a fair bit on natural logic. The primary application of natural logic in prior work has been for textual entailment and related fields. For example, work by MacCartney [70, 71, 72, 72] has applied natural logic to the RTE challenges described above. This work approaches the entailment task by inducing an alignment between the premise and the hypothesis, classifying entailed pairs into natural logic relations, and then inferring from this alignment and the semantics of natural logic what the correct entailment relation is between the sentences. This line of work has been extended since in, e.g., Watanabe et al. [110]. More background on natural logic is given in Chapter 3.

More recently, work by Bowman [18] (and Bowman et al. [19] has explored using vector spaces as a meaning representation for natural language semantics. Entailment was chosen as difficult, comprehensive evaluation of natural language understanding, and since natural logic is inherently lexical, it was chosen as the formalism against which to test these models. Results from these papers show that, indeed, appropriately composing vector-space representations of words can yield systems which capture a notion of entailment with high accuracy.

Since the discontinuation of the RTE challenges, there have been a number of new efforts in creating datasets for textual entailment against which systems can be trained. For example, Marelli et al. [73] produce a dataset of entailment pairs from image captions which is an order of magnitude larger than the RTE datasets. Subsequently, Bowman et al. [19] created a very large corpus of human-annotated entailment pairs, again from image captions. This dataset has since been used as the de-facto large-scale dataset on which to evaluate entailment models, and most prominently neural sequence models for entailment. For example, Rocktäschel et al. [89] construct a recurrent neural network with attention

for entailment; Cheng et al. [27] and other follow up on that work with a new attention mechanism.

2.5 Question Answering

Many parts of this dissertation are reminiscent of question answering tasks. Related work on question answering can be segmented into two broad categories: Early work focused on question answering directly over the text of a source corpus. For example, taking as input a Wikipedia article, and using that text to answer questions about the article's subject. Later work, in contrast, focuses on using structured knowledge bases for question answering with an emphasis on moving in the direction of complicated compositional queries.

A representative task in the first line of work – making direct use of text for question answering – is the TREC Q/A competitions [34, 107, 108, 109]. An early example of such a system is Textract [98], based around a pipeline of information extraction components. The system would take a question (e.g., *who is the president of the United States?*) and convert it into a target named entity type (i.e., PERSON), and a set of keywords (i.e., *president*, *United States*). The keywords are then fed into a query expansion module to expand the domain of candidate premises returned, and the resulting sentences are scanned for an entity of the right named entity type based on some heuristics.

This can be cast as a simplistic version of the two-stage Q/A approach of running information retrieval (IR), and then classifying the results into whether they are an answer to the questions. Subsequent work has expanded on this sort of IR+classify approach to, e.g., use language modeling to assist in reranking results [25], incorporate more sophisticated methods for learning to rank [21], incorporating additional knowledge sources such as Wikipedia [1], etc.

The other main line of work uses structured knowledge bases for question answering. The main line of work here is in the tradition of semantic parsing [60, 65, 117]. An input query is parsed into a structured logical representation; this representation can then be run against a knowledge base to retrieve the answer to the query. For example, Zettlemoyer and Collins [117] and Liang et al. [65] parse complex, compositional geological queries into a logical form that can be evaluated against a database of geological facts. A complex

sentence like:

What states border the state that borders the most states?

Would be parsed into a logical form (effectively, a structured query) like:

$$\lambda x.state(x) \wedge borders(x, \text{argmax}(\lambda y.state(y), \lambda y.count(\lambda z.state(x) \wedge borders(y, z))))$$

Zettlemoyer and Collins [117] approaches the problem by learning a synchronous combinatory categorical grammar (CCG) [17] parse over lexical items and logical form fragments, given a fixed grammar. Until convergence, the algorithm performs the following two steps:

- Expand the set of lexical mappings (e.g., *state* means $\lambda x.state(x)$) known to the algorithm. This is done by performing the following on each example in the training set: (1) Generate all possible lexical mappings, given the logical form and surface form of the example. (2) Parse the example, given the current learned parameters. (3) Add to the set of lexical mappings all lexical mappings in the parsed logical form.
- Learn a new parameter vector for the parser, given the lexicon so far and the training set.

Work by Liang et al. [65] builds on this approach by removing the need for an annotated logical form, and instead running a two-step EM-like learning algorithm (on a fixed CFG rather than CCG grammar): First, a set of possible parses for each example are induced given the current parameter vector. The parses which evaluate to the correct *answer* are treated as correct and renormalized proportional to their original score. This is then used as a training signal to learn a new parameter vector.

This same type of approach can be applied more broadly to more practical types of questions. For example, Berant et al. [11] apply a similar approach to answering questions about entities in Freebase – a popular large knowledge base containing facts about people, places, organizations, etc. In this case, utterances are factoid-style Q/A questions similar to the TREC questions; e.g., *what college did Obama go to?*. This is then parsed into a logical

form, and executed against the Freebase knowledge graph to produce the desired answer: Columbia and Harvard.

Subsequent work has also applied semantic parsing to even more broad-domain representations. For example, work by Artzi et al. [6] uses semantic parsing methods to parse into the Abstract Meaning Representation [7] – a broad-domain structured meaning representation. New methods (e.g., Bordes et al. [15]) have applied neural methods to embed knowledge base fragments and natural language questions in vector spaces, and answer questions based on neural network methods.

Lastly, there has been work on incorporating more structured logical reasoning for question answering. The COGEX system [79] incorporates a theorem prover into a QA system, boosting overall performance. Similarly, Watson [42] incorporates logical reasoning components. In a new spin on logical methods for question-answering, Hixon et al. [50] proposes a dialog system to augment a knowledge graph used for answering science exam questions. This is in a sense an oracle measure, where a human is consulted while answering the question to substantially improve accuracy. Furthermore, they show that their additional extractions help answer questions other than the one the dialog was collected for; that is, human-in-the-loop Q/A improves accuracy even on unseen questions.

Chapter 3

Natural Logic

A recurring theme throughout this dissertation will be the use of natural logics for natural language tasks. To begin, we should briefly motivate the use of any logical formalism at all for natural language. A wide range of successful approaches in NLP neither use nor need logical reasoning to perform their task: statistical parsing, part-of-speech tagging, document classification, etc. are all fairly agnostic to any logical phenomena in the documents they are operating over. However, as the field of NLP moves away from syntax and shallow classification tasks and increasingly towards natural language *understanding*, we are faced with a necessity to understand meaning of sentences, and understand the implications that can be drawn from them.

Taking some simple examples across a few tasks: in relation extraction, it would be incorrect to assume that Obama was born in Kenya from a sentence *Obama was not born in Kenya*. For sentiment analysis, the sentence *The movie's box office failure should not be taken as indicative of its quality* should get positive sentiment. In question answering, returning *Harrison Ford* for a query asking for all the female stars of Star Wars would be incorrect.

In all of these cases, a system must either explicitly or implicitly learn how to handle these sorts of logical subtleties. Implicitly, a neural network or sufficiently powerful statistical classifier could learn how to treat these queries, at the expense of sufficiently large amount of training data (in the extreme, one can always simply memorizing the logic). Explicitly, a first order theorem prover or Markov Logic network could capture the logical

content of a sentence, at the expense of a drop in expresivity and computational efficiency. In this chapter we'll motivate natural logics as an optimal midway between offloading this sort of logical reasoning entirely to statistical / neural methods, and enforcing a rigid logical framework on the problem.

Broadly, the aim of natural logics are to capture a subset of valid logical inferences by appealing directly to the structure of language, as opposed to running deduction in an abstract logical language (e.g., well-formed first-order formulae composed predicates, variables, connectives, and quantifiers). That is to say, a natural logic is primarily a *proof theory*, where the deductive system operates over natural language syntax. To illustrate with an example, we can consider a simple premise:

The cat ate a mouse.

From this, we can run the following logical derivation, using the syntax of natural language as the proof language:

1	*The cat ate a mouse.*
2	*The cat ate a rodent.*
3	*The cat ate an animal.*
4	*The feline ate an animal.*
5	*The carnivore ate an animal.*
6	$\neg$ *No carnivore ate an animal.*

That is to say, if the cat ate a mouse, it is false that no carnivore ate an animal, and this is by virtue of the "proof" presented above. Although it's not unnatural to use a line of reasoning like the one above to justify a conclusion informally, it may be strange the perspective of propositional and first-order logics to treat this as a formal proof. Nonetheless, it is not an altogether unfamiliar sort of reasoning. Natural logics can trace their origins to the syllogistic reasoning of Aristotle. For instance, the proof above is similar to a simple syllogism:

1	*All cats eat mice.*
2	*All cats are carnivores.*
3	*All carnivores eat mice.*

We can furthermore chain these syllogisms in much the same way you would chain a first-order proof. To introduce a motivating example, an Athenian philosopher might retort with the following if a Persian invader claims that all heroes are Persians:

> *Clearly you are wrong. You see, all Gods live on Mount Olympus. Some heroes are Gods. And, of course, no one who lives on Mount Olympus is Persian.*

Sadly our poor philosopher will have likely perished by this point – lest a Greek hero saves him, in which case then there would remain neither a need nor an audience for his argument. But but I can still convince you, the reader, that this was a valid retort by constructing a first-order proof, given in Figure 3.1. The proof follows a proof by contradiction, showing that the hypothesized hero-God (*"some heroes are Gods"*) would have to both live on Olympus and be Persian – a contradiction with our premise: *"no one who lives on Mount Olympus is Persian."*

To contrast with the first order logic proof, there also exists a natural logic proof of our contradiction, based entirely on Aristotilean syllogisms. This proof makes use of two syllogistic patterns chained together, and one of the axiomatic negations:

1	*All Gods live on Mount Olympus*	
2	*Some heroes are Gods*	
3	*Nobody who lives on Mount Olympus is Persian*	
4	*Some heroes live on Mount Olympus*	AII (Darii), 1, 2
5	*Some heroes are not Persian*	EIO (Ferio), 4, 3
6	$\neg$ *All heroes are Persian*	SaP $\perp$ SoP, 5

1	$\forall x$ God$(x) \supset$ LivesOnOlympus(x)	
2	$\exists x$ Hero$(x) \wedge$ God(x)	
3	$\neg\exists x$ LivesOnOlympus$(x) \wedge$ Persian(x)	
4	$\quad\forall x$ Hero(x) $\supset$ Persian(x)	
5	$\quad a \quad$ Hero$(a) \wedge$ God(a)	$\exists$E, 2
6	$\quad\quad$ Hero(a)	$\wedge$E, 5
7	$\quad\quad$ Hero(a) $\supset$ Persian(a)	$\forall$E, 4
8	$\quad\quad$ Persian(a)	$\Rightarrow$E, 6, 7
9	$\quad\quad$ God(a)	$\wedge$E, 5
10	$\quad\quad$ God$(a) \supset$ LivesOnOlympus(a)	$\forall$E, 1
11	$\quad\quad$ LivesOnOlympus(a)	$\Rightarrow$E, 9, 10
12	$\quad\quad$ LivesOnOlympus$(a) \wedge$ Persian(a)	$\wedge$I, 8, 11
13	$\quad\quad \exists x$ LivesOnOlympus$(x) \wedge$ Persian(x)	$\exists$I, 12
14	$\quad\exists x$ LivesOnOlympus$(x) \wedge$ Persian(x)	R, 12
15	$\quad\perp$	$\perp$ I, 3, 14
16	$\neg\ \forall x$ Hero(x) $\supset$ Persian(x)	$\neg$ I, 4—15

Figure 3.1: A Fitch-style first order logic proof refuting *"all heroes are Persian"* given the premises *"All Gods live on Mount Olympus," "Some heroes are Gods,"* and *"No one who lives on Mount Olympus is Persian."*. The proof follows a proof by contradiction (lines 4–15), hinging on showing that the hero-God hypothesized in premise 2, and instantiated as a on line 5, would have to be a Persian who lives on Olympus. This would contradict premise 3.

This example – and in particular the contrast between the two proof approaches above – provides an excellent context for motivating why the enterprise of natural logics is worthwhile, and why research in the area can have a large impact on natural language processing. I'll highlight three concrete motivations in more detail: (1) natural logics are easy to parse into, (2) this parsing is, in a sense, "lossless," and (3) the proofs have the potential to be much more efficient than first order approaches.

Natural logics are easy to parse into. When performing inference in propositional or first order logic, the premises are no longer in natural language, but rather have been parsed into a special logical form. Ignoring the aesthetics of the logical form itself, this mapping is by no means trivial; in fact, an entire subfield of NLP – semantic parsing – focuses exclusively on this problem [11, 59, 65, 117]. Even in our simple example, an intuitively simple utterance like *"some heroes are Gods"* parses into a premise which reads most naturally as *"there exists something which is a hero and a God."* Even more subtly, consider the difference in form for the following two translations:

Sentence	**Logical Form**
Everyone on Mount Olympus is Persian	$\forall x$ LivesOnOlympus$(x) \supset$ Persian(x)
No one on Mount Olympus is Persian	$\neg\exists x$ LivesOnOlympus$(x) \wedge$ Persian(x)

Despite the lexical forms being nearly identical, the logical form has an entirely different structure, changing not only the quantifier but also the connective ($\supset$ versus $\wedge$). By contrast, "parsing to a logical form" in syllogistic logic is nonexistent, and in the more sophisticated natural logics later in this chapter generally reduces to a shallow syntactic parse of the sentence.

Natural logic parsing is "lossless." In general, natural language has more semantic (not to mention pragmatic) content than the logical propositions we extract from it. This is evident in the fact that it's necessary to retrain semantic parsers for new tasks, and underlies the research agenda of defining general semantic representations – e.g., the Abstract Meaning

Representation [7]. Despite being picked to be as concise as possible, even our motivating example has hints of this. The derivation in Figure 3.1 has defined the predicates necessary for our particular derivation (LivesOnOlympus, God, Hero, Persian); but, these are not the only things that could be extracted from the sentences. Semantically, we could extract that Gods are alive. Pragmatically, we should extract that there exist heroes.

To contrast, by operating over natural language directly, the limitations on what natural logics can infer are entirely due to the limitations of the inference mechanism, rather than limitations in what information was successfully extracted from the sentence.

Natural logic proof are efficient. Annecdotally, it should be clear that the proof in Figure 3.1 is longer and more nuanced than the corresponding syllogistic proof. In fact, to a first approximation, one could make a syllogistic theorem prover with nothing more than regular expressions and a small search algorithm. Of course, more expressive natural logics have more difficult search problems; but, nonetheless, they remain in the same spirit of searching over lexical mutations rather than applying symbolic rule sets.

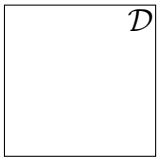
Contrasting with approaches like Markov Logic Networks makes this difference especially salient [87]. Formulating our motivating example as a Markov Logic Network requires outright instantiating the denotations of our predicates on a closed world. That is, we must first enumerate all the heroes, Gods, and residents of Olympus. Thereafter, the task of grounding our formulae on this world and running heavyweight statistical inference algorithm the resulting Markov Network. Making this grounding and inference process tractable is in itself an active area of research [81, 119]. By contrast, natural logics are almost by definition a *proof-theoretic* logic: quantifiers can be reasoned over efficiently using only lexical mutation based proofs. Although they have a model-theoretic interpretation, actually simulating this model is unnecessary and would be impractical.

Of course, despite these advantages it would be unreasonable to advocate for syllogistic reasoning as a practical formalism for modern AI applications. Syllogisms don't allow for compositionality, and are generally restrictive in the range of inferences they warrant. This dissertation instead adopts variants of *monotonicity calculus* [90, 104] – a more general-purpose logic which handles a wide range of common phenomena in human language, while nonetheless still operating over the syntax of the language itself.

The remainder of chapter will review denotational semantics (Section 3.1) as a prelude to introducing the monotonicity calculus of Sánchez Valencia [90], described in detail in Section 3.2. From here on, *natural logic* will be used to refer to the monotonicity calculus described in Sánchez Valencia [90]. These sections are intended to give a *model-theoretic* interpretation of the logic that can then be used to justify the potentially ad-hoc seeming proof theory described here. We motivate further research into related formalisms in Section 3.3.

3.1 Denotations

An underlying central concept behind monotonicity calculus (the natural logic used throughout this thesis) is the notion that lexical items can be interpreted in terms of their effect on the set of objects in the world. To be precise, let's introduce a domain $\mathcal{D}$, representing the set of all items and concepts in the world. We'll show this $\mathcal{D}$ as an empty box:



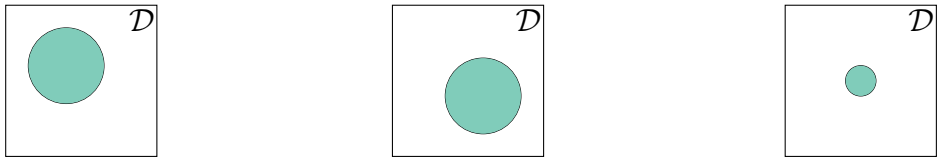
Now, we can start labeling items in this set. For example, this dissertation is an item in the world; your hand is likewise an item in the world. We can write these statements as: *this thesis*$\in \mathcal{D}$ and *your hand*$\in \mathcal{D}$; visually, we might show the following:

The central dissertation behind denotational semantics is the notion that words have *denotations*, which are sets of elements in a particular domain. For example, this dissertation

and your hand have denotations which are the singleton sets of elements in the domain of *things in the world*. Analogously, cats will be the set of cats in the world, *run* will be the set of actions which we'd label as running, and so forth. The remainder of this section will go over how we map different lexical items to denotations in different domains. This will form the basis of the model theory behind monotonicity calculus, introduced in Section 3.2.

3.1.1 Nouns and Adjectives are Sets

We represent nouns and adjectives as sets – more precisely, as subsets of our domain $\mathcal{D}$. For example, the word *cat* refers to the set of all cats in the world. *Cute* refers to the set of all cute things in the world. Note that this definition subsumes the cases in the previous section: *this thesis* becomes simply the singleton set containing this thesis. We represent these denotations as $[\![cat]\!] \subseteq \mathcal{D}$ (the set of all cats), $[\![cute]\!] \subseteq \mathcal{D}$ (the set of all cute things), etc. Visually, we can show these as a subset of our domain $\mathcal{D}$:



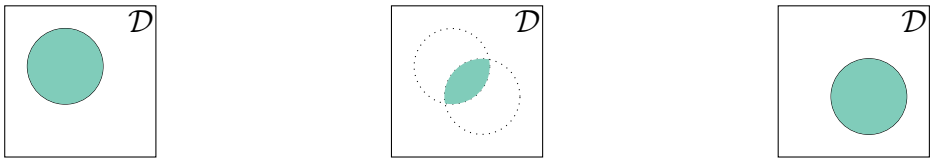
$[\![cute]\!]$ $[\![cat]\!]$ $[\![this\ thesis]\!]$

This is the same sort of representation used in other areas of NLP, most prominently the semantic parsing literature (see, e.g., Liang and Potts [66]). This similarity becomes more clear if we consider nouns and adjectives as *predicates* rather than sets. That is, the word *cat* is a predicate which is true if its argument is a cat, and false otherwise. These two interpretations are, of course, equivalent. A predicate can be represented as the set of entities in the world which satisfy it, and a set of entities in the world can be represented as the predicate that selects them.

The key difference between denotations in monotonicity calculus and semantic parsing is that in the later, the world $\mathcal{D}$ and the denotations of words have concrete (domain-specific) instantiations. For example, $\mathcal{D}$ may be a database of places, people, etc. that can be queried against. In that case, the word *river* would correspond to a finite set of database

rows for the rivers in the world. In monotonicity calculus, as we'll see more clearly in Section 3.2, we will never appeal to the explicit denotations of these lexical items. It is sufficient to know that there exist a set of rivers in the world; we never need to explicitly enumerate them.

A natural next question is how we compose lexical items into compound nouns and noun phrases. Like nouns and adjective, noun phrases are denoted as sets of object: *cute cat* is simply the subset of items in the world which are cute cats. Naïvely, this composition amounts to a simple set intersection: *cute cat* refers to the set of things which are both cats and cute; that is, $[\![cute\ cat]\!] = [\![cute]\!] \cap [\![cat]\!]$:



$[\![cute]\!]$ $[\![cute\ cat]\!]$ $[\![cat]\!]$

Less naïvely, we could consider other types of composition (for more information, see e.g., Kamp [58]). For instance, not all adjectives behave intersectively in the way shown above. A *quick DMV visit* is still a DMV visit, but is likely not in the denotation of quick things. Similarly, a *small planet* is nonetheless not generally considered a *small* thing. We refer to these as *subsective* adjectives – the denotation of the compound is a subset of the denotation of the noun, but not a subset of the denotation of the adjective. Another class of adjectives are outright non-subsective. A pseudo-science is not a science; a fake gun is not a gun. In idiomatic expressions, the denotation of the compound has nothing to do with the denotations of the components: a red herring is neither red nor a herring.

We'll revisit compositionality in Section 3.2 and show that handling these sorts of phenomena is important to ensure that the logic remains sound. However, for now we'll continue to review the basic components of the logic.

3.1.2 Sentences are Truth Values

Like most logics, the primary goal of natural logic is to assign truth values to sentences. In natural logic the model-theoretic interpretation of a sentence is simply its truth value. That is to say, sentences can have one of two denotations: they are either true, or they are false.

To make this more formal, and to lay the notational groundwork for the rest of this chapter, let us redefine our domain $\mathcal{D}$ to be $\mathcal{D}_e$ – the domain of entities in the world. So, now, $[\![cat]\!] \subseteq \mathcal{D}_e$. We are now free to specify other domain for other types of lexical items. In our case, let us define $\mathcal{D}_t$ as the domain of truth values. The natural way to define $\mathcal{D}_t$ would be to say: $\mathcal{D}_t = \{true, false\}$ (equivalently, and as we'll find useful later, $\mathcal{D}_t = \{1, 0\}$). The denotation of a sentence is an element in $\mathcal{D}_t$:

$$[\![\textit{cats eat mice}]\!] = true \in \mathcal{D}_t$$
$$[\![\textit{cats eat carrots}]\!] = false \in \mathcal{D}_t$$

3.1.3 Other Lexical Items are Functions

Our review of possible linguistic phenomena or types of lexical items is by no means exhaustive. For instance, we have not covered verbs, or adverbs. Without belaboring the chapter with pedantic thoroughness, the claim of natural logic is that we can construct a denotation for any of these items inductively from the two basic domain – the domain of entities $\mathcal{D}_e$ and the domain of truth values $\mathcal{D}_t$ – and functions based around these domains.

To begin, let's define a set $\mathbf{D}_e$ to be the power set $\mathcal{D}_e$: the set of all subsets of items in the world. This is the set of all possible denotations. We can then define a class of functions $f : \mathbf{D}_e \rightarrow \mathcal{D}_t$. That is, the class of functions mapping from an entity to a truth value. This corresponds most intuitively to the class of intransitive verbs: *plays*, *runs*, *eats*, *barks*, etc.; it also corresponds to the denotations for longer phrases like *plays chess* and *barks at the neighbor*. As a useful visualization, we can "plot" this function along its domain and range in Figure 3.2. The x axis lists the set of denotations in the world. Recall that each of these is just an arbitrary set of items, although we will label them as denotations of words. The y axis is simply the set of true and false.

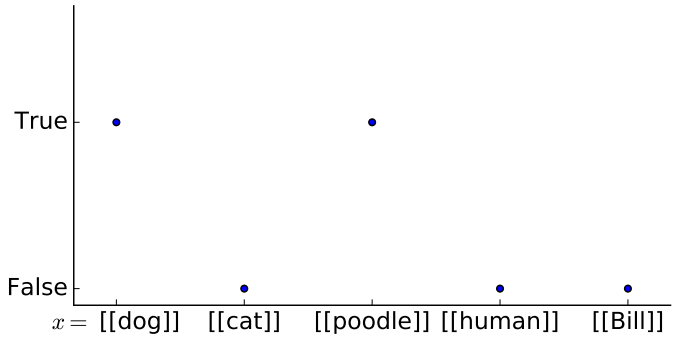


Figure 3.2: A visualization of the denotation of *barks*. The x axis corresponds to denotations of nouns (i.e., sets of entities in the world); the y axis is the domain of truth values.

Importantly, we can keep composing functions from known domains and form new domains corresponding to these new functions. For example, if the domain of intransitive functions defined above is $\mathcal{D}_f$, we can define a class of transitive functions as $f : \mathcal{D}_e \to \mathcal{D}_f$. We can alternately write this as $f : \mathcal{D}_e \to (\mathcal{D}_e \to \mathcal{D}_t)$. The denotation of any span of text is therefore either an entity ($\mathcal{D}_e$), a truth value ($\mathcal{D}_t$), or some class of function inductively defined above.

Notational Note It becomes tedious to write long functions as $\mathcal{D}_e \to (\mathcal{D}_e \to \mathcal{D}_t)$; not to mention defining a new set for every function type, as we did for $\mathcal{D}_f$. Therefore, from now on, we'll denote the set of entities as $e = \mathcal{D}_e$, the set of truth values as $t = \mathcal{D}_t$, and functions in the usual way of $a \to b$. In this notational scheme, intransitive verbs are written as $e \to t$; transitive verbs are $e \to (e \to t)$, and so forth.

3.1.4 Quantifiers (Operators) are Functions

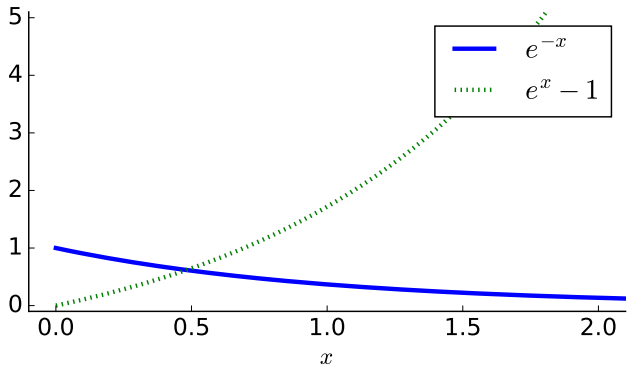
An important class of lexical items are quantifiers (and more generally natural language operators). From a denotational semantics point of view, these behave just like any other

function: *all* has the same denotation as a transitive verb: $e \to (e \to t)$; *not* has the same denotation as an intransitive verb: $e \to t$. However, natural language operators has an important additional property: they are often *monotonic* functions. This property is the happy accident of language that underpins much of the usefulness of monotonicity calculus as a natural logic, and is the main topic of the next section.

3.2 Monotonicity Calculus

We begin this section by reviewing monotonicity in its familiar context: monotonicity of algebraic functions. For example, $f(x) = e^x - 1$ is a *monotone* function – as we increase x, the value of $f(x)$ monotonically increases. A function may also be *antitone*: $f(x) = e^{-x}$ is antitone, since the value of $f(x)$ decreases monotonically as x increases. Lastly, a function can be *nonmonotone* – distinct from antitone – if it is neither monotone nor antitone.

Visually, the plots below show a monotone function ($e^x - 1$) and an antitone function (e^{-x}). We will appeal to analogous visualizations when we move towards working with language.



Monotonicity is an appealing tool because it lets us reason about functions without having to evaluate them. To illustrate, we can define an arbitrarily complex function $f : \mathbb{R} \to \mathbb{R}$, which we are told is monotone. Without evaluating the function, we are able to

conclude that $f(x+1) > f(x)$. This is precisely the type of tool we would like to use to manipulate language: constructing a concrete interpretation of language – like evaluating a complex function – is at best undesirable and at worst impossible. However, if we know some properties about the "monotonicity" of the language, we can manipulate the text such that we preserve some key relations between the original and our mutated text – analogous to the greater-than relation in our algebraic example.

This analogy is much more direct than it may at first appear: we defined a class of functions in Sections 3.1.3 and 3.1.4, and monotonicity calculus will be the calculus of valid inferences that we can draw from reasoning about the monotonicity of these functions. The remainder of this section will explore how to apply monotonicity to the denotational semantics in Section 3.1, and then introduce reasoning about exclusion (Section 3.2.2). The section will conclude by introducing the notion of *polarity*, and exploring how to compose monotone functions in a sentence.

3.2.1 Monotonicity in Language

In generality, a monotone function is a function between partially-ordered sets that preserves the given order. For a function f with domain $\mathbf{X}$ and range $\mathbf{Y}$, we define a partial order over $\leq_X$ and $\leq_Y$. This function is monotone iff:

$$\forall x_1, x_2 \in \mathbf{X} \text{ such that } x_1 \leq_X x_2; \;\; f(x_1) \leq_Y f(x_2) \tag{3.1}$$

We note from Section 3.1 that, by and large, sentences are constructed by composing one or more functions (e.g., verbs, operators). To reason about whether these functions are monotone (or, by extenstion, antitone), we need to show that each of our domains forms a partial order we can define monotonicity against.

First: the domain of noun denotations: $\mathcal{D}_e$ (or, e). We define our partial order $\leq_e$ to be the subset operator: $\subseteq$. That is, if the denotation of a word is completely contained in the denotation of another word, we consider the first word to be "less than" the second. This is intuitively encoding hypernymy as a partial order. For example, $[\![cat]\!] \leq_e [\![feline]\!]$ because any entity which is a cat is also necessarily a feline.

Second: the domain of truth values: $\mathcal{D}_t$ (or, t). Here, we axiomatically define a partial

order $\leq_t$ to be:[1]

$$false \leq_t false$$
$$false \leq_t true$$
$$true \leq_t true$$

The very important observation to make at this point is that **the partial order $\leq_t$ corresponds exactly to the material conditional** $\supset$. So, for any two propositions A and B, A entails B ($A \supset B$) is the same as $A \leq_t B$. This is the key insight tying together the concepts of monotonicity and entailment. We also point out the unfortunate fact that the symbol for the material conditional is the same as the superset symbol, which would seem to evoke "greater than" rather than the "less than" it actually corresponds to. Unfortunately, this dissertation is not the place to redefine either of these symbols.

Lastly: we must define a partial order over our inductively defined function types. A function is less than another function if, for all values in the domain of the functions, the value of the first function is less than the value of the second. Formally: for two functions f and g with the same domain and range $\mathbf{X} \rightarrow \mathbf{Y}$, we say $f \leq_f g$ iff:

$$\forall x \in \mathbf{X};\ \ f(x) \leq_Y g(x) \tag{3.2}$$

For the remainder of this thesis, we will collapse all of these partial orders – $\leq_e$, $\leq_t$, and $\leq_f$ – into a single symbol: $\sqsubseteq$.

Monotonicity is Entailment-Preserving The most important insight from the partial orders above is that our partial order over truth values corresponds exactly to entailment. Although "entailment" is not particularly well-defined for the other denotation types (i.e., does *cat* entail *animal*?), for the purposes of monotonicity calculus and natural logic we

[1]The elegance of the interpretation of false as the empty set and true as a singleton set becomes clear here: we can define the partial order over truth values to be the same subset operator as the partial order over noun denotations.

will take the symbol $\sqsubseteq$ to be entailment.[2] By extension, we can define $A \sqsupseteq B$ to mean that $B \sqsubseteq A$, and define $\equiv$ to be equivalence. That is to say, $A \equiv B$ is the same as $A \sqsubseteq B$ and $B \sqsubseteq A$.

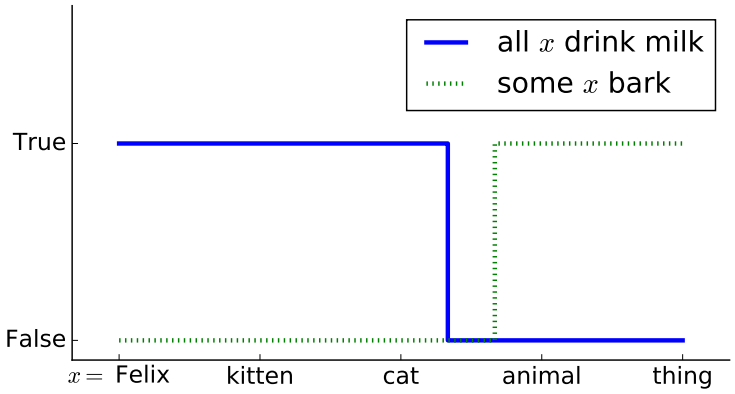
This means that monotone functions are entailment preserving. If a sentence is true, and the function used to construct its denotation (i.e., truth) is monotone with respect to the denotation of a word, then replacing that word with another word whose denotation is a superset of the original word will maintain the truth of the sentence. Taking a concrete example: *all* (a function $e \to (e \to t)$) is antitone in its first argument and monotone in its second. So, the sentence *all cats drink milk* is antitone with respect to *cats* and monotone with respect to *drink milk*. Furthermore, we know that $[\![drink\ milk]\!] \sqsubseteq [\![drink\ dairy]\!]$ because $[\![milk]\!] \sqsubseteq [\![dairy]\!]$. Therefore, by the definition of monotonicity, we can replace *drink milk* with *drink dairy* and the resulting sentence (*all cats drink dairy*) is guaranteed to be true if the original sentence was true.

The fact that quantifiers and other operators in natural language have this property of monotonicity is wonderfully convenient. Grounding an interpretation for *all cats drink milk* would be rather difficult in the general case – and certainly difficult for longer utterances. But by appealing to the monotonicity of quantifiers, we do not need to ground the sentence to run an entailment proof on it. Antitone functions behave analogously to monotone functions, but with the direction of the lexical mutation reversed. For example, if *all cats drink milk*, we can infer that *all kittens drink milk* because $[\![cat]\!] \sqsupseteq [\![kitten]\!]$.

Like the visualization with monotone algebraic functions earlier in the section, we can visualize monotonicity over denotations. In the chart below, the x axis is an ordering over denotations.[3] The y axis is the ordering over truth values. We are plotting two functions: *all* x *drink milk* – antitone in x; and *some* x *bark* – monotone in x:

[2] This is, again, not to be confused with the symbol for entailment in propositional logic: $\supset$.

[3] In general this is a partial order; however, partial orders are difficult to plot on an axis.



Monotonicity Calculus as a proof theory At this point, we can begin looking at monotonicity calculus as the sort of natural logic proof system we demonstrated at the beginning of the chapter. For example, our inference from *the cat ate a mouse* to *the carnivore ate an animal* could be formally justified with the following proof. We note that the quantifier *the* – like *some* – is monotone in both of its arguments.

1	*The cat ate a mouse.*	
2	*The cat ate a rodent.*	$[\![\textit{mouse}]\!] \sqsubseteq [\![\textit{rodent}]\!]$, 1
3	*The cat ate an animal.*	$[\![\textit{rodent}]\!] \sqsubseteq [\![\textit{animal}]\!]$, 2
4	*The feline ate an animal.*	$[\![\textit{cat}]\!] \sqsubseteq [\![\textit{feline}]\!]$, 3
5	*The carnivore ate an animal.*	$[\![\textit{carnivore}]\!] \sqsubseteq [\![\textit{animal}]\!]$, 4

However, we still lack the tools to infer that *no carnivore ate an animal* is false given our premise. For this, we need a theory of how to reason with *exclusion* – which we will review in the next section. Furthermore, our theory currently does not handle nested quantification. In Section 3.2.4 we introduce *Polarity* and the mechanism for propagating monotonicity information to determine the "monotonicity" of a sentence composed of multiple quantifiers.

3.2.2 Exclusion

Although the monotonicity calculus of Sánchez Valencia [90] can already do a range of interesting inferences, it is nonetheless still a very restricted logic. This section reviews work by MacCartney and Manning [71] and Icard and Moss [53] on how natural logic can be extended to handle negation and antonomy by reasoning about *exclusion*. Much of the notation is adapted from Icard and Moss [53].

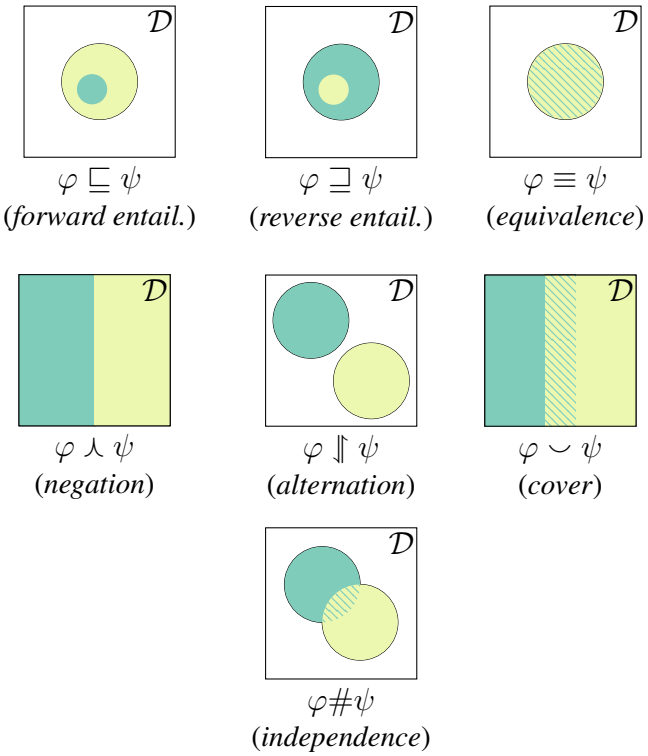


Figure 3.3: An enumeration of the possible relations between two sets φ (dark green) and ψ (light yellow). The top three relations are the simple relations used in monotonicity calculus; the middle three are relevant for exclusion; the bottom relation denotes the case where nothing of interest can be said about the two sets.

The foundations for exclusion come from the observation that there are more relations you can define between sets than the subset / superset / equality relations used in the monotonicity calculus described in Section 3.2.1. Given a set φ and a set ψ, these relations are enumerated in Figure 3.3. The top row correspond to the relations we are familiar with

($\sqsubseteq$, $\sqsupseteq$, $\equiv$) along with their interpretation in natural logic. The middle row describes the three relations relevant for exclusion. The bottom row describes the independence relation, meaning that nothing of interest can be said about the two relations.

We review each of these four new relations, and their interpretation for our three entity types: denotations, truth values, and functions.

First, however; to extend these definition beyond sets, and therefore beyond denotations to truth values and to functions, we introduce a bit of notation. In particular, we define a *partially distributive lattice*. This is a 5-tuple: $(\mathbf{D}, \vee, \wedge, \bot, \top)$, consisting of a domain $\mathbf{D}$ (the set of entities in the lattice), two binary operators $\vee$ and $\wedge$ corresponding to a generalized sense of *maximum* and *minimum* respectively,[4] and two elements in $\mathbf{D}$, $\bot$ and $\top$, corresponding intuitively to the smallest and largest element of $\mathbf{D}$, respectively.

For denotations, we define this lattice as follows, mirroring previous sections.

$\mathbf{D}$ is the power set of all denotations in $\mathcal{D}_e$.
$\vee$ is the union operator: $\cup$.
$\wedge$ is the intersect operator: $\cap$.
$\bot$ is the empty set: $\{\}$.
$\top$ is the full domain of denotations: $\mathcal{D}_e$.

For truth values, we define the lattice straightforwardly as:

$\mathbf{D}$ is the set $\{0, 1\}$, where 0 corresponds to false and 1 corresponds to true.
$\vee$ maximum (i.e., $1 \vee 0 = 1$)
$\wedge$ minimum (i.e., $1 \wedge 0 = 0$)
$\bot$ is false: 0.
$\top$ is true: 1.

Defining this lattice over functions is a bit more verbose, but intuitively analogous to the denotation and truth value cases. Since the range of our functions is ordered (e.g., the domain of truth values t is ordered in a function $e \rightarrow t$), and the range has a maximum and minimum element (e.g., true and false respectively for t), we can from this define our "largest" and "smallest" functions $\top$ and $\bot$. $\top$ is the function which takes any element in

[4] $\vee$ and $\wedge$ must also be commutative, associative, and idempotent; and they must distribute over each other.

its domain, and maps it to the maximum element in the function's range. As a concrete example, for a function $e \to t$ this corresponds to a tautology – e.g., x *is* x maps everything to $\top$ in the domain of truth values (i.e., *true*). The function corresponding to $\bot$, conversely, maps every element in its domain to the smallest element (i.e., $\bot$) in the function's range.

We further define the $\vee$ and $\wedge$ of two function to be the element-wise $\vee$ and $\wedge$ of the functions. That is, for functions $f : A \to B$ and $g : A \to B$, $h = f \vee g$ iff $\forall x, h(x) = f(x) \vee g(x)$. $f \vee g$ is defined analogously. To summarize, the lattice for a function $f : A \to B$ is:

$\mathbf{D}$ is the set of all functions from A to B.
$\vee$ is the element-wise $\vee$ of the functions.
$\wedge$ is the element-wise $\wedge$ of the functions.
$\bot$ is the function mapping any A to $\bot$ in B.
$\bot$ is the function mapping any A to $\top$ in B.

We can use our definition of this lattice to formally define the new relations in Figure 3.3, and give new interpretations to the relations from above.

Negation (人) From Figure 3.3, two sets are in *negation* if the union of the sets is the entire domain, and the intersection of the sets is empty. That is, for two sets φ and ϕ, $\varphi \cup \psi = \mathcal{D}$ and $\varphi \cap \psi = \{\}$. Generalizing this to our lattice definition above, we say that two terms are in negation with each other iff $x \vee y = \top$ and $x \wedge y = \bot$. As the name would imply, the most natural examples of negation appearing for pairs of denotations usually involve some sort of morphological negation:

$[\![$*cat*$]\!]$ 人 $[\![$*noncat*$]\!]$
$[\![$*living thing*$]\!]$ 人 $[\![$*nonliving thing*$]\!]$
$[\![$*possible thing*$]\!]$ 人 $[\![$*impossible thing*$]\!]$

For truth values, negation (unsurprisingly) corresponds to logical negation. We can recover the truth table for negation straightforwardly from the definition of the lattice, recalling that $\vee$ is $\max$, $\wedge$ is $\min$, $\top$ is 1, and $\bot$ is 0:

x	y	$x \vee y$	$x \wedge y$	$x \vee y = \top$ and $x \wedge y = \bot$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

The definition of negation for functions likewise follows. Two functions f and g are then negations of each other ($f \curlywedge g$) iff elementwise $\max(f, g)$ (i.e., $f \vee y$) always maps to $\top$, and $\min(f, g)$ (i.e., $f \wedge y$) always maps to $\bot$. To illustrate with a concrete example, the functions x *is living* and x *is nonliving* are in negation, since for any x it is either true that it is living, or it's true that it is nonliving; and, for any x, it is it is never true that it is both living and nonliving. This extends trivially to quantifiers. For example, *no* and *some* are negations of each other.

The next two relations – alternation ($\Downarrow\!\Uparrow$) and cover ($\smile$) – can be though of as holding one, but not both of the conditions of negation. In particular, two entities in the negation relation are also necessarily in the alternation and cover relations.

Alternation ($\Downarrow\!\Uparrow$) Alternation can be thought of as another form of negation, which is weaker in some important respects which will become clear later. In the generalized case, two entities are in alternation iff $x \wedge y = \bot$.

Two denotations are in *alternation* if their intersection is empty. That is, for sets φ and ψ, $\varphi \cap \psi = \{\}$, but unlike negation we do not know anything about their union. This is commonly the relation which holds between antonyms and otherwise contradictory nouns:

$[\![\textit{cat}]\!] \Downarrow\!\Uparrow [\![\textit{dog}]\!]$
$[\![\textit{genius}]\!] \Downarrow\!\Uparrow [\![\textit{idiot}]\!]$
$[\![\textit{good deed}]\!] \Downarrow\!\Uparrow [\![\textit{bad deed}]\!]$

For truth values, alternation equates pragmatically to negation in the context of proving entailment. That is, false $\Downarrow\!\Uparrow$ false is true (whereas false $\curlywedge$ false is false); however, this is only relevant if we are assuming that our premise is false. Since we are (axiomatically) assuming

that our premise is true, this case will never arise, and the truth table looks otherwise equivalent to full negation:

x	y	$x \vee y$	$x \wedge y$	$x \wedge y = \bot$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

The intuition for when functions are in alternation becomes potentially hairy, but not awful. Adjective antonyms are a clear example: *hot* $x \Downarrow$ *cold* x, since for any x we know that x is not both hot and cold. The quantifiers *all* and *no* are similarly in alternation.

Cover ($\smile$) In many ways, cover is the most strange of the relations in this section. For *nearly* all intents and purposes, this relation indicates neither entailment nor negation between its two entities, but it occasionally conveys a hint of negation. Concretely, cover behaves as negation when reasoning about a counter-factual premise – e.g., if in an entailment chain you have negated your premise – and are now continuing to reason about this presumed false intermediate statement. Formally, two entities are in the cover relation iff $x \vee y = \top$.

For denotations, this is a quintessentially rare case ($\varphi \cup \psi = \mathcal{D}$), and examples which don't amount to outright negation are almost always a bit contrived:

$[\![\textit{animal}]\!] \smile [\![\textit{non-cat}]\!]$
$[\![\textit{smartphone}]\!] \smile [\![\textit{non-iphone}]\!]$

The behavior of the cover relation becomes a bit more apparent in the case of truth values. Analogous to how alternation ($\Downarrow$) was pragmatically negation when the premise is true, cover is pragmatically negation when the premise is *false*. We will, of course, never assume that the premise in a proof is false; but certainly intermediate steps in the proof may find us with a presumed false statement. In these cases, the cover relation allows us to "negate" this false statement.

x	y	$x \vee y$	$x \wedge y$	$x \vee y = \top$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

The cover relation is virtually unseen for functions, although of course the definition still carries over fine.

Independence ($\#$) The last relation is independence, which corresponds to no relation holding purely by virtue of the constructed lattice. This is the case for, e.g.,

$[\![\textit{cat}]\!] \# [\![\textit{black animal}]\!]$
$[\![\textit{happy}]\!] \# [\![\textit{excited}]\!]$
$[\![\textit{play}]\!] \# [\![\textit{run}]\!]$

The old relations ($\sqsubseteq$, $\sqsupseteq$, $\equiv$) The relations from the simple monotonicity calculus in Section 3.2.1 of course also have generalized interpretations in the context of our lattice. The behavior remains identical to before. The generalized definitions are:

$x \sqsubseteq y$ iff $x \wedge y = x$
$x \sqsupseteq y$ iff $x \vee y = x$
$x \equiv y$ iff $x \wedge y = x$ and $x \vee y = x$

3.2.3 Proofs with Exclusion

We showed how to run simple proofs in monotonicity calculus in Section 3.2.1 by simply appealing to the transitivity of the $\sqsubseteq$ relation. In effect, we implicitly defined a transitivity table, or *join table*, for how to assess the relation between x and z if we know the relation between x and y, and between y and z. We then *join* these two relations together with the $\bowtie$ operator defined below to obtain the final relation between x and z:

$\bowtie$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\curlyvee$	$\#$
$\equiv$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\curlyvee$	$\#$
$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\#$	$\Downarrow$	$\Downarrow$	$\#$	$\#$
$\sqsupseteq$	$\sqsupseteq$	$\#$	$\sqsupseteq$	$\curlyvee$	$\#$	$\curlyvee$	$\#$
$\curlywedge$	$\curlywedge$	$\curlyvee$	$\Downarrow$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$	$\#$
$\Downarrow$	$\Downarrow$	$\#$	$\Downarrow$	$\sqsubseteq$	$\#$	$\sqsubseteq$	$\#$
$\curlyvee$	$\curlyvee$	$\curlyvee$	$\#$	$\sqsupseteq$	$\sqsupseteq$	$\#$	$\#$
$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$

Table 3.1: The join table as taken from Icard [52].
The join table as taken from Icard [52]. Entries in the table are the result of joining a row with a column. Note that the $\#$ always joins to yield $\#$, and $\equiv$ always joins to yield the input relation.

$\bowtie$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$
$\equiv$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$
$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\#$
$\sqsupseteq$	$\sqsupseteq$	$\#$	$\sqsupseteq$

As expected, we see the the transitivity of $\sqsubseteq$: this is the key property we needed to run our proofs. We can now define a similar (if larger) join table for our full set of relations. This table is given in Table 3.1.

However, a much more convenient representation of this join table is as a finite state machine. We further will show that we can losslessly collapse this finite state machine into only three intuitive inference states. These observations allow us to formulate a proof as a path through this collapsed state machine, making reasoning with exclusion almost as simple as the original monotonicity proofs.

We construct a finite state machine over states $s \in \{\sqsubseteq, \sqsupseteq, \dots\}$. A machine in state s_i corresponds to relation s_i holding between the initial premise and the derived fact so far. States therefore correspond to states of *logical validity*. The start state is $\equiv$. Outgoing transitions correspond to *inference steps*. Each transition is labeled with a projected relation $\rho(r) \in \{\sqsubseteq, \sqsupseteq, \dots\}$, and spans from a source state s to a target s' according to the join table. That is, the transition $s \xrightarrow{\rho(r)} s'$ exists iff $s' = s \bowtie \rho(r)$. Figure 3.4a shows the automaton, with trivial edges omitted for clarity.

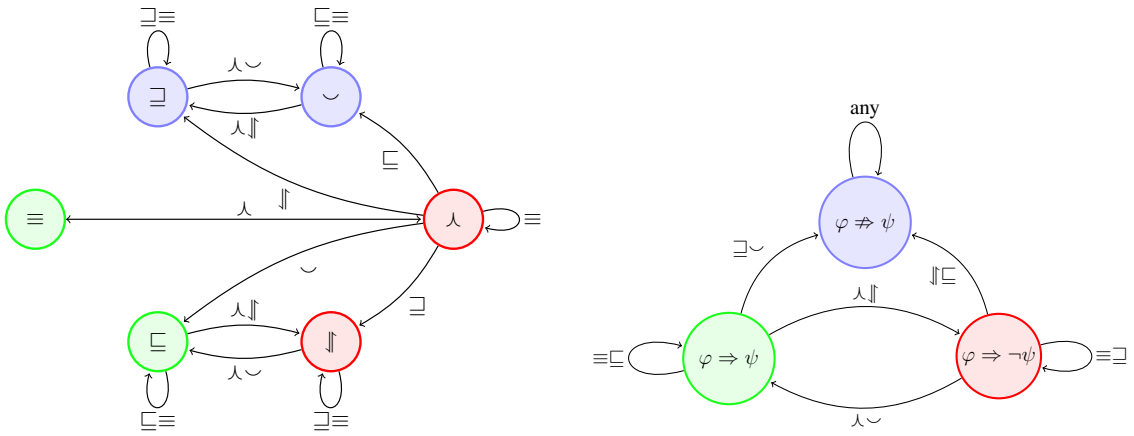


Figure 3.4: Natural logic inference expressed as a (possibly collapsed) finite state automaton.
(a) Natural logic inference expressed as a finite state automaton. Omitted edges go to the unknown state (#), with the exception of omitted edges from $\equiv$, which go to the state of the edge type. Green states ($\equiv$, $\sqsubseteq$) denote valid inferences; red states ($\Downarrow$, $\nwarrow$) denote invalid inferences; blue states ($\sqsupseteq$, $\smile$) denote inferences of unknown validity. (b) The join table collapsed into the three meaningful states over truth values.

We further collapse this automaton into the three meaningful states we use as output: *valid* ($\varphi \Rightarrow \psi$), *invalid* ($\varphi \Rightarrow \neg\psi$), and *unknown validity* ($\varphi \not\Rightarrow \psi$). We can cluster states in Figure 3.4a into these three categories. The relations $\equiv$ and $\sqsubseteq$ correspond to valid inferences; $\nwarrow$ and $\Downarrow$ correspond to invalid inferences; $\sqsupseteq$, $\smile$ and # correspond to unknown validity. This clustering mirrors that used by MacCartney for his textual entailment experiments.

Collapsing the FSA into the form in Figure 3.4b becomes straightforward from observing the regularities in Figure 3.4a. Nodes in the valid cluster transition to invalid nodes always and only on the relations $\nwarrow$ and $\Downarrow$. Symmetrically, invalid nodes transition to valid nodes always and only on $\nwarrow$ and $\smile$. A similar pattern holds for the other transitions.

Formally, for every relation r and nodes a_1 and a_2 in the same cluster, if we have transitions $a_1 \xrightarrow{r} b_1$ and $a_2 \xrightarrow{r} b_2$ then b_1 and b_2 are necessarily in the same cluster. As a concrete example, we can take $r = \nwarrow$ and the two states in the *invalid* cluster: $a_1 = \nwarrow$, $a_2 = \Downarrow$. Although $\nwarrow \xrightarrow{\nwarrow} \equiv$ and $\Downarrow \xrightarrow{\nwarrow} \sqsubseteq$, both $\equiv$ and $\sqsubseteq$ are in the same cluster (*valid*). It is not

trivial *a priori* that the join table should have this regularity, and it certainly simplifies the logic for inference tasks.

We can now return to our running example, augmented with negation, and prove that if *the cat ate a mouse* then it is false that *no carnivores ate an animal*. At each inference step, we note the transition we took to reach it form the previous statement, and the new state we are in.

1	*The cat ate a mouse.*		
2	*The cat ate a rodent.*	rel: $\sqsubseteq$	state:$\Rightarrow$, 1
3	*The cat ate an animal.*	rel: $\sqsubseteq$	state:$\Rightarrow$, 2
4	*The feline ate an animal.*	rel: $\sqsubseteq$	state:$\Rightarrow$, 3
5	*The carnivore ate an animal.*	rel: $\sqsubseteq$	state:$\Rightarrow$, 4
6	*No carnivore ate an animal.*	rel: $\curlywedge$	state:$\Rightarrow \neg$, 5

We then notice that the final state we end up in is $\Rightarrow \neg$ – that is, negation. Taking another example, to prove that if Spock is logical, then he is not very illogical:

1	*Spock is logical*		
2	*Spock is illogical*	rel: $\curlywedge$	state:$\Rightarrow \neg$, 1
3	*Spock is very illogical*	rel: $\sqsupseteq$	state:$\Rightarrow \neg$, 2
4	*Spock is not very illogical*	rel: $\curlywedge$	state:$\Rightarrow$, 3

A few final observations deserve passing remark. First, even though the states $\sqsupseteq$ and $\smile$ appear meaningful, in fact there is no "escaping" these states to either a valid or invalid inference. Second, the hierarchy over relations presented in Section 3.2.2 becomes even more apparent – in particular, $\curlywedge$ always behaves as negation, whereas its two "weaker" versions ($\Downarrow$ and $\smile$) only behave as negation in certain contexts. Lastly, with probabilistic inference, transitioning to the unknown state can be replaced with staying in the current state at a (potentially arbitrarily large) cost to the confidence of validity. This allows us to make use of only two states: *valid* and *invalid*.

3.2.4 Polarity: Composing Monotonicity

So far, we have been talking primarily about natural logic relations between lexical items. That is, $[\![cat]\!] \sqsubseteq [\![animal]\!]$, and $[\![black\ cat]\!] \sqsubseteq [\![cat]\!]$, and $[\![happy]\!] \between [\![sad]\!]$, and true $\sqsubseteq$ true, and so forth. For our proofs over sentences, we have limited ourselves to a single quantifier, which has allowed us to reason about inferences directly based off of the monotonicity of the quantifier. In this section, we explore two important concepts to complete our theory of monotonicity calculus: we describe how we compose monotonicity when a lexical item is under the scope of multiple quantifiers, and we formally characterize how we can account for apparent nuances between the behaviors of different quantifiers when dealing with exclusion.

To motivate the discussion, we can consider a simple inference:

1	*No cats don't eat meat*	
2	*No cats don't eat food*	$[\![meat]\!] \sqsubseteq [\![food]\!]$, 1

Here, *meat* is under the scope of two quantifiers: *no* and *n't* (not).[5] Recall that both of these quantifiers are downward monotone with respect to both of their arguments. Were the word under the scope of a single downward monotone quantifier, the relation $\sqsubseteq$ would not be entailment-preserving. For example, when under the scope of a single negation, the following inference is invalid:

1	*No mice eat meat*	
2	*No mice eat food*	$[\![meat]\!] \sqsubseteq [\![food]\!]$, 1

To address this, we introduce a notion of *polarity*. Polarity is a property we assign to lexical items. It can be thought of as a function which takes as input a relation which holds between a lexical item and its mutation, and produces as output the relation that holds between the containing sentences. For instance, the polarity of *meat* in the sentence *No mice eat meat* would be a function which translates $\sqsubseteq$ to $\sqsupseteq$ and $\sqsupseteq$ to $\sqsubseteq$. The polarity of *meat* in the sentence *No cats don't eat meat* would be a function translating $\sqsubseteq$ to $\sqsubseteq$ and

[5] Technically, we are abusing terminology; not all quantifiers in natural logic are quantifiers in the linguistic sense.

$\sqsupseteq$ to $\sqsupseteq$. This is no different in nature than monotonicity, and is in fact no more than the composition of the monotonicities of the quantifiers acting on a lexical item.

The algorithm for determining polarity is simple. Beginning from the lexical item in question, we list the quantifiers which have that lexical item in their scope, and order that list from narrowest to broadest scope. This gives us a list $q_0, q_1, \ldots q_n$. In our example, *No cats don't eat meat*, the ordering for *meat* would be $q_0 =$ *n't* and $q_1 =$ *no*. We begin with the identity function as our polarity; then, for each quantifier q_i, we compose our polarity so far with that function's monotonicity.

In the simplest case, this takes the form of flipping an item's polarity between *upward* to *downward* and visa versa for every downward monotone quantifier in its scope. For our double negation case above, we begin with an upward polarity, and flip the polarity twice (once for *no* and once for *n't*) to arrive back at an upward polarity context. As we'll see in the next section, this process becomes more nuanced once we get into the exclusion relations ($\Downarrow$, $\curlywedge$, $\smile$), but for now this intuition is enough to formulate a complete proof theory for monotonicity calculus.

Following a variant of the notation in MacCartney and Manning [71], we can construct a natural logic proof as a table. Each row corresponds to a single lexical mutation on the previous row; the first row is the premise fact. The first column is the hypothesis. The second column is the lexical relation induced by the mutation performed to obtain the given row. The third column is this relation *projected* up the sentence, based on the lexical item's polarity. The last column is the truth state of the proof, as determined by the previous proof state and the projected relation (see Figure 3.4). To make this concrete, Table 3.2 shows an example inference from *no cats don't eat meat* negating that *black cats don't eat food*.

3.2.5 Additive and Multiplicative Quantifiers

The last topic for this section deals with polarity when we are projecting one of the exclusion relations through the sentence. For example, based on the proof theory in Section 3.2.4, if we assume that the exclusion relations propagate up with the identity relation, we'd get incorrect entailments like the following:

Sentence	Lexical Rel	Projected Rel	Truth
No cats don't eat meat			$\Rightarrow$
No cats don't eat ***food***	$\sqsubseteq$	$\sqsubseteq$	$\Rightarrow$
No ***black cats*** *don't eat food*	$\sqsupseteq$	$\sqsubseteq$	$\Rightarrow$
Some *black cats don't eat food*	$\curlywedge$	$\curlywedge$	$\Rightarrow \neg$

Table 3.2: A tabular natural logic proof negating *no carnivores eat animals* from *the cat ate a mouse*.

A tabular proof negating *no carnivores eat animals* from *the cat ate a mouse*. The first column tracks the sentence as it mutates. The second column tracks the lexical natural logic relation induced by the mutation. The third column tracks the lexical relation projected up the sentence. The last column tracks the truth state of the proof, as determined by the FSA in Figure 3.4.

Sentence	Lexical Rel	Projected Rel	Truth
Every cat has a tail			$\Rightarrow$
Every ***dog*** *has a tail*	$\between$	$\between$	$\Rightarrow \neg$

That is to say, despite the fact that $[\![cat]\!] \between [\![dog]\!]$, it's not the case that every cat having a tail contradicts every dog having a tail. To understand why, we introduce the notion of *multiplicative* and *additive* quantifiers, described in more detail in Icard and Moss [53].

Recall that quantifiers are simply functions that map from, e.g., denotations to truth values. Recall further that each of our domains (denotations, truth values, etc.) can be described in terms of a partially distributive lattice: $(\mathbf{D}, \vee, \wedge, \bot, \top)$. The operator $\vee$ is intuitively a union operator, $\wedge$ is intuitively an intersect operator. An upwards monotone quantifier is then:

- *multiplicative* iff $f(x \wedge y) = f(x) \wedge f(y)$.
- *additive* iff $f(x \vee y) = f(x) \vee f(y)$.

Notice that a quantifier can be both additive and multiplicative, and can also be neither additive or multiplicative. Conversely, a downwards monotone quantifier can be anti-additive and anti-multiplicative:

	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\curlyvee$
upward	$\sqsubseteq$	$\sqsupseteq$	#	#	#
additive	$\sqsubseteq$	$\sqsupseteq$	$\curlyvee$	#	$\curlyvee$
multiplicative	$\sqsubseteq$	$\sqsupseteq$	$\Downarrow$	$\Downarrow$	#
additive+multiplicative	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\curlyvee$
downward	$\sqsupseteq$	$\sqsubseteq$	#	#	#
anti-additive	$\sqsupseteq$	$\sqsubseteq$	$\Downarrow$	#	$\Downarrow$
anti-multiplicative	$\sqsupseteq$	$\sqsubseteq$	$\curlyvee$	$\curlyvee$	#
anti-additive+anti-multiplicative	$\sqsupseteq$	$\sqsubseteq$	$\curlywedge$	$\curlyvee$	$\Downarrow$

Table 3.3: Monotonicity for quantifiers marked with additivity / multiplicativity information

The definition of the monotonicity for quantifiers marked with additivity / multiplicativity information. Given that the argument in the domain of the quantifier is mutated according to the relation in the header, the resulting element in the range of the quantifier is guaranteed to have mutated according to the relation in this table. That is, for example, for an additive quantifier $f : \mathcal{X} \to \mathcal{Y}$, if $x_0 \in \mathcal{X}$, $x_1 \in \mathcal{X}$, $f(x_0) = y_0$, $f(x_1) = y_1$, and $x_0 \curlywedge x_1$, then we can say that $y_0 \curlyvee y_1$.

- *anti-multiplicative* iff $f(x \vee y) = f(x) \vee f(y)$.
- *anti-additive* iff $f(x \wedge y) = f(x) \wedge f(y)$.

From our example above, we notice that according to this definition *every* is anti-additive in its first argument: *Every nook and cranny was searched* is equivalent to *every nook was searched* and *every cranny was searched*; but, *every Monday or Wednesday it rains* does implies neither that it rains every Monday, or that it rains every Wednesday.[6] We can also notice that the second argument of *every* is multiplicative: *every cat likes sleeping and eating* entails that *every cat likes sleeping* and *every cat likes eating*. In a similar vein, *no* is anti-additive in both of its arguments; *some* is both additive and multiplicative in both its arguments; and so forth.

Now that we have a more precise notion of monotonicity beyond *upward* and *downward*, we can more accurately characterize the effect quantifiers have on lexical relations

[6]Except on the reading of *Monday or Wednesday* as *Monday and Wednesday* – which is a separate, but quite interesting linguistic issue.

as they project up the sentence. We give this refined function in Table 3.3. Note that for the core monotonicity relations ($\sqsubseteq$ and $\sqsupseteq$), the additivity and multiplicativity of the quantifiers is irrelevant, but that these properties have a rather large effect on the other three meaningful relations. Another interesting observation is that the symmetry between cover ($\smile$) and alternation ($\Downarrow$) that we see in the FSA in Figure 3.4 also appears in the monotonicity function. The difference between additive and anti-additive, multiplicative and anti-multiplicative, etc., is simply replacing all instances of $\sqsubseteq$ with $\sqsupseteq$ (from the definition of monotonicity), and all instances of $\smile$ with $\Downarrow$, and visa versa.

If we return to our naive inference from the beginning of the section which incorrectly inferred that *evey cat has a tail* negates *every dog has a tail*, we can show that we obtain the correct result if we take into consideration that *every* is anti-additive in its first argument. In particular, alternation ($\Downarrow$) projects up to the independence relation ($\#$) when in the scope of a single anti-additive quantifier. Therefore, no inference can be drawn about the two sentences:

Sentence	**Lexical Rel**	**Projected Rel**	**Truth**
Every cat has a tail			$\Rightarrow$
Every ***dog*** *has a tail*	$\Downarrow$	$\#$	$\neg \Rightarrow$

This concludes our exploration of monotonicity calculus. The subsequent section explores some ideas in extending natural logics to propositional reasoning; the remainder of the dissertation thereafter will explore applications of natural logic to large-scale open-domain reasoning problems.

3.3 A Propositional Natural Logic

So far all of our natural logic proofs have had a single premise. This is not accidental – a key shortcoming of monotonicity calculus and modern natural logics is that by their very nature they operate on a single premise. That premise is mutated in well-specified ways which preserve entailment; but there's no theory of how to combine multiple premises together for an inference. For example, the disjunctive syllogism is not supported by natural logic:

1	*Either Bill or Ted stole the cookies.*
2	*Ted did not steal the cookies.*
3	*Bill stole the cookies.*

This section sketches a theoretical approach for performing these inferences by taking a hybrid propositional and natural logic. In this way, we can leverage the relative strengths of both formalisms. Propositional logic has rich notions of the common connectives: conjunction, disjunction, etc.; monotonicity calculus is complementary to propositional logic in its handling of quantifiers and lexical entailment. At a high level, we would like a propositional logic to handle inferences between natural language propositions, while deferring to a natural logic to handle the entailments within propositions.

We briefly review propositional logic, and then show how we can perform hybrid natural logic and propositional proofs.

3.3.1 Propositional Logic

In propositional logic, we have propositions (usually denoted by capital letters), and connectives between these propositions. A proposition has a truth value in a given model; propositions are things like "cats have tails" or "the sky is blue". A special proposition denotes contradiction ($\bot$) – an axiomatically false statement. The connectives in propositional logic are a subset of the connectives in first order logic: conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$). Propositional logic does not have predicates (e.g., $P(x)$), and we will not be considering equality and identity (i.e., $P = Q$ or $x = y$). Furthermore, for simplicity, we will treat material implication axiomatically in terms of the core connectives:

$$\neg A \vee B \equiv A \rightarrow B$$

This section is not intended to be a formal introduction to propositional proofs, but rather a review of the *natural deduction* system and Fitch-style proofs we will use in the remainder of this section [43, 44, 54]. In particular, for each of our connectives (and the contradiction symbol), we define a rule for when we can *introduce* that connective, and

when we can *eliminate* the connective. In the next section, we will show how we can augment these rules with monotonicity calculus.

Conjunction Rules ($\wedge$) The two rules for conjunction are quite straightforward. We can always take as true either side of a conjunction, and we can introduce a conjunction only if we can prove both conjuncts:

$$\begin{array}{lll} 1 & A \wedge B & \\ \hline 2 & A & \wedge \text{E}, 1 \end{array} \qquad \begin{array}{lll} 1 & A \wedge B & \\ \hline 2 & B & \wedge \text{E}, 1 \end{array}$$

$\wedge$ elimination

$$\begin{array}{lll} 1 & A & \\ 2 & B & \\ \hline 3 & A \wedge B & \wedge \text{I}, 1, 2 \end{array}$$

$\wedge$ introduction

Disjunction Rules ($\vee$) Conjunction is, of course, quite boring on its own. Disjunction is a measure more interesting. Introducing disjunction is straightforward: if we know A, we also know that either A or B. However, eliminating a disjunction requires reasoning by cases. We have to show that the same formula is true if it is true for all the disjuncts. Formally, the rules are:

1	$A \vee B$	
2	$\quad A$	
$\vdots$	$\quad \vdots$	
n	$\quad Q$	
$n+1$	$\quad B$	
$\vdots$	$\quad \vdots$	
m	$\quad Q$	
$m+1$	Q	$\vee$ E, n, m

$\vee$ elimination

1	A	
2	$A \vee B$	$\vee$ I, 1

$\vee$ introduction

1	$\neg\neg A$	
2	A	$\neg$ E, 1

$\neg$ elimination

1	$\quad A$	
$\vdots$	$\quad \vdots$	
n	$\quad \bot$	
$n+1$	$\neg A$	$\neg$ I, 1–n

$\neg$ introduction

Negation Rules ($\neg$) Negation elimination is straightforward: we can eliminate double negations. Negation can be introduced, in turn, via a proof by contradiction:

Contradiction Rules ($\bot$) Contradiction can be introduced when we derive a proposition and its negation. Contradiction elimination is clearly the least intuitive of the deduction rules: if we have derived a contradiction, we can state any proposition as true. The Fitch rules for contradiction are:

$$\begin{array}{l|ll} 1 & \bot & \\ \hline 2 & Q & \bot \text{ E}, 1 \end{array}$$

¬ elimination

$$\begin{array}{l|ll} 1 & A & \\ \hline \vdots & \vdots & \\ n & \neg A & \\ n+1 & \bot & \bot \text{ I}, 1, n \end{array}$$

¬ introduction

The next section shows how we can adapt these rules to the incorporate natural logic.

3.3.2 A Hybrid Logic

In the previous section, we treated propositions as atomic units. A formula $A \vee B$ has no substructure beyond being a disjunction of two propositions. However, in most cases, these propositions *do* in fact have structure. Furthermore, most of the time we can express a proposition as a sentence. In this section we propose an extension to propositional logic which uses monotonicity calculus to augment the semantics of these atomic propositions (conversely: an extension to monotonicity calculus which incorporates propositional reasoning).

Recall that in natural logic, a sentence is represented as a truth value. Whereas $[\![\textit{cat}]\!]$ is a set of entities which are cats, and verbs like $[\![\textit{run}]\!]$ are denoted as functions from sets to truth values, a complete sentence (e.g., $[\![\textit{cats run}]\!]$) is either *true* or *false*. We can treat any denotation in natural logic which falls into this domain of {true, false} as propositions for propositional reasoning. From this point, the propositional logic side of our proof theory no longer has to care about the substructure of the sentence, and can treat it as an atomic proposition.

Notationally, we define the space of well formed formulas in our hybrid logic analogously to how a formula is defined in propositional logic:

1. $[\![x]\!]$ is a well-formed formula iff $[\![x]\!] \in \{true, false\}$. $[\![\textit{cat}]\!]$ is not a well-formed formula, but $[\![\textit{cats have tails}]\!]$ is.

2. $\bot$ is a well-formed formula.

3. If F is a well-formed formula, so is $\neg F$.

4. If F and G are well-formed formulas, so is $F \wedge G$.

5. If F and G are well-formed formulas, so is $F \vee G$.

Having defined our logical language, we now turn to our hybrid proof theory. Trivially, all of the inference rules of propositional logic hold, if we do not mutate any of the atomic sentences. Likewise, trivially all of the natural logic inferences hold, if they are not embedded in a larger propositional formula. Our task is then twofold: (1) what natural logic inferences are warranted inside of a given propositional formula; and (2) what *additional* propositional inferences can be made by appealing to the semantics of natural logic.

The first of these questions is answered by noticing that the propositional connectives, like natural language quantifiers, have monotonicity. Conjunction ($\wedge$) is upward monotone, and both additive and multiplicative; disjunction ($\vee$) is upward monotone, but only additive. Conjunction is multiplicative because $(A \wedge B) \wedge C \vDash (A \wedge C) \wedge (B \wedge C)$, and additive because $(A \vee B) \wedge C \vDash (A \wedge C) \vee (B \wedge C)$. Analogously, disjunction is not multiplicative: $(A \wedge B) \vee C \nvDash (A \wedge C) \vee (B \wedge C)$; but it is additive: $(A \vee B) \vee C \vDash (A \vee C) \vee (B \vee C)$. Negation, in turn, is downward monotone, and anti-multiplicative but not anti-additive: $\neg(A \vee B) \vDash \neg A \vee \neg B$, but $\neg(A \wedge B) \nvDash \neg A \wedge \neg B$.

From here, we can use our normal projection rules to mutate lexical items not only in a sentence, but also embedded in a propositional formula. For example, the following is now a valid inference using only natural logic:

1	$\neg[\![\textit{all cats are friendly}]\!] \wedge [\![\textit{all cats are cute}]\!]$	
2	$\neg[\![\textit{all } \textbf{\textit{felines}} \textit{ are friendly}]\!] \wedge [\![\textit{all cats are cute}]\!]$	$[\![\textit{cat}]\!] \sqsubseteq [\![\textit{feline}]\!]$, 1
3	$\neg[\![\textit{all felines are friendly}]\!] \wedge [\![\textit{all } \textbf{\textit{tabby cats}} \textit{ are cute}]\!]$	$[\![\textit{cat}]\!] \sqsupseteq [\![\textit{tabby cat}]\!]$, 2

Of course, we should be able to now apply the propositional rules from Section 3.3.1:

4	$[\![\textit{all tabby cats are cute}]\!]$	$\wedge$ E, 3

Perhaps the most interesting question is, what *additional* inferences can we draw from this hybrid logic? In particular, our two logics each have their own notion of negation: in monotonicity calculus, ⅄, ⇂, and ⌣ each behave a bit like negation. In fact, this lets us define a new negation introduction rule in propositional logic. In short, if we derive a new sentence from a presupposed true premise, and it is in the natural logic relations ⅄ or ⇂ with the original sentence, we can introduce the negation of the original sentence. Analogously, if we derive a new sentence from a presupposed false premise, and it is in the natural logic relations ⅄ or ⌣ with the original sentence, we can introduce the negation of the original sentence (to yield a double negation). This falls directly out of the state formulation of the finite state automata described in Figure 3.4. Formally:

1	$[\![X]\!]$	state:⇒
⋮	⋮	
n	$[\![Y]\!]$	state:¬ ⇒, 1–n
$n+1$	$\neg[\![X]\!]$	¬ I, 1, n

natural ¬ introduction 1

1	$\neg[\![X]\!]$	state:⇒
2	$[\![X]\!]$	state:¬ ⇒, 1
⋮	⋮	
n	$[\![Y]\!]$	state:⇒, 2–n
$n+1$	$\neg\neg[\![X]\!]$	¬ I, 1, 2–n

natural ¬ introduction 2

The added complication here is that we now have to keep track of whether introduced formulas are presupposed to be true or false. For instance, line 2 in the right ¬ introduction proof presupposes that the sentence $[\![X]\!]$ is false, and therefore the cover (⌣) relation can flip the truth of the sentence to be true. Similarly, in the left proof we assume that our premises are true – this is not necessarily the case if, e.g., you are trying to prove your premises false.[7] If this information is not available *a priori* in the proof, the new negation introduction rules are only warranted by the full negation relation (⅄).

We can combine these insights to solve a our motivating example of the disjunctive syllogism, shown in Figure 3.5.

[7]This is a problem even in vanilla natural logic proofs. If you believe the premise to be false, you have to start in the false state of the finite state automata in Figure 3.4.

1	⟦*Bill stole the cookies*⟧ ∨ ⟦*Ted stole the cookies*⟧	
2	⟦*Ted did not steal the cookies*⟧	
3	⟦*Bill stole the cookies*⟧	
4	⟦*Bill stole the cookies*⟧	R, 3
5	⟦*Ted stole the cookies*⟧	
6	⟦*Ted did steal the cookies*⟧	⟦*steal*⟧ ≡ ⟦*did steal*⟧; state:⇒, 5
7	⟦*Ted did not steal the cookies*⟧	⟦*did*⟧ ↓ ⟦*did not*⟧; state:¬ ⇒, 6
8	¬⟦*Ted stole the cookies*⟧	¬ I, 5, 7
9	⊥	⊥ I, 5, 8
10	⟦*Bill stole the cookies*⟧	⊥ E, 9
11	⟦*Bill stole the cookies*⟧	∨ E, 1, 3–5, 5–10

Figure 3.5: A hybrid propositional + natural logic proof showing the disjunctive syllogism.

3.3.3 Shallow Semantic Parsing

The hybrid proof system from Section 3.3.2 lays out an elegant theory for entailment when we are composing atomic sentences. However, in natural language, propositional statements are usually not said in a such a straightforward way. Even taking our simple running example, it's much more natural to say *Either Bill or Ted stole the cookies* than *Bill stole the cookies or Ted stole the cookies*. Therefore, it is clearly useful to have a semantic parser which takes as input a complex utterance, and produces as output a hybrid natural/propositional logic formula. Such a parser would allow us to do powerful forms of logical reasoning, without requiring a heavyweight logical representation (e.g., the Abstract Meaning Representation [7]). In this section, we outline some of the challenges for such a parser.

***And* is not always conjunction** The classic case of this is a sentence like *Jack and Jill are friends*, where of course this sentence should not be parsed as *Jack is friends* and *Jill is friends*. However, there are more subtle cases of this as well. For example, consider the exchange:

> *What would you like for dinner?*
> *Taco Bell and Pizza Hut are my choice.*

Here, the utterance *Taco Bell and Pizza Hut are my choice* has a meaning more akin to *Taco Bell is my choice* or *Pizza Hut is my choice*.

***Or* is not always disjunction** Conversely, *or* does not always imply a disjunction. For example, consider the exchange:

> *Can I have pets in the apartment?*
> *Cats or dogs are ok.*

Clearly here, the semantics of the second statement is: $[\![\textit{cats are ok}]\!] \wedge [\![\textit{dogs are ok}]\!]$.

***Or* is often exclusive** In propositional logic, disjunction is not exclusive: $A \vee B$ does not forbid $A \wedge B$. This is played out in many uses of the word *or*; for example: *you should do your taxes or your homework*. However, in a surprising number of cases, *or* implies an exclusive or. Take, for instance:

> *Do or do not; there is no try.*
> *Either cats or dogs chase mice.*
> *Jack or Jill is responsible for that.*

Handling these and other phenomena in language is a nontrivial problem. However, this is true for any semantic parsing task; and if a good parser can be built then propositional and natural logic provide a fast and expressive paradigm for solving entailment tasks.

The rest of this dissertation will use various aspects of natural logic in practical applications, focusing on large-scale textual inference and question answering tasks. The first

challenge, addressed by the next chapter, is on extending the proof theory described here to operate over not a single premise, but a very large set of candidate premises. That is, we are using natural logic to look for any supporting premise for a hypothesis in a very large collection of plain text.

Chapter 4

Common-Sense Reasoning

4.1 Introduction

We approach the task of common-sense reasoning by casting it as a *database completion* task: given a non-exhaustive database of true facts, we would like to predict whether an unseen fact is true and should belong in the database. This is intuitively cast as an inference problem from a collection of candidate premises to the truth of the query. For example, we would like to infer that *no carnivores eat animals* is false given a database containing *the cat ate a mouse* (see Figure 5.1).

These inferences are difficult to capture in a principled way while maintaining high recall, particularly for large scale open-domain tasks. Learned inference rules are difficult to generalize to arbitrary relations, and standard IR methods easily miss small but semantically important lexical differences. Furthermore, many methods require explicitly modeling either the database, the query, or both in a formal meaning representation (e.g., Freebase tuples).

Although projects like the Abstract Meaning Representation [7] have made headway in providing broad-coverage meaning representations, it remains appealing to use human language as the vessel for inference. Furthermore, OpenIE and similar projects have been very successful at collecting databases of natural language snippets from an ever-increasing corpus of unstructured text. These factors motivate our use of Natural Logic – a proof system built on the syntax of human language – for broad coverage database completion.

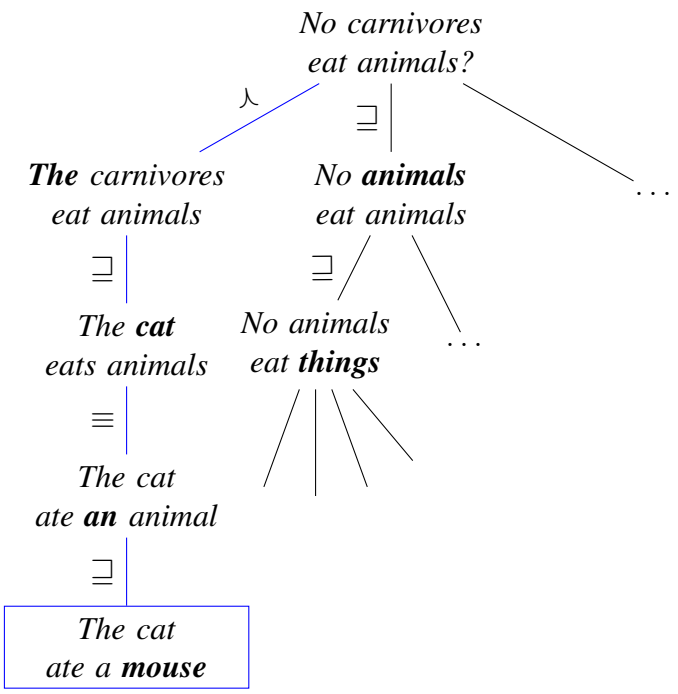


Figure 4.1: Natural Logic inference cast as search. The path to the boxed premise *the cat ate a mouse* disproves the query *no carnivores eat animals*, as it passes through the negation relation (⅄). This path is one of many candidates taken; the premise is one of many known facts in the database. The edge labels denote Natural Logic inference steps.

Prior work on Natural Logic has focused on inferences from a single relevant premise, making use of only formally valid inferences. We improve upon computational Natural Logic in three ways: (i) our approach operates over a very large set of candidate premises simultaneously; (ii) we do not require explicit alignment between a premise and the query; and (iii) we allow imprecise inferences at an associated cost learned from data.

Our approach casts inference as a single unified search problem from a query to any valid supporting premise. Each transition along the search denotes a (reverse) inference step in Natural Logic, and incurs a cost reflecting the system's confidence in the validity of that step. This approach offers two contributions over prior work in database completion: (i) it allows for unstructured text as the input database without any assumptions about the schema or domain of the text, and (ii) it proposes Natural Logic for inference, rather than translating to a formal logic syntax. Moreover, the entire pipeline is implemented in a

$\bowtie$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\smile$	$\#$
$\equiv$	$\equiv$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\Downarrow$	$\smile$	$\#$
$\sqsubseteq$	$\sqsubseteq$	$\sqsubseteq$	$\#$	$\Downarrow$	$\Downarrow$	$\#$	$\#$
$\sqsupseteq$	$\sqsupseteq$	$\#$	$\sqsupseteq$	$\smile$	$\#$	$\smile$	$\#$
$\curlywedge$	$\curlywedge$	$\smile$	$\Downarrow$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$	$\#$
$\Downarrow$	$\Downarrow$	$\#$	$\Downarrow$	$\sqsubseteq$	$\#$	$\sqsubseteq$	$\#$
$\smile$	$\smile$	$\smile$	$\#$	$\sqsupseteq$	$\sqsupseteq$	$\#$	$\#$
$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$	$\#$

Table 4.1: The join table as shown in Icard [52]. Entries in the table are the result of joining a row with a column. Note that the $\#$ always joins to yield $\#$, and $\equiv$ always joins to yield the input relation.

single elegant search framework, which scales easily to large databases.

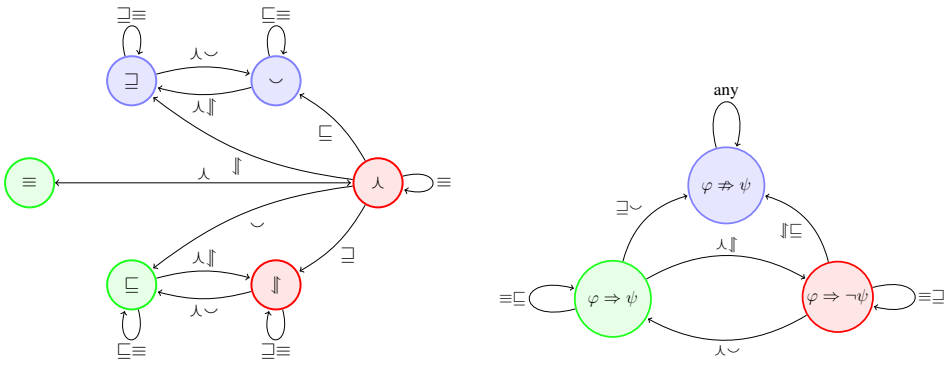
4.2 MacCartney's Proofs By Aligmnent

MacCartney and Manning [70] approach inference for natural logic in the context of inferring whether a single relevant premise entails a query. Their approach first generates an alignment between the premise and the query, and then classifies each aligned segment into one of the lexical relations described in Chapter 3. Inference reduces to projecting each of these relations according to the projection function ρ (Table 3.3) and iteratively *joining* two projected relations together to get the final entailment relation. This join relation, denoted as $\bowtie$, is given in Table 4.1.

To illustrate, we can consider MacCartney's example inference from *Stimpy is a cat* to *Stimpy is not a poodle*. An alignment of the two statements would provide three lexical mutations: $r_1 \coloneqq \textit{cat} \rightarrow \textit{dog}$, $r_2 \coloneqq \cdot \rightarrow \textit{not}$, and $r_3 \coloneqq \textit{dog} \rightarrow \textit{poodle}$. Each of these are then projected with the projection function ρ, and are joined using the join relation:

$$r_0 \bowtie \rho(r_1) \bowtie \rho(r_2) \bowtie \rho(r_3),$$

where the initial relation r_0 is axiomatically $\equiv$. In MacCartney's work this style of proof is presented as a table. The last column (s_i) is the relation between the premise and the i^{th} step in the proof, and is constructed inductively as $s_i \coloneqq s_{i-1} \bowtie \rho(r_i)$:



(a) (b)

Figure 4.2: (a) Natural logic inference expressed as a finite state automaton. Omitted edges go to the unknown state (#), with the exception of omitted edges from ≡, which go to the state of the edge type. Green states (≡, ⊑) denote valid inferences; red states (⫛, ⅄) denote invalid inferences; blue states (⊒, ⌣) denote inferences of unknown validity. (b) The join table collapsed into the three meaningful states over truth values.

	Mutation	r_i	$\rho(r_i)$	s_i
r_1	*cat*→*dog*	⫛	⫛	⫛
r_2	· →*not*	⅄	⅄	⊑
r_3	*dog*→*poodle*	⊒	⊏	⊑

In our example, we would conclude that *Stimpy is a cat* ⊑ *Stimpy is not a poodle* since s_3 is ⊑; therefore the inference is valid. More details on natural logic can be found in Chapter 3.

4.3 Inference as a Finite State Machine

We show that the tabular proof formulation from Section 4.2 can be viewed as a finite state machine, and present a novel observation that we can losslessly collapse this finite state machine into only three intuitive inference states. These observations allow us to formulate our search problem such that a search path corresponds to an input to (i.e., path through) this collapsed state machine.

Taking notation from Section 4.2, we construct a finite state machine over states $s \in \{\sqsubseteq, \sqsupseteq, \dots\}$. A machine in state s_i corresponds to relation s_i holding between the initial

premise and the derived fact so far. States therefore correspond to states of *logical validity*. The start state is $\equiv$. Outgoing transitions correspond to *inference steps*. Each transition is labeled with a projected relation $\rho(r) \in \{\sqsubseteq, \sqsupseteq, \dots\}$, and spans from a source state s to a target s' according to the join table. That is, the transition $s \xrightarrow{\rho(r)} s'$ exists iff $s' = s \bowtie \rho(r)$. For example, the path in Figure 5.1 yields the transitions $\equiv \xrightarrow{\perp} \perp \xrightarrow{\sqsupseteq} \Vdash \xrightarrow{\equiv} \Vdash \xrightarrow{\sqsupseteq} \Vdash$. Figure 4.2a shows the automaton, with trivial edges omitted for clarity.

Our second contribution is collapsing this automaton into the three meaningful states we use as output: *valid* ($\varphi \Rightarrow \psi$), *invalid* ($\varphi \Rightarrow \neg\psi$), and *unknown validity* ($\varphi \not\Rightarrow \psi$). We can cluster states in Figure 4.2a into these three categories. The relations $\equiv$ and $\sqsubseteq$ correspond to valid inferences; $\perp$ and $\Vdash$ correspond to invalid inferences; $\sqsupseteq$, $\smile$ and $\#$ correspond to unknown validity. This clustering mirrors that used by MacCartney for his textual entailment experiments.

Collapsing the FSA into the form in Figure 4.2b becomes straightforward from observing the regularities in Figure 4.2a. Nodes in the valid cluster transition to invalid nodes always and only on the relations $\perp$ and $\Vdash$. Symmetrically, invalid nodes transition to valid nodes always and only on $\perp$ and $\smile$. A similar pattern holds for the other transitions.

Formally, for every relation r and nodes a_1 and a_2 in the same cluster, if we have transitions $a_1 \xrightarrow{r} b_1$ and $a_2 \xrightarrow{r} b_2$ then b_1 and b_2 are necessarily in the same cluster. As a concrete example, we can take $r = \perp$ and the two states in the *invalid* cluster: $a_1 = \perp$, $a_2 = \Vdash$. Although $\perp \xrightarrow{\perp} \equiv$ and $\Vdash \xrightarrow{\perp} \sqsubseteq$, both $\equiv$ and $\sqsubseteq$ are in the same cluster (*valid*). It is not trivial *a priori* that the join table should have this regularity, and it certainly simplifies the logic for inference tasks.

A few observations deserve passing remark. First, even though the states $\sqsupseteq$ and $\smile$ appear meaningful, in fact there is no "escaping" these states to either a valid or invalid inference. Second, the hierarchy over relations presented in Icard [52] becomes apparent – in particular, $\perp$ always behaves as negation, whereas its two "weaker" versions ($\Vdash$ and $\smile$) only behave as negation in certain contexts. Lastly, with probabilistic inference, transitioning to the unknown state can be replaced with staying in the current state at a (potentially arbitrarily large) cost to the confidence of validity. This allows us to make use of only two states: *valid* and *invalid*.

4.4 Inference As Search

Natural Logic allows us to formalize our approach elegantly as a single search problem. Given a query, we search over the space of possible facts for a valid premise in our database. The nodes in our search problem correspond to candidate facts (Section 4.4.1); the edges are mutations of these facts (Section 4.4.2); the costs over these edges encode the confidence that this edge maintains an informative inference (Section 4.4.5). This mirrors the automaton defined in Section 4.3, except importantly we are constructing a reversed derivation, and are therefore "traversing" the FSA backwards.

This approach is efficient over a large database of 270 million entries without making use of explicit queries over the database; nor does the approach make use of any sort of approximate matching against the database, beyond lemmatizing individual lexical items. The motivation in prior work for approximate matches – to improve the recall of candidate premises – is captured elegantly by relaxing Natural Logic itself. We show that allowing invalid transitions with appropriate costs generalizes JC distance [57] – a common thesaurus-based similarity metric (Section 4.4.3). Importantly, however, the entire inference pipeline is done within the framework of weighted lexical transitions in Natural Logic.

4.4.1 Nodes

The space of possible nodes in our search is the set of possible partial derivations. To a first approximation, this is a pair (w, s) of a surface form w tagged with word sense and polarity, and an inference state $s \in \{\text{valid}, \text{invalid}\}$ in our collapsed FSA (Figure 4.2b). For example, the search path in Figure 5.1 traverses the nodes:

(*No carnivores eat animals*, valid)
(*The carnivores eat animals*, invalid)
(*The cat eats animals*, invalid)
(*The cat eats an animal*, invalid)
(*The cat ate a mouse*, invalid)

During search, we assume that the validity states s are reversible – if we know that *the cat ate a mouse* is true, we can infer that *no carnivores eat animals* is false. In addition,

our search keeps track of some additional information:

Mutation Index Edges between sentences are most naturally defined to correspond to mutations of individual lexical items. We therefore maintain an index of the next item to mutate at each search state. Importantly, this enforces that each derivation orders mutations left-to-right; this is computationally efficient, at the expense of rare search errors. A similar observation is noted in MacCartney [69], where prematurely collapsing to # occasionally misses inferences.

Polarity Mutating operators can change the polarity on a span in the fact. Since we do not have the full parse tree at our disposal at search time, we track a small amount of metadata to guess the scope of the mutated operator.

4.4.2 Transitions

We begin by introducing some terminology. A *transition template* is a broad class of transitions; for instance WordNet hypernymy. A *transition* (or *transition instance*) is a particular instantiation of a transition template. For example, the transition from *cat* to *feline*. Lastly, an *edge* in the search space connects two nodes, which are separated by a single transition instance. For example, an edge exists between *some felines have tails* and *some cats have tails*. Transition [instances] are stored statically in memory, whereas edges are constructed on demand.

Transition templates provide a means of defining transitions and subsequently edges in our search space using existing lexical resources (e.g., WordNet, distributional similarity, etc.). We can then define a mapping from these templates to Natural Logic lexical relations. This allows us to map every edge in our search graph back to the Natural Logic relation it instantiates. The full table of transition templates is given in Table 4.2, along with the Natural Logic relation that instances of the template introduce. We include most relations in WordNet as transitions, and parametrize insertions and deletions by the part of speech of the token being inserted/deleted.

Once we have an edge defining a lexical mutation with an associated Natural Logic relation r, we can construct the corresponding end node (w', s') such that w' is the sentence

Transition Template	Relation
WordNet hypernym	$\sqsubseteq$
WordNet hyponym	$\sqsupseteq$
WordNet antonym†	$\Downarrow$
WordNet synonym/pertainym†	$\equiv$
Distributional nearest neighbor	$\equiv$
Delete word†	$\sqsubseteq$
Add word†	$\sqsupseteq$
Operator weaken	$\sqsubseteq$
Operator strengthen	$\sqsupseteq$
Operator negate	$\curlywedge$
Operator synonym	$\equiv$
Change word sense	$\equiv$

Table 4.2: The edges allowed during inference. Entries with a dagger (†) are parametrized by their part-of-speech tag, from the restricted list of {noun,adjective,verb,other}. The first column describes the type of the transition. The set-theoretic relation introduced by each relation is given in the second column.

with the lexical mutation applied, and s' is the validity state obtained from the FSA in Section 4.3. For instance, if our edge begins at (w, s), and there exists a transition in the FSA from $s' \xrightarrow{r} s$, then we define the end point of the edge to be (w', s'). To illustrate concretely, suppose our search state is:

(*some felines have tails*, valid)

The transition template for WordNet hypernymy gives us a transition instance from *feline* to *cat*, corresponding to the Natural Logic inference *cat* $\overset{\sqsubseteq}{\Rightarrow}$ *feline*. Recall, we are constructing the inference in reverse, starting from the consequent (query). We then notice that the transition $valid \overset{\sqsubseteq}{\Rightarrow} valid$ in the FSA ends in our current inference state (*valid*), and set our new inference state to be the start state of the FSA transition – in this case, we maintain validity.

Note that negation is somewhat subtle, as the transitions are not symmetric from valid to invalid and visa versa, and we do not know our true inference state with respect to the premise yet. In practice, the search procedure treats all three of $\{\curlywedge, \Downarrow, \smile\}$ as negation, and re-scores complete derivations once their inference states are known.

It should be noted that the mapping from transition templates to relation types is intentionally imprecise. For instance, clearly nearest neighbors do not preserve equivalence ($\equiv$); more subtly, while *all cats like milk* $\Downarrow$ *all cats hate milk*, it is not the case that *some cats like milk* $\Downarrow$ *some cats hate milk*.[1] We mitigate this imprecision by introducing a cost for each transition, and learning the appropriate value for this cost (see Section 4.5). The cost of an edge from fact (w, v) with surface form w and validity v to a new fact (w', v'), using a transition instance t_i of template t and mutating a word with polarity p, is given by $f_{t_i} \cdot \theta_{t,v,p}$. We define this as:

f_{t_i}: A value associated with every transition instance t_i, intuitively corresponding to how "far" the endpoints of the transition are.

$\theta_{t,v,p}$: A learned cost for taking a transition of template t, if the source of the edge is in a inference state of v and the word being mutated has polarity p.

The notation for f_{t_i} is chosen to evoke an analogy to features. We set f_{t_i} to be 1 in most cases; the exceptions are the edges over the WordNet hypernym tree and the nearest neighbors edges. In the first case, taking the hypernymy relation from w to w' to be $\uparrow_{w \to w'}$, we set:

$$f_{\uparrow_{w \to w'}} = \log \frac{p(w')}{p(w)} = \log p(w') - \log p(w).$$

The value $f_{\downarrow_{w \to w'}}$ is set analogously. We define $p(w)$ to be the "probability" of a concept – that is, the normalized frequency of a word w or any of its hyponyms in the Google N-Grams corpus [20]. Intuitively, this ensures that relatively long paths through fine-grained sections of WordNet are not unduly penalized. For instance, the path from *cat* to *animal* traverses six intermediate nodes, naïvely yielding a prohibitive search depth of 6. However, many of these transitions have low weight: for instance $f_{\uparrow_{cat \to feline}}$ is only 0.37.

For nearest neighbors edges, we take Neural Network embeddings learned in Huang et al. [51] corresponding to each vocabulary entry. We then define $f_{NN_{w \to w'}}$ to be the arc

[1] The latter example is actually a consequence of the projection function used in this work being overly optimistic, and does not take into account additivity and multiplicativity.

cosine of the cosine similarity (i.e., the angle) between word vectors associated with lexical items w and w':

$$f_{NN_{w \to w'}} = \arccos\left(\frac{w \cdot w'}{\|w\|\|w'\|}\right).$$

For instance, $f_{NN_{cat \to dog}} = 0.43$. In practice, we explore the 100 nearest neighbors of each word.

We can express f_{t_i} as a feature vector by representing it as a vector with value f_{t_i} at the index corresponding to (t, v, p) – the transition template, the validity of the inference, and the polarity of the mutated word. Note that the size of this vector mirrors the number of cost parameters $\theta_{t,v,p}$, and is in general smaller than the number of transition instances.

A search path can then be parametrized by a sequence of feature vectors $f_1, f_2, \ldots, f_n$, which in turn can be collapsed into a single vector $\mathbf{f} = \sum_i f_i$. The cost of a path is defined as $\theta \cdot \mathbf{f}$, where θ is the vector of $\theta_{t,v,p}$ values. Both $\mathbf{f}$ and θ are constrained to be non-negative, or else the search problem is misspecified.

4.4.3 Generalizing Similarities

An elegant property of our definitions of f_{t_i} is its ability to generalize JC distance. Let us assume we have lexical items w_1 and w_2, with a least common subsumer lcs. The JC distance $\text{dist}_{\text{jc}}(w_1, w_2)$ is:

$$\text{dist}_{\text{jc}}(w_1, w_2) = \log \frac{p(\text{lcs})^2}{p(w_1) \cdot p(w_2)}. \tag{4.1}$$

For simplicity, we simplify $\theta_{\uparrow,v,p}$ and $\theta_{\downarrow,v,p}$ as simply $\theta_\uparrow$ and $\theta_\downarrow$. Without loss of generality, we also assume that a path in our search is only modifying a single lexical item w_1, eventually reaching a mutated form w_2.

We can factorize the cost of a path, $\theta \cdot \mathbf{f}$, along the path from w_1 to w_2 through its lowest common subsumer (lcs), $[w_1, w_1^{(1)}, \ldots, \text{lcs}, \ldots, w_2^{(1)}, w_2]$, as follows:

$$\begin{aligned}\theta \cdot \phi &= \theta_{\uparrow}\left(\left[\log p(w_1^{(1)}) - \log p(w_1)\right] + \dots\right) + \\ &\quad \theta_{\downarrow}\left(\left[\log p(\text{lcs}) - \log p(w_1^{(n)})\right] + \dots\right) \\ &= \theta_{\uparrow}\left(\log \frac{p(\text{lcs})}{p(w_1)}\right) + \theta_{\downarrow}\left(\log \frac{p(\text{lcs})}{p(w_2)}\right) \\ &= \log \frac{p(\text{lcs})^{\theta_{\uparrow}+\theta_{\downarrow}}}{p(w_1)^{\theta_{\uparrow}} \cdot p(w_2)^{\theta_{\downarrow}}}.\end{aligned}$$

Note that setting both $\theta_{\uparrow}$ and $\theta_{\downarrow}$ to 1 exactly yields Formula (4.1) for JC distance. This, in addition to the inclusion of nearest neighbors as transitions, allows the search to capture the intuition that similar objects have similar properties (e.g., as used in Angeli and Manning [2]).

4.4.4 Deletions in Inference

Although inserting lexical items in a derivation (deleting words from the reversed derivation) is trivial, the other direction is not. For brevity, we refer to a deletion in the derivation as an insertion, since from the perspective of search we are inserting lexical items.

Naïvely, at every node in our search we must consider every item in the vocabulary as a possible insertion. We can limit the number of items we consider by storing the database as a trie. Since the search mutates the fact left-to-right (as per Section 4.4.1), we can consider children of a trie node as candidate insertions. To illustrate, given a search state with fact $w_0\, w_1 \dots w_n$ and mutation index i, we would look up completions w_{i+1} for $w_0\, w_1 \dots w_i$ in our trie of known facts.

Although this approach works well when i is relatively large, there are too many candidate insertions for small i. We special case the most extreme example for this, where $i = 0$ – that is, when we are inserting into the beginning of the fact. In this case, rather than taking all possible lexical items that start any fact, we take all items which are followed by the first word of our current fact. To illustrate, given a search state with fact $w_0\, w_1 \dots w_n$, we would propose candidate insertions w_{-1} such that $w_{-1}\, w_0\, w'_1 \dots w'_k$ is a known fact for some $w'_1 \dots w'_k$. More concretely, if we know that *fluffy cats have tails*, and are at a node

corresponding to *cats like boxes*, we propose *fluffy* as a possible insertion: *fluffy cats like boxes*.

4.4.5 Confidence Estimation

The last component in inference is translating a search path into a probability of truth. We notice from Section 4.4.2 that the *cost* of a path can be represented as $\theta \cdot \mathbf{f}$. We can normalize this value by negating every element of the cost vector θ and passing it through a sigmoid:

$$\text{confidence} = \frac{1}{1 + e^{-(-\theta \cdot \mathbf{f})}}.$$

Importantly, note that the cost vector must be non-negative for the search to be well-defined, and therefore the confidence value will be constrained to be between 0 and $\frac{1}{2}$.

At this point, we have a confidence that the given path has not violated strict Natural Logic. However, to translate this value into a probability we need to incorporate whether the inference path is confidently valid, or confidently invalid. To illustrate, a fact with a low confidence should translate to a probability of $\frac{1}{2}$, rather than a probability of 0. We therefore define the probability of validity as follows: We take v to be 1 if the query is in the *valid* state with respect to the premise, and -1 if the query is in the *invalid* state. For completeness, if no path is given we can set $v = 0$. The probability of validity becomes:

$$p(\text{valid}) = \frac{v}{2} + \frac{1}{1 + e^{v\theta \cdot \mathbf{f}}}. \tag{4.2}$$

Note that in the case where $v = -1$, the above expression reduces to $\frac{1}{2} -$ confidence; in the case where $v = 0$ it reduces to simply $\frac{1}{2}$. Furthermore, note that the probability of truth makes use of the same parameters as the cost in the search.

4.5 Learning Transition Costs

We describe our procedure for learning the transition costs θ. Our training data $\mathcal{D}$ consists of query facts q and their associated gold truth values y. Equation (4.2) gives us a probability that a particular inference is *valid*; we axiomatically consider a valid inference from a known premise to be justification for the truth of the query. This is at the expense of the (often incorrect) assumption that our database is clean and only contains true facts.

We optimize the likelihood of our gold annotations according to this probability, subject to the constraint that all elements in our cost vector θ be non-negative. We run the search algorithm described in Section 4.4 on every query $q_i \in \mathcal{D}$. This produces the highest confidence path x_1, along with its inference state v_i. We now have annotated tuples: $((x_i, v_i), y_i)$ for every element in our training set. Analogous to logistic regression, the log likelihood of our training data $\mathcal{D}$, subject to costs θ, is:

$$l_\theta(\mathcal{D}) = \sum_{0 \le i < |\mathcal{D}|} \Big[y_i \log \left(\frac{v_i}{2} + \frac{1}{1 + e^{v_i \theta \cdot \mathbf{f}(x_i)}} \right) + (1 - y_i) \log \left(\frac{-v_i}{2} + \frac{1}{1 + e^{-v_i \theta \cdot \mathbf{f}(x_i)}} \right) \Big],$$

where y_i is 1 if the example is annotated true and 0 otherwise, and $\mathbf{f}(x_i)$ are the features extracted for path x_i. The objective function is the negative log likelihood with an L_2 regularization term and a log barrier function to prohibit negative costs:

$$O(\mathcal{D}) = -l_\theta(\mathcal{D}) + \frac{1}{2\sigma^2} \|\theta\|_2^2 - \epsilon \log(\theta).$$

We optimize this objective using conjugate gradient descent. Although the objective is non-convex, in practice we can find a good initialization of weights to reduce the risk of arriving at local optima.

An elegant property of this formulation is that the weights we are optimizing correspond

directly to the costs used during search. This creates a positive feedback loop – as better weights are learned, the search algorithm is more likely to find confident paths, and more data is available to train from. We therefore run this learning step for multiple epochs, re-running search after each epoch. The weights for the first epoch are initialized to an approximation of valid Natural Logic weights. Subsequent epochs initialize their weights to the output of the previous epoch.

4.6 Experiments

We evaluate our system on two tasks: the FraCaS test suite, used by MacCartney and Manning [70, 71] evaluates the system's ability to capture Natural Logic inferences even without the explicit alignments of these previous systems. In addition, we evaluate the system's ability to predict common-sense facts from a large corpus of OpenIE extractions.

4.6.1 FraCaS Entailment Corpus

The FraCaS corpus [32] is a small corpus of entailment problems, aimed at providing a comprehensive test of a system's handling of various entailment patterns. We process the corpus following MacCartney and Manning [70]. It should be noted that many of the sections of the corpus are not directly applicable to Natural Logic inferences; MacCartney and Manning [70] identify three sections which are in the scope of their system, and consequently our system as well.

Results on the dataset are given in Table 4.3. Since the corpus is not a blind test set, the results are presented less as a comparison of performance, but rather to validate the expressive power of our search-based approach against MacCartney's align-and-classify approach. For the experiments, costs were set to express valid Natural Logic inference as a hard constraint.

The results show that the system is able to capture Natural Logic inferences with similar accuracy to the state-of-the-art system of MacCartney and Manning [71]. Note that our system is comparatively crippled in this framework along at least two dimensions: It cannot appeal to the premise when constructing the search, leading to the introduction of a class

§	Category	Count	Precision		Recall		Accuracy		
			N	M08	N	M08	N	M07	M08
1	**Quantifiers**	**44**	**91**	**95**	**100**	**100**	**95**	**84**	**97**
2	Plurals	24	80	90	29	64	38	42	75
3	Anaphora	6	100	100	20	60	33	50	50
4	Ellipses	25	100	100	5	5	28	28	24
5	**Adjectives**	**15**	**80**	**71**	**66**	**83**	**73**	**60**	**80**
6	**Comparatives**	**16**	**90**	**88**	**100**	**89**	**87**	**69**	**81**
7	Temporal	36	75	86	53	71	52	61	58
8	Verbs	8	–	80	0	66	25	63	62
9	Attitudes	9	–	100	0	83	22	55	89
Applicable (1,5,6)		**75**	**89**	**89**	**94**	**94**	**89**	**76**	**90**

Table 4.3: Results on the FraCaS textual entailment suite. N is this work; M07 refers to MacCartney and Manning [70]; M08 refers to MacCartney and Manning [71]. The relevant sections of the corpus intended to be handled by this system are sections 1, 5, and 6 (although not 2 and 9, which are also included in M08).

of search errors which are entirely absent from prior work. Second, the derivation process itself does not have access to the full parse tree of the candidate fact.

Although precision is fairly high even on the non-applicable sections of FraCaS, recall is significantly lower than prior work. This is a direct consequence of not having alignments to appeal to. For instance, we can consider two inferences:

Jack saw Jill is playing $\stackrel{?}{\Rightarrow}$ *Jill is playing*

Jill saw Jack is playing $\stackrel{?}{\Rightarrow}$ *Jill is playing*

It is clear from the parse of the sentence that the first is valid and the second is not; however, from the perspective of the search algorithm both make the same two edits: inserting *Jack* and *saw*. In order to err on the side of safety, we disallow deleting the verb *saw*.

4.6.2 Common Sense Reasoning

We validate our system's ability to infer unseen common sense facts from a large database of such facts. Whereas evaluation on FraCaS shows that our search formulation captures

System	P	R	F_1	Accuracy
Lookup	100.0	12.1	21.6	56.0
NaturalLI Only	88.8	40.1	55.2	67.5
NaturalLI + Lookup	90.6	49.1	63.7	72.0

Table 4.4: Accuracy inferring common-sense facts on a balanced test set. *Lookup* queries the lemmatized lower-case fact directly in the 270M fact database. *NaturalLI Only* disallows such lookups, and infers every query from only distinct premises in the database. *NaturalLI + Lookup* takes the union of the two systems.

applicable inferences as well as prior work, this evaluation presents a novel use-case for Natural Logic inference.

For our database of facts, we run the Ollie OpenIE system [74] over Wikipedia,[2] Simple Wikipedia,[3] and a random 5% of CommonCrawl. Extractions with confidence below 0.25 or which contained pronouns were discarded. This yielded a total of 305 million unique extractions composed entirely of lexical items which mapped into our vocabulary ($186\,707$ items). Each of these extracted triples (e_1, r, e_2) was then flattened into a plain-text fact $e_1\ r\ e_2$ and lemmatized. This yields 270 million unique lemmatized premises in our database. In general, each fact in the database could be arbitrary unstructured text; our use of Ollie extractions is motivated only by a desire to extract short, concise facts.

For our evaluation, we infer the top 689 most confident facts from the ConceptNet project [103]. To avoid redundancy with WordNet, we take facts from eight ConceptNet relations: MemberOf, HasA, UsedFor, CapableOf, Causes, HasProperty, Desires, and CreatedBy. We then treat the *surface text* field of these facts as our candidate query. This yields queries like the following:

not all birds can fly

noses are used to smell

nobody wants to die

music is used for pleasure

[2] `http://wikipedia.org/` (2013-07-03)

[3] `http://simple.wikipedia.org/` (2014-03-25)

For negative examples, we take the 689 ReVerb extractions [40] judged as false by Mechanical Turk workers [2]. This provides a set of plausible but nonetheless incorrect examples, and ensures that our recall is not due to over-zealous search. Search costs are tuned from an additional set of 540 true ConceptNet and 540 false ReVerb extractions.

Results are shown in Table 4.4. We compare against the baseline of looking up each fact verbatim in the fact database. Note that both the query and the facts in the database are short snippets, already lemmatized and lower-cased; therefore, it is not in principle unreasonable to expect a database of 270 million extractions to contain these facts. Nonetheless, only 12% of facts were found via a direct lookup. We show that NaturalLI (allowing lookups) improves this recall four-fold, at only an 9.4% drop in precision.

Chapter 5

Open Domain Information Extraction

5.1 Introduction

Open information extraction (open IE) has been shown to be useful in a number of NLP tasks, such as question answering [41], relation extraction [95], and information retrieval [38]. Conventionally, open IE systems search a collection of patterns over either the surface form or dependency tree of a sentence. Although a small set of patterns covers most simple sentences (e.g., subject verb object constructions), relevant relations are often spread across clauses (see Figure 5.1) or presented in a non-canonical form.

Systems like Ollie [74] approach this problem by using a bootstrapping method to create a large corpus of broad-coverage partially lexicalized patterns. Although this is effective at capturing many of these patterns, it can lead to unintuitive behavior on out-of-domain text. For instance, while *Obama is president* is extracted correctly by Ollie as *(Obama; is; president)*, replacing *is* with *are* in *cats are felines* produces no extractions. Furthermore, existing systems struggle at producing canonical argument forms – for example, in Figure 5.1 the argument *Heinz Fischer of Austria* is likely less useful for downstream applications than *Heinz Fischer*.

In this work, we shift the burden of extracting informative and broad coverage triples away from this large pattern set. Rather, we first pre-process the sentence in linguistically motivated ways to produce coherent clauses which are (1) logically entailed by the original sentence, and (2) easy to segment into open IE triples. Our approach consists of two stages:

Born in Honolulu, Hawaii, Obama is a US Citizen.

Our System	**Ollie**
(Obama; is; US citizen)	(Obama; is; a US citizen)
(Obama; born in; Honolulu, Hawaii)	(Obama; be born in; Honolulu)
	(Honolulu; be born in; Hawaii)
	(Obama; is citizen of; US)

Friends give true praise.
Enemies give fake praise.

Our System	**Ollie**
(friends; give; true praise)	(friends; give; true praise)
(friends; give; praise)	
(enemies; give; fake praise)	(enemies; give; fake praise)

Heinz Fischer of Austria visits the US

Our System	**Ollie**
(Heinz Fischer; visits; US)	(Heinz Fischer of Austria; visits; the US)

Figure 5.1: Open IE extractions produced by the system, alongside extractions from the state-of-the-art Ollie system. Generating coherent clauses before applying patterns helps reduce false matches such as *(Honolulu; be born in; Hawaii)*. Inference over the substructure of arguments, in turn, allows us to drop unnecessary information (e.g., *of Austria*), but only when it is warranted (e.g., keep *fake* in *fake praise*).

we first learn a classifier for splitting a sentence into shorter utterances (Section 5.2), and then appeal to natural logic [90] to maximally shorten these utterances while maintaining necessary context (Section 5.3.1). A small set of 14 hand-crafted patterns can then be used to segment an utterance into an open IE triple.

We treat the first stage as a greedy search problem: we traverse a dependency parse tree recursively, at each step predicting whether an edge should yield an independent clause. Importantly, in many cases naïvely yielding a clause on a dependency edge produces an incomplete utterance (e.g., *Born in Honolulu, Hawaii*, from Figure 5.1). These are often attributable to control relationships, where either the subject or object of the governing clause controls the subject of the subordinate clause. We therefore allow the produced

clause to sometimes inherit the subject or object of its governor. This allows us to capture a large variety of long range dependencies with a concise classifier.

From these independent clauses, we then extract shorter sentences, which will produce shorter arguments more likely to be useful for downstream applications. A natural framework for solving this problem is natural logic – a proof system built on the syntax of human language (see Section 5.3.1). We can then observe that *Heinz Fischer of Austria visits China* entails that *Heinz Fischer visits China*. On the other hand, we respect situations where it is incorrect to shorten an argument. For example, *No house cats have rabies* should not entail that *cats have rabies*, or even that *house cats have rabies*.

When careful attention to logical validity is necessary – such as textual entailment – this approach captures even more subtle phenomena. For example, whereas *all rabbits eat fresh vegetables* yields *(rabbits; eat; vegetables)*, the apparently similar sentence *all young rabbits drink milk* does not yield *(rabbits; drink; milk)*.

We show that our new system performs well on a real world evaluation – the TAC KBP Slot Filling challenge [100]. We outperform both an official submission on open IE, and a baseline of replacing our extractor with Ollie, a state-of-the-art open IE systems.

5.2 Inter-Clause Open IE

In the first stage of our method, we produce a set of self-contained clauses from a longer utterance. Our objective is to produce a set of clauses which can stand on their own syntactically and semantically, and are entailed by the original sentence (see Figure 5.2). Note that this task is not specific to extracting open IE triples. Conventional relation extractors, entailment systems, and other NLP applications may also benefit from such a system.

We frame this task as a search problem. At a given node in the parse tree, we classify each outgoing arc $e = p \xrightarrow{l} c$, from the governor p to a dependent c with [collapsed] Stanford Dependency label l, into an action to perform on that arc. Once we have chosen an action to take on that arc, we can recurse on the dependent node. We decompose the action into two parts: (1) the action to take on the outgoing edge e, and (2) the action to take on the governor p. For example, in our motivating example, we are considering the arc: $e = \textit{took} \xrightarrow{vmod} \textit{born}$. In this case, the correct action is to (1) yield a new clause rooted

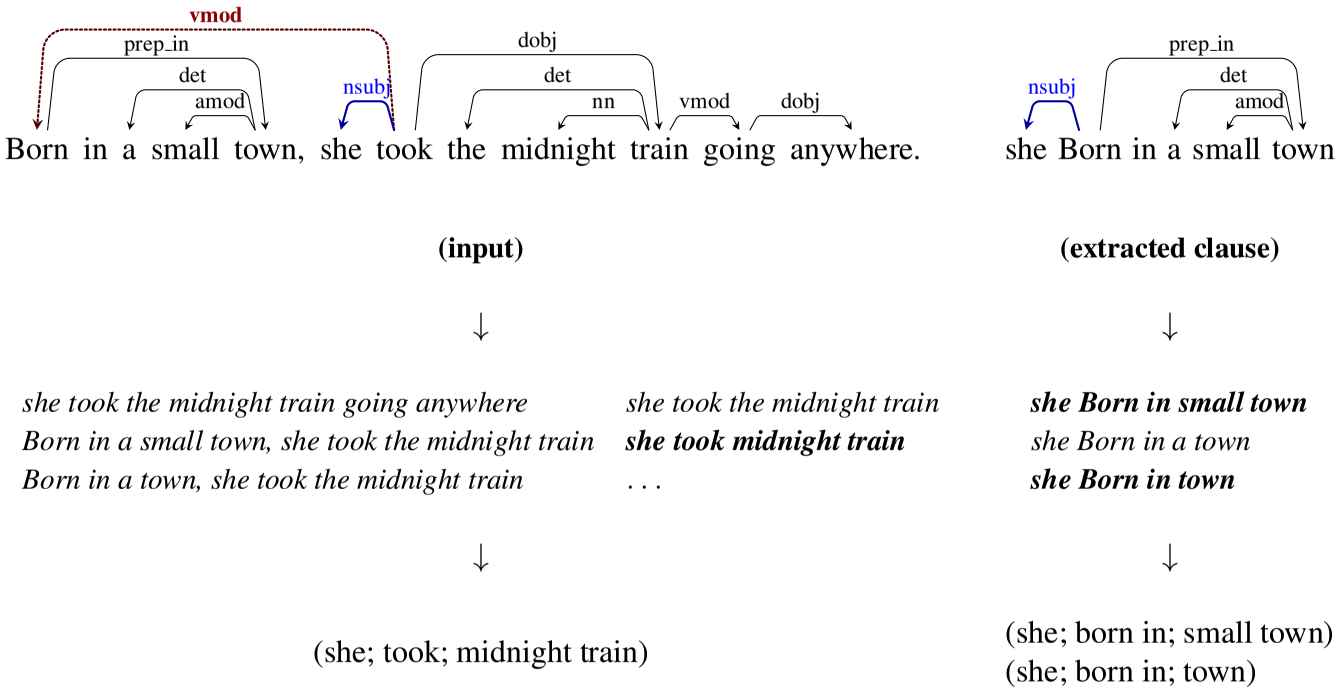


Figure 5.2: An illustration of our approach. From left to right, a sentence yields a number of independent clauses (e.g., *she Born in a small town* – see Section 5.2). From top to bottom, each clause produces a set of entailed shorter utterances, and segments the ones which match an atomic pattern into a relation triple (see Section 5.3.1).

at *born*, and (2) interpret the subject of *born* as the subject of *took*.

We proceed to describe this action space in more detail, followed by an explanation of our training data, and finally our classifier.

5.2.1 Action Space

The three actions we can perform on a dependency edge are:

Yield Yields a new clause on this dependency arc. A canonical case of this action is the arc *suggest* $\xrightarrow{ccomp}$ *brush* in *Dentists suggest that you should brush your teeth*, yielding *you should brush your teeth*.

Recurse Recurse on this dependency arc, but do not yield it as a new clause. For example, in the sentence *faeries are dancing in the field where I lost my bike*, we must recurse through the intermediate constituent *the field where I lost my bike* – which itself is not relevant – to get to the clause of interest: *I lost my bike*.

Stop Do not recurse on this arc, as the subtree under this arc is not entailed by the parent sentence. This is the case, for example, for most leaf nodes (*furry cats are cute* should not entail the clause *furry*), and is an important action for the efficiency of the algorithm.

With these three actions, a search path through the tree becomes a sequence of **Recurse** and **Yield** actions, terminated by a **Stop** action (or leaf node). For example, a search sequence $A \xrightarrow{Recurse} B \xrightarrow{Yield} C \xrightarrow{Stop} D$ would yield a clause rooted at C. A sequence $A \xrightarrow{Yield} B \xrightarrow{Yield} C \xrightarrow{Stop} D$ would yield clauses rooted at both B and C. Finding all such sequences is in general exponential in the size of the tree. In practice, during training we run breadth first search to collect the first $10\,000$ sequences. During inference we run uniform cost search until our classifier predictions fall below a given threshold.

For the **Stop** action, we do not need to further specify an action to take on the parent node. However, for both of the other actions, it is often the case that we would like to capture a controller in the higher clause. We define three such common actions:

Subject Controller If the arc we are considering is not already a subject arc, we can copy the subject of the parent node and attach it as a subject of the child node. This is the action taken in the example *Born in a small town, she took the midnight train*.

Object Controller Analogous to the subject controller action above, but taking the object instead. This is the intended action for examples like *I persuaded Fred to leave the room*.[1]

Parent Subject If the arc we are taking is the only outgoing arc from a node, we take the parent node as the (passive) subject of the child. This is the action taken in the example *Obama, our 44th president* to yield a clause with the semantics of *Obama [is] our 44th president*.

Although additional actions are easy to imagine, we found empirically that these cover a wide range of applicable cases. We turn our attention to the training data for learning these actions.

5.2.2 Training

We collect a noisy dataset to train our clause generation model. We leverage the *distant supervision* assumption for relation extraction, which creates a noisy corpus of sentences annotated with relation mentions (subject and object spans in the sentence with a known relation). Then, we take this annotation as itself distant supervision for a correct sequence of actions to take: any sequence which recovers the known relation is correct.

We use a small subset of the KBP source documents for 2010 [56] and 2013 [100] as our distantly supervised corpus. To try to maximize the density of known relations in the training sentences, we take all sentences which have at least one known relation for every 10 tokens in the sentence, resulting in $43\,155$ sentences. In addition, we incorporate the $23\,725$ manually annotated examples from Angeli et al. [4].

Once we are given a collection of labeled sentences, we assume that a sequence of actions which leads to a correct extraction of a known relation is a *positive sequence*. A correct extraction is any extraction we produce from our model (see Section 5.3) which has

[1]The system currently misses most most such cases due to insufficient support in the training data.

the same arguments as the known relation. For instance, if we know that Obama was born in Hawaii from the sentence *Born in Hawaii, Obama . . .* , and an action sequence produces the triple *(*Obama, born in, Hawaii), then we take that action sequence as a positive sequence.

Any sequence of actions which results in a clause which produces no relations is in turn considered a *negative sequence*. The third case to consider is a sequence of actions which produces a relation, but it is not one of the annotated relations. This arises from the *incomplete negatives* problem in distantly supervised relation extraction [77]: since our knowledge base is not exhaustive, we cannot be sure if an extracted relation is incorrect or correct but previously unknown. Although many of these unmatched relations are indeed incorrect, the dataset is sufficiently biased towards the STOP action that the occasional false negative hurts end-to-end performance. Therefore, we simply discard such sequences.

Given a set of noisy positive and negative *sequences*, we construct training data for our action classifier. All but the last action in a positive sequence are added to the training set with the label **Recurse**; the last action is added with the label **Split**. Only the last action in a negative sequence is added with the label **Stop**. We partition the feature space of our dataset according to the action applied to the parent node.

5.2.3 Inference

We train a multinomial logistic regression classifier on our noisy training data, using the features in Table 5.1. The most salient features are the label of the edge being taken, the incoming edge to the parent of the edge being taken, neighboring edges for both the parent and child of the edge, and the part of speech tag of the endpoints of the edge. The dataset is weighted to give $3\times$ weight to examples in the **Recurse** class, as precision errors in this class are relatively harmless for accuracy, while recall errors are directly harmful to recall.

Inference now reduces to a search problem. Beginning at the root of the tree, we consider every outgoing edge. For every possible action to be performed on the parent (i.e., clone subject, clone root, no action), we apply our trained classifier to determine whether we (1) split the edge off as a clause, and recurse; (2) do not split the edge, and recurse; or (3) do not recurse. In the first two cases, we recurse on the child of the arc, and continue until either all arcs have been exhausted, or all remaining candidate arcs have been marked

Feature Class	Feature Templates
Edge taken	$\{l, \text{short\_name}(l)\}$
Last edge taken	$\{\text{incoming\_edge}(p)\}$
Neighbors of parent	$\{\text{nbr}(p), (p, \text{nbr}(p))\}$
Grandchild edges	$\{\text{out\_edge}(c),$ $(e, \text{out\_edge}(c))\}$
Grandchild count	$\{\text{count}\,(\text{nbr}(e_{\text{child}}))$ $(e, \text{count}\,(\text{nbr}(e_{\text{child}})))\}$
Has subject/object	$\forall_{e \in \{e, e_{\text{child}}\}} \forall_{l \in \{subj, obj\}}$ $\mathbb{1}(l \in \text{nbr}(e))$
POS tag signature	$\{\text{pos}(p), \text{pos}(c),$ $(\text{pos}(p), \text{pos}(c))\}$
Features at root	$\{\mathbb{1}(p = root), \text{POS}(p)\}$

Table 5.1: Features for the clause splitter model, deciding to split on the arc $e = p \xrightarrow{l} c$. The feature class is a high level description of features; the feature templates are the particular templates used. For instance, the POS signature contains the tag of the parent, the tag of the child, and both tags joined in a single feature. Note that all features are joined with the action to be taken on the parent.

as not recursable.

We will use the scores from this classifier to inform the score assigned to our generated open IE extractions (Section 5.3). The score of a clause is the product of the scores of actions taken to reach the clause. The score of an extraction will be this score multiplied by the score of the extraction given the clause.

5.3 Intra-Clause Open IE

We now turn to the task of generating a maximally compact sentence which retains the core semantics of the original utterance, and parsing the sentence into a conventional open IE subject verb object triple. This is often a key component in downstream applications, where extractions need to be not only *correct*, but also *informative*. Whereas an argument like *Heinz Fischer of Austria* is often correct, a downstream application must apply further processing to recover information about either *Heinz Fischer*, or *Austria*. Moreover, it must do so without the ability to appeal to the larger context of the sentence.

5.3.1 Validating Deletions with Natural Logic

We adopt a subset of natural logic semantics dictating contexts in which lexical items can be removed. Natural logic as a formalism captures common logical inferences appealing directly to the form of language, rather than parsing to a specialized logical syntax. It provides a proof theory for lexical mutations to a sentence which either preserve or negate the truth of the premise.

For instance, if *all rabbits eat vegetables* then *all cute rabbits eat vegetables*, since we are allowed to mutate the lexical item *rabbit* to *cute rabbit*. This is done by observing that *rabbit* is in scope of the first argument to the operator *all*. Since *all* induces a *downward polarity* environment for its first argument, we are allowed to replace *rabbit* with an item which is more specific – in this case *cute rabbit*. To contrast, the operator *some* induces an *upward polarity* environment for its first argument, and therefore we may derive the inference from *cute rabbit* to *rabbit* in: *some cute rabbits are small* therefore *some rabbits are small*. For a more comprehensive introduction to natural logic, see van Benthem [105].

We mark the scopes of all operators (*all*, *no*, *many*, etc.) in a sentence, and from this determine whether every lexical item can be replaced by something more general (has upward polarity), more specific (downward polarity), or neither. In the absence of operators, all items have upwards polarity.

Each dependency arc is then classified into whether deleting the dependent of that arc makes the governing constituent at that node more general, more specific (a rare case), or neither.[2] For example, removing the *amod* edge in *cute* $\xleftarrow{amod}$ *rabbit* yields the more general lexical item *rabbit*. However, removing the *nsubj* edge in *Fido* $\xleftarrow{nsubj}$ *runs* would yield the unentailed (and nonsensical) phrase *runs*. The last, rare, case is an edge that causes the resulting item to be more specific – e.g., *quantmod*: *about* $\xleftarrow{quantmod}$ *200* is more general than *200*.

For most dependencies, this semantics can be hard-coded with high accuracy. However, there are at least two cases where more attention is warranted. The first of these concerns non-subsective adjectives: for example a *fake gun* is not a gun. For this case, we make use of the list of non-subsective adjectives collected in Nayak et al. [80], and prohibit their

[2] We use the Stanford Dependencies representation [35].

deletion as a hard constraint.

The second concern is with prepositional attachment, and direct object edges. For example, whereas *Alice went to the playground* $\xrightarrow{prep\_with}$ *Bob* entails that *Alice went to the playground*, it is not meaningful to infer that *Alice is friends* $\xrightarrow{prep\_with}$ *Bob* entails *Alice is friends*. Analogously, *Alice played* $\xrightarrow{dobj}$ *baseball on Sunday* entails that *Alice played on Sunday*; but, *Obama signed* $\xrightarrow{dobj}$ *the bill on Sunday* should not entail the awkward phrase **Obama signed on Sunday*.

We learn these attachment affinities empirically from the syntactic n-grams corpus of Goldberg and Orwant [46]. This gives us counts for how often object and preposition edges occur in the context of the governing verb and relevant neighboring edges. We hypothesize that edges which are frequently seen to co-occur are likely to be essential to the meaning of the sentence. To this end, we compute the probability of seeing an arc of a given type, conditioned on the most specific context we have statistics for. These contexts, and the order we back off to more general contexts, is given in Figure 5.3.

To compute a score s of *deleting* the edge from the affinity probability p collected from the syntactic n-grams, we simply cap the affinity and subtract it from 1:

$$s = 1 - \min(1, \frac{p}{K})$$

where K is a hyperparameter denoting the minimum fraction of the time an edge should occur in a context to be considered entirely unremovable. In our experiments, we set $K = \frac{1}{3}$.

The score of an extraction, then, is the product of the scores of each deletion multiplied by the score from the clause splitting step in Section 5.2.

5.3.2 Atomic Patterns

Once a set of short entailed sentences is produced, it becomes straightforward to segment them into conventional open IE triples. We employ 6 simple dependency patterns, given in Table 5.2, which cover the majority of atomic relations we are interested in.

When information is available to disambiguate the substructure of compound nouns (e.g., named entity segmentation), we extract additional relations with 5 dependency and

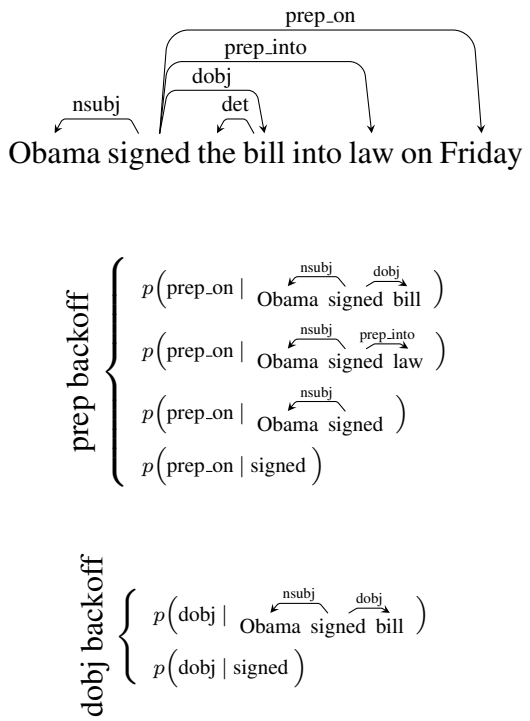


Figure 5.3: The ordered list of backoff probabilities when deciding to drop a prepositional phrase or direct object. The most specific context is chosen for which an empirical probability exists; if no context is found then we allow dropping prepositional phrases and disallow dropping direct objects. Note that this backoff arbitrarily orders contexts of the same size.

3 TokensRegex [23] surface form patterns. These are given in Table 5.3; we refer to these as *nominal relations*. Note that the constraint of named entity information is by no means required for the system. In other applications – for example, applications in vision – the otherwise trivial nominal relations could be quite useful.

5.4 Mapping OpenIE to a Known Relation Schema

A common use case for open IE systems is to map them to a known relation schema. This can either be done manually with minimal annotation effort, or automatically from available training data. We use both methods in our TAC-KBP evaluation. A collection

Input	Extraction
cats play with yarn	(cats; play with; yarn)
fish like to swim	(fish; like to; swim)
cats have tails	(cats; have; tails)
cats are cute	(cats; are; cute)
Tom and Jerry are fighting	(Tom; fighting; Jerry)
There are cats with tails	(cats; have; tails)

Table 5.2: The six dependency patterns used to segment an atomic sentence into an open IE triple.

Input	Extraction
Durin, son of Thorin	(Durin; is son of; Thorin)
Thorin's son, Durin	(Thorin; 's son; Durin)
IBM CEO Rometty	(Rometty; is CEO of; IBM)
President Obama	(Obama; is; President)
Fischer of Austria	(Fischer; is of; Austria)
IBM's research group	(IBM; 's; research group)
US president Obama	(Obama; president of; US)
Our president, Obama,	(Our president; be; Obama)

Table 5.3: The eight patterns used to segment a noun phrase into an open IE triple. The first five are dependency patterns; the last three are surface patterns.

of relation mappings was constructed by a single annotator in approximately a day,[3] and a relation mapping was learned using the procedure described in this section.

We map open IE relations to the KBP schema by searching for co-occurring relations in a large distantly-labeled corpus, and marking open IE and KBP relation pairs which have a high PMI^2 value [8, 39] conditioned on their type signatures matching. To compute PMI^2, we collect probabilities for the open IE and KBP relation co-occurring, the probability of the open IE relation occurring, and the probability of the KBP relation occurring. Each of these probabilities is conditioned on the type signature of the relation. For example, the joint probability of KBP relation r_k and open IE relation r_o, given a type signature of t_1, t_2, would be

[3]The official submission we compare against claimed two weeks for constructing their manual mapping, although a version of their system constructed in only 3 hours performs nearly as well.

KBP Relation	Open IE Relation	PMI^2	KBP Relation	Open IE Relation	PMI^2
Org:Founded	*found in*	1.17	Per:Date_Of_Birth	*be bear on*	1.83
	be found in	1.15		*bear on*	1.28
Org:Dissolved	**buy Chrysler in*	0.95	Per:Date_Of_Death	*die on*	0.70
	**membership in*	0.60		*be assassinate on*	0.65
Org:LOC_Of_HQ	*in*	2.12	Per:LOC_Of_Birth	*be bear in*	1.21
	base in	1.82	Per:LOC_Of_Death	**elect president of*	2.89
Org:Member_Of	**tough away game in*	1.80	Per:Religion	*speak about*	0.67
	**away game in*	1.80		*popular for*	0.60
Org:Parents	*'s bank*	1.65	Per:Parents	*daughter of*	0.54
	**also add to*	1.52		*son of*	1.52
Org:Founded_By	*invest fund of*	1.48	Per:LOC_Residence	*of*	1.48
	own stake besides	1.18		**independent from*	1.18

Table 5.4: A selection of the mapping from KBP to lemmatized open IE relations, conditioned on the types of the arguments being correct. The top one or two relations are shown for 7 person and 6 organization relations. Incorrect or dubious mappings are marked with an asterisk.

$$p(r_k, r_o \mid t_1, t_2) = \frac{\text{count}(r_k, r_o, t_1, t_2)}{\sum_{r'_k, r'_o} \text{count}(r'_k, r'_o, t_1, t_2)}.$$

Omitting the conditioning on the type signature for notational convenience, and defining $p(r_k)$ and $p(r_o)$ analogously, we can then compute The PMI^2 value between the two relations:

$$\text{PMI}^2(r_k, r_o) = \log\left(\frac{p(r_k, r_o)^2}{p(r_k) \cdot p(r_o)}\right)$$

Note that in addition to being a measure related to PMI, this captures a notion similar to *alignment by agreement* [67]; the formula can be equivalently written as $\log\left[p(r_k \mid r_o)p(r_o \mid r_k)\right]$. It is also functionally the same as the JC WordNet distance measure [57].

Some sample type checked relation mappings are given in Table 5.4. In addition to intuitive mappings (e.g., *found in* $\rightarrow$ Org:Founded), we can note some rare, but high precision pairs (e.g., *invest fund of* $\rightarrow$ Org:Founded_By). We can also see the noise in distant supervision occasionally permeate the mapping, e.g., with *elect president of* $\rightarrow$

Per:LOC_Of_Death – a president is likely to die in his own country.

5.5 Evaluation

We evaluate our approach in the context of a real-world end-to-end relation extraction task – the TAC KBP Slot Filling challenge. In Slot Filling, we are given a large unlabeled corpus of text, a fixed schema of relations (see Section 5.4), and a set of query entities. The task is to find all relation triples in the corpus that have as a subject the query entity, and as a relation one of the defined relations. This can be viewed intuitively as populating Wikipedia Infoboxes from a large unstructured corpus of text.

We compare our approach to the University of Washington submission to TAC-KBP 2013 [96]. Their system used OpenIE v4.0 (a successor to Ollie) run over the KBP corpus and then they generated a mapping from the extracted relations to the fixed schema. Unlike our system, Open IE v4.0 employs a semantic role component extracting structured SRL frames, alongside a conventional open IE system. Furthermore, the UW submission allows for extracting relations and entities from substrings of an open IE triple argument. For example, from the triple *(Smith; was appointed; acting director of Acme Corporation)*, they extract that Smith is employed by Acme Corporation. We disallow such extractions, passing the burden of finding correct precise extractions to the open IE system itself (see Section 5.3).

For entity linking, the UW submission uses Tom Lin's entity linker [68]; our submission uses the Illinois Wikifier [85] without the relational inference component, for efficiency. For coreference, UW uses the Stanford coreference system [62]; we employ a variant of the simple coref system described in [84].

We report our results in Table 5.5.[4] UW Official refers to the official submission in the 2013 challenge; we show a 3.1 F_1 improvement (to 22.7 F_1) over this submission, evaluated using a comparable approach. A common technique in KBP systems but not employed by the official UW submission in 2013 is to add alternate names based on entity linking and

[4]All results are reported with the `anydoc` flag set to true in the evaluation script, meaning that only the truth of the extracted knowledge base entry and not the associated provenance is scored. In absence of human evaluators, this is in order to not penalize our system unfairly for extracting a new correct provenance.

System	P	R	F_1
UW Official*	**69.8**	11.4	19.6
Ollie†	57.4	4.8	8.9
+ Nominal Rels*	57.7	11.8	19.6
Our System			
- Nominal Rels†	64.3	8.6	15.2
+ Nominal Rels*	61.9	13.9	22.7
+ Alt. Name	57.8	17.8	27.1
+ Alt. Name + Website	58.6	**18.6**	**28.3**

Table 5.5: A summary of our results on the end-to-end KBP Slot Filling task. UW official is the submission made to the 2013 challenge. The second row is the accuracy of Ollie embedded in our framework, and of Ollie evaluated with nominal relations from our system. Lastly, we report our system, our system with nominal relations removed, and our system combined with an alternate names detector and rule-based website detector. Comparable systems are marked with a dagger† or asterisk*.

coreference. Additionally, websites are often extracted using heuristic name-matching as they are hard to capture with traditional relation extraction techniques. If we make use of both of these, our end-to-end accuracy becomes 28.2 F_1.

We attempt to remove the variance in scores from the influence of other components in an end-to-end KBP system. We ran the Ollie open IE system [74] in an identical framework to ours, and report accuracy in Table 5.5. Note that when an argument to an Ollie extraction contains a named entity, we take the argument to be that named entity. The low performance of this system can be partially attributed to its inability to extract nominal relations. To normalize for this, we report results when the Ollie extractions are supplemented with the nominal relations produced by our system (Ollie + Nominal Rels in Table 5.5). Conversely, we can remove the nominal relation extractions from our system; in both cases we outperform Ollie on the task.

5.5.1 Discussion

We plot a precision/recall curve of our extractions in Figure 5.4 in order to get an informal sense of the calibration of our confidence estimates. Since confidences only apply to standard extractions, we plot the curves without including any of the nominal relations. The

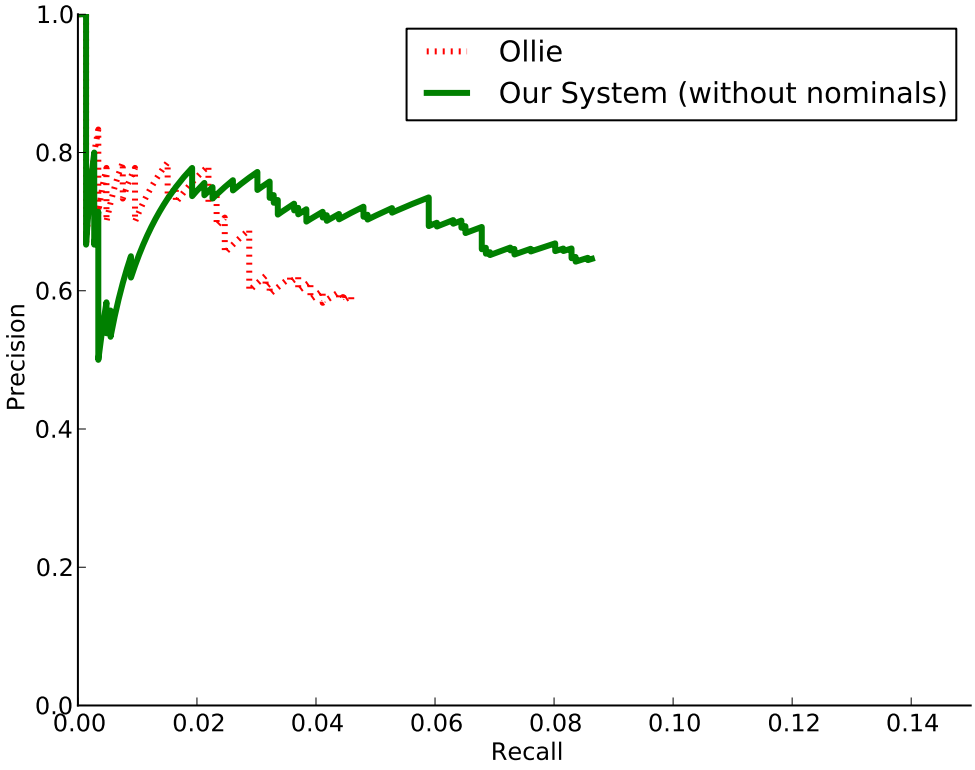


Figure 5.4: A precision/recall curve for Ollie and our system (without nominals). For clarity, recall is plotted on a range from 0 to 0.15.

confidence of a KBP extraction in our system is calculated as the sum of the confidences of the open IE extractions that support it. So, for instance, if we find (Obama; be bear in; Hawaii) n times with confidences $c_1 \ldots c_n$, the confidence of the KBP extraction would be $\sum_{i=0}^{n} c_i$. It is therefore important to note that the curve in Figure 5.4 necessarily conflates the confidences of individual extractions, and the frequency of an extraction.

With this in mind, the curves lend some interesting insights. Although our system is very high precision on the most confident extractions, it has a large dip in precision early in the curve. This suggests that the model is extracting multiple instances of a bad relation. Systematic errors in the clause splitter are the likely cause of these errors. While the approach of splitting sentences into clauses generalizes better to out-of-domain text, it

is reasonable that the errors made in the clause splitter manifest across a range of sentences more often than the fine-grained patterns of Ollie would.

On the right half of the PR curve, however, our system achieves both higher precision and extends to a higher recall than Ollie. Furthermore, the curve is relatively smooth near the tail, suggesting that indeed we are learning a reasonable estimate of confidence for extractions that have only one supporting instance in the text – empirically, 46% of our extractions.

In total, we extract 42 662 862 open IE triples which link to a pair of entities in the corpus (i.e., are candidate KBP extractions), covering 1 180 770 relation types. 202 797 of these relation types appear in more than 10 extraction instances; 28 782 in more than 100 instances, and 4079 in more than 1000 instances. 308 293 relation types appear only once. Note that our system over-produces extractions when both a general and specific extraction are warranted; therefore these numbers are an overestimate of the number of semantically meaningful facts.

For comparison, Ollie extracted 12 274 319 triples, covering 2 873 239 relation types. 1 983 300 of these appeared only once; 69 010 appeared in more than 10 instances, 7951 in more than 100 instances, and 870 in more than 1000 instances.

Chapter 6

Open Domain Question Answering

6.1 Introduction

Question answering is an important task in NLP, but becomes increasingly difficult as the domain diverges from that of existing lexical resources. In these cases, viewing question answering as textual entailment over a very large premise set can offer a means of generalizing reliably over these open domains. We present an approach for answering 4^{th} grade science exam questions using textual entailment methods which combines logical reasoning and broad-coverage lexical methods in a coherent framework based around natural logic.

A natural approach to textual entailment is to treat it as a logical entailment problem. However, this high-precision approach is not feasible in cases where a formal proof is difficult or impossible. For example, consider the following hypothesis (H) and its supporting premise (P) for the question *Which part of a plant produces the seeds?*:

P: *Ovaries are the female part of the flower, which produces eggs that are needed for making seeds.*

H: *A flower produces the seeds.*

In contrast, even a simple lexical overlap classifier could correctly predict the entailment. In fact, such a bag-of-words entailment model has been shown to be surprisingly effective on the Recognizing Textual Entailment (RTE) challenges [69]. On the other hand,

such methods are also notorious for ignoring even trivial cases of nonentailment that are easy for natural logic, e.g., recognizing negation. For example:

P: *Eating candy for dinner is an example of a poor health habit.*
H: *Eating candy is an example of a good health habit.*

We present an approach to leverage the benefits of both methods. Natural logic – a proof theory over the syntax of natural language – offers a framework for logical inference which is already familiar to lexical methods. As an inference system searches closer to a valid premise, the candidates it explores will generally become more lexically similar to that premise.

We therefore extend a natural logic inference engine in two key ways: first, we handle relational entailment and meronymy, increasing the total number of inferences that can be made. For example, a hypothesis *a flower produces the seeds* can yield a candidate premise *a flower grows seeds*, because *grow* entails *produce*. We further implement an *evaluation function* which quickly provides an estimate for how likely a candidate premise is to be supported by the knowledge base, without running the full search. This can then more easily match a known premise (e.g., *seeds grow inside a flower*) despite still not matching exactly.

We present the following contributions: (1) we extend the classes of inferences NaturalLI can perform on real-world sentences by incorporating relational entailment and meronymy, and by operating over dependency trees; (2) we augment NaturalLI with an evaluation function to provide an estimate of entailment for any query; and (3) we run our system over the Aristo science questions corpus, achieving the best published numbers.

6.2 Improving Inference in NaturalLI

We extend NaturalLI in a few key ways to improve its coverage for question answering tasks over complex sentences. We adapt the search algorithm to operate over dependency trees rather than the surface forms (Section 6.2.1). We enrich the class of inferences warranted by natural logic beyond hypernymy and operator rewording to also encompass meronymy and relational entailment (Section 6.2.2). Lastly, we handle token insertions

during search more elegantly (Section 6.2.3).

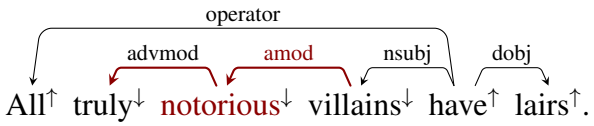
The general search algorithm in NaturalLI is parametrized as follows: First, an order is chosen to traverse the tokens in a sentence. For example, the original paper traverses tokens left-to-right. At each token, one of three operations can be performed: these are *deleting* a token (corresponding to inserting a word during natural logic inference), *mutating* a token, and *inserting* a token (again, corresponding to deleting a token in the resulting proof derivation.) For brevity, we will refer to insertions during inference (deletions during search) as *deletions*, and vice versa for *insertions*.

6.2.1 Natural logic over Dependency Trees

Operating over dependency trees rather than a token sequence requires reworking (1) the semantics of deleting a token during search, and (2) the order in which the sentence is traversed.

Recent work by Angeli et al. [5] defined a mapping from Stanford Dependency relations to the associated lexical relation deleting the dependent subtree would induce. We adapt this mapping to yield the relation induced by *inserting* a given dependency edge, corresponding to our deletions in search; we also convert the mapping to use Universal Dependencies [36]. This now lends a natural deletion operation: at a given node, the subtree rooted at that node can be deleted to induce the associated natural logic relation.

For example, we can infer that *all truly notorious villains have lairs* from the premise *all villains have lairs* by observing that deleting an *amod* arc induces the relation $\sqsupseteq$, which in the downward polarity context of *villains*$^{\downarrow}$ projects to $\sqsubseteq$ (entailment):



This leaves the question of the order in which to traverse the tokens in the sentence. The natural order is a breadth-first traversal of the dependency tree. This avoids repeated deletion of nodes, as we do not have to traverse a deleted subtree.

An admittedly rare but interesting subtlety is the effect mutating an operator has on the polarity of its arguments – for example, mutating *some* to *all*. There are cases where we

must mutate the argument to the operator before the operator itself, as well as cases where we must mutate the operator before its arguments. Consider, for instance:

P: *All felines have a tail*
H: *Some cats have a tail*

where we must first mutate *cat* to *feline*, versus:

P: *All cats have a tail*
H: *Some felines have a tail*

where we must first mutate *some* to *all*. Therefore, our traversal first visits each operator, then performs a breadth-first traversal of the tree, and then visits each operator a second time.

6.2.2 Meronymy and Relational Entailment

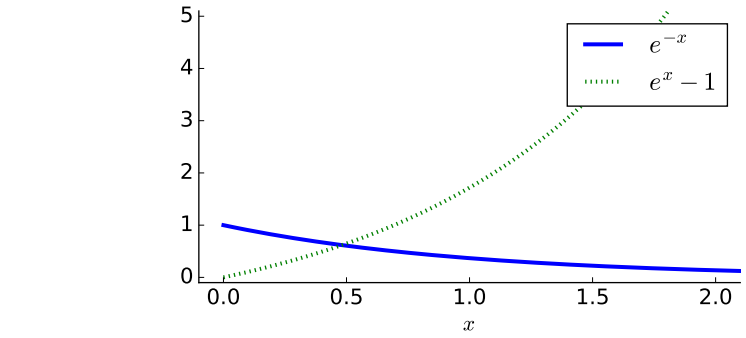
Although natural logic and the underlying monotonicity calculus has only been explored in the context of hypernymy, the underlying framework can be applied to any partial order.

Natural language operators can be defined as a mapping from denotations of objects, to truth values.[1] The domain of word denotations is then ordered by the subset operator, corresponding to ordering by hypernymy over the words. However, hypernymy is not the only useful partial ordering over denotations. We include two additional orderings as motivating examples: relational entailment and meronymy.

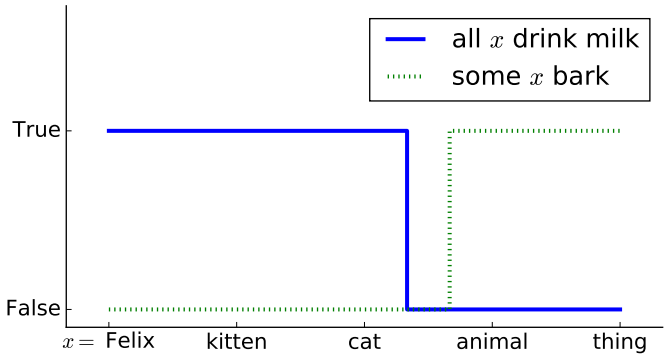
Relational Entailment For two verbs v_1 and v_2, we define $v_1 \leq v_2$ if the first verb entails the second. In many cases, a verb v_1 may entail a verb v_2 even if v_2 is not a hypernym of v_1. For example, to *sell* something (hopefully) entails *own*ing that thing. Apart from context-specific cases (e.g., *orbit* entails *launch* only for man-made objects), these hold largely independent of context.

This information was incorporated using data from VERBOCEAN [28], adapting the

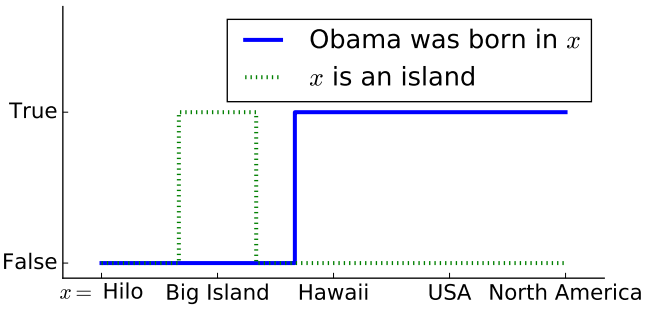
[1]Truth values are a trivial partial order corresponding to entailment: if $t_1 \leq t_2$ (i.e., $t_1 \sqsubseteq t_2$), and you know that t_1 is true, then t_2 must be true.



(a)



(b)



(c)

Figure 6.1: An illustration of monotonicity using different partial orders. (a) Monotonicity illustrated over real numbers: $e^x - 1$ is monotone whereas e^{-x} is antitone. (a) The monotonicity of *all* and *some* in their first arguments, over a domain of denotations. (b) An illustration of the *born in* monotone operator over the meronymy hierarchy, and the operator *is an island* as neither monotone or antitone.

confidence weights as transition costs. VERBOCEAN uses lexicosyntactic patterns to score pairs of verbs as candidate participants in a set of relations. We approximate the VERBOCEAN relations $stronger\text{-}than(v_1, v_2)$ and $happens\text{-}before(v_2, v_1)$ to indicate that v_1 entails v_2.

Meronymy The most salient use-case for meronymy is with locations. For example, if Obama was born in Hawaii, then we know that Obama was born in America, because Hawaii is a meronym of (part of) America. Unlike relational entailment and hypernymy, meronymy is operated on by a distinct set of operators: if *Hawaii is an island*, we cannot necessarily entail that *America is an island*.

We collect a set of 81 operators, which are manually labeled as monotonic or antitonic with respect to the meronymy hierarchy (e.g., *born in*, *visited*); these then compose in the usual way with the conventional operators (e.g., *some*, *all*). These operators consist of dependency paths of length 2 that co-occurred in newswire text with a named entity of type *PERSON* and two different named entities of type *LOCATION*, such that one location was a meronym of the other.

Meronymy transitions are drawn from instances of the relation *location-contains* in Freebase [13]. This relation exists between entities of type *location* in Freebase, where one location exists completely within the boundaries of the other location. We are able to use a weighting scheme analogous to that used for the hypernymy transitions.

6.2.3 Removing the Insertion Transition

Inserting words during search poses an inherent problem, as the space of possible words to insert at any position is on the order of the size of the vocabulary. In NaturalLI, this was solved by keeping a trie of possible insertions, and using that to prune this space. However, this is both computationally slow and adapts awkwardly to a search over dependency trees.

Therefore, this work instead opts to perform a bidirectional search: when constructing the knowledge base, we add not only the original sentence but also all entailments with subtrees deleted. For example, a premise of *some furry cats have tails* would yield two facts for the knowledge base: *some furry cats have tails* as well as *some cats have tails*.

The new challenge this introduces, of course, is the additional space required to store the new facts. To mitigate this, we hash every fact into a 64 bit integer, and store only the hashed value in the knowledge base. We construct this hash function such that it operates over a bag of edges in the dependency tree – in particular, the XOR of the hash of each dependency edge in the tree. This has two key properties: it allows us to be invariant to the word order of of the sentence, and more importantly it allows us to run our search directly over modifications to this hash function.

To elaborate, we notice that each of the two classes of operations our search is performing are done locally over a single dependency edge. When adding an edge, we can simply take the XOR of the hash saved in the parent state and the hash of the added edge. When mutating an edge, we XOR the hash of the parent state with the edge we are mutating, and again with the mutated edge. This results in an incidental contribution of having made NaturalLI significantly faster and more memory efficient – e.g., each search state can fit into only 32 bytes.

6.3 An Evaluation Function for NaturalLI

There are many cases – particularly as the length of the premise and the hypothesis grow – where despite our improvements NaturalLI will fail to find any supporting premises; for example:

P: *Food serves mainly for growth, energy and body repair, maintenance and protection.*

H: *Animals get energy for growth and repair from food.*

In addition to requiring reasoning with multiple implicit premises (a concomitant weak point of natural logic), a correct interpretation of the sentence requires fairly nontrivial nonlocal reasoning: *Food serves mainly for* $x \rightarrow$ *Animals get* x *from food.*

Nonetheless, there are plenty of clues that the two sentences are related, and even a simple entailment classifier would get the example correct. We build such a classifier and adapt it as an evaluation function inside NaturalLI in case no premises are found during search.

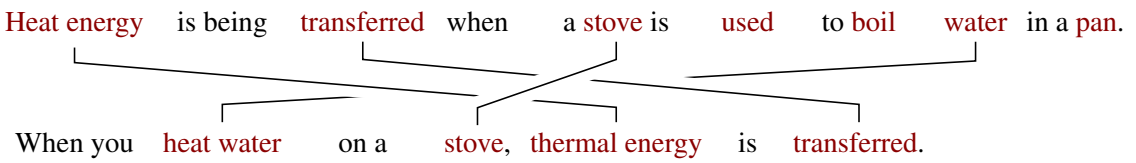


Figure 6.2: An illustration of an alignment between a premise and a hypothesis. Keyphrases can be multiple words (e.g., *heat energy*), and can be approximately matched (e.g., to *thermal energy*). In the premise, *used*, *boil* and *pan* are unaligned. Note that *heat water* is incorrectly tagged as a compound noun.

6.3.1 A Standalone Entailment Classifier

Our entailment classifier is designed to be as domain independent as possible; therefore we define only 5 unlexicalized real-valued features, with an optional sixth feature encoding the score output by the Solr information extraction system (in turn built upon Lucene). Additional features were not shown to improve performance on the training set. In fact, this classifier is a stronger baseline than it may seem: evaluating the system on RTE-3 [45] yielded **63.75**% accuracy – 2 points above the median submission.

All five of the core features are based on an alignment of keyphrases between the premise and the hypothesis. A keyphrase is defined as a span of text which is either (1) a possibly empty sequence of adjectives and adverbs followed by a sequence of nouns, and optionally followed by either *of* or the possessive marker (*'s*), and another noun (e.g., *sneaky kitten* or *pail of water*); (2) a possibly empty sequence of adverbs followed by a verb (e.g., *quietly pounce*); or (3) a gerund followed by a noun (e.g., *flowing water*). The verb *to be* is never a keyphrase. We make a distinction between a *keyphrase* and a *keyword* – the latter is a single noun, adjective, or verb.

We then align keyphrases in the premise and hypothesis by applying a series of sieves. First, all exact matches are aligned to each other. Then, prefix or suffix matches are aligned, then if either keyphrase contains the other they are aligned as well. Last, we align a keyphrase in the premise p_i to a keyphrase in the hypothesis h_k if there is an alignment between p_{i-1} and h_{k-1} and between p_{i+1} and h_{k+1}. This forces any keyphrase pair which is "sandwiched" between aligned pairs to be aligned as well. An example alignment is given

in Figure 6.2.

Features are extracted for the number of alignments, the numbers of alignments which do and do not match perfectly, and the number of keyphrases in the premise and hypothesis which were not aligned. A feature for the Solr score of the premise given the hypothesis is optionally included; we revisit this issue in the evaluation.

6.3.2 An Evaluation Function for Search

A version of the classifier constructed in Section 6.3.1, but over *keywords* rather than keyphrases can be incorporated directly into NaturalLI's search to give a score for each candidate premise visited. This can be thought of as analogous to the evaluation function in game-playing algorithms.

Using keywords rather than keyphrases is in general a hindrance to the fuzzy alignments the system can produce. Importantly though, this allows the feature values to be computed incrementally as the search progresses, based on the score of the parent state and the mutation or deletion being performed. For instance, if we are deleting a word which was previously aligned perfectly to the premise, we would subtract the weight for a perfect and imperfect alignment, and add the weight for an unaligned premise keyphrase.

In addition to finding entailments from candidate premises, our system also allows us to encode a notion of likely negation. We can consider the following two statements naïvely sharing every keyword. Each token marked with its polarity:

P: $some^{\uparrow}$ $cats^{\uparrow}$ $have^{\uparrow}$ $tails^{\uparrow}$
H: $no^{\uparrow}$ $cats^{\downarrow}$ $have^{\downarrow}$ $tails^{\downarrow}$

However, we note that all of the keyword pairs are in opposite polarity contexts. We can therefore define a pair of keywords as *matching* in NaturalLI if the following two conditions hold: (1) their lemmatized surface forms match exactly, and (2) they have the same polarity in the sentence. The second constraint encodes a good approximation for negation. To illustrate, consider the polarity signatures of common operators:

Operators	Subj. polarity	Obj. polarity
Some, few, etc.	↑	↑
All, every, etc.	↓	↑
Not all, etc.	↑	↓
No, not, etc.	↓	↓
Most, many, etc.	–	↑

We note that most contradictory operators (e.g., *some*/*no*; *all*/*not all*) induce the exact opposite polarity on their arguments. The conspicuous counterexamples to this is the operator pairs *all* and *no*, which have the same monotonicity in their subjects but are nonetheless negations of each other. Otherwise, pairs of operators which share half their signature are usually compatible with each other (e.g., *some* and *all*). In all, we consider this a good simple approximation at a low cost.

This suggests a criterion for likely negation: If the highest classifier score is produced by a contradictory candidate premise, we have reason to believe that we may have found a contradiction. To illustrate with our example, NaturalLI would mutate *no cats have tails* to *the cats have tails*, at which point it has found a contradictory candidate premise which has perfect overlap with the premise *some cats have tails*. Even had we not found the exact premise, this suggests that the hypothesis is likely false.

6.4 System Design

A key design principle behind NaturalLI and its extensions was the hypothesis that the speed at which the search could be performed would be a significant factor in the success of the the system. In fact, NaturalLI over dependency trees runs at nearly 1M search states per second per core, and around 100k search states when the evaluation function is enabled. This section details some of the engineering tricks used to allow the system to run efficiently.

Data Structures Like many NLP applications, the fundamental bottleneck for speed is the memory bus rather than the CPU's clock speed. Therefore, NaturalLI was designed with a careful eye on managing memory. The search problem is at its core composed of two data

structures: a fringe for the search boundary and a set of terminal states. In a uniform cost search (like the one NaturalLI runs), the fringe is conceptually a priority queue; we use the sequence heap described in Sanders [91]. This data structure is more friendly to the cache than a more conventional binary heap, and empirically obtains a noticeable performance improvement for large heap sizes.

An interesting empirical finding was that storing the set of terminal states in a btree set was substantially more efficient than the default hash map implementations. We hypothesize that this again has to do with caching behavior. Although a btree has a theoretical lookup time of $O(\log(n))$, in practice the top of the tree remains in cache between lookups. Therefore, fewer requests have to be made to main memory, and many of those are at least roughly co-located. The default hash map implementation (and even open addressed maps), by contrast, often have to make many random accesses to main memory when there is a hash code collision. This is particularly bad in the default STL implementation of `unordered_map`, where the list at each bucket is implemented as a linked list.

The implementation of the mutation graph is less important, as it tends to be significantly smaller than both the fringe and the knowledge base of premises. The graph is implemented straightforwardly; the only important subtlety is to ensure that the set of candidate edges (i.e., incoming edges) are located in the same block of memory. Fortunately, this is the natural way to store a graph in memory anyways.

Cache Optimization Selecting cache-aware data structures is half the battle, but some attention should also be paid to the data being put into these structures. Again, the two relevant bits of data are (1) search states, which go into the fringe, and (2) facts in the knowledge base, which go into the premise set.

We begin with the knowledge base facts. Recall that a fact is a plain text sentence, and the associated dependency tree. Naively, this takes up a fair bit of memory. Fortunately, all we need the knowledge base for is to check if a given fact is contained in the KB – we never have to reconstruct the sentence from the stored fact. This lends itself to a natural solution: we hash each fact, and store that hash in the knowledge base. In practice, we use 64 bits to hash the fact, though of course this can be increased or decreased.

In addition to making the knowledge base smaller, the fact that we are hashing facts also

means that our search states can be compacted significantly. Rather than storing an entire sentence, a search state only needs to keep the hash of the mutated sentence, and update that hash as the search progresses. This, however, does mean that the hash function has to be a bit carefully designed: in particular, if we know what natural logic mutation we are applying, we must be able to apply it *directly to the hash* without knowing the underlying sentence. Three insights make this possible:

1. If the hash function is an order-invariant hash of the edges in the dependency tree, then it is robust to reordering, deletion order, etc. We accomplish this simply by defining the hash of a sentence to be the xor of the hash of the dependency edges – (governor, relation, dependent) triples – in the sentence.

2. Modulo deletions (in reverse inference), the *structure* of the dependency tree of the sentence cannot be changed by natural logic inferences. Mutations, of course, only mutate a given lexical item. Insertions (in search, therefore deletions in the inference) are in turn handled before the search by the OpenIE system maximally shortening clauses. Therefore, we can store the structure of the dependency tree only once, and appeal to that structure for each search state.

3. We mutate a lexical item in bulk at once, and never return to it during the search. Therefore, once we have finished mutating an item in the search, we do not have to remember what that lexical item is. The exception here is quantifiers, which require special treatment anyways.

These insights allow us to compress a search state into only 32 bytes – exactly half of a cache line. The 32 bytes (256 bits) are allocated as follows:

64 bits The hash of the fact in the current search state. This is used to check if we've reached a terminal state

6 bits The index of the sentence we are currently mutating. This limits the length of queries NaturalLI can handle to a reasonable 64 tokens.

24 bits The word we are mutating. This allows for a vocabulary size of 16.7M; words outside of this vocabulary are treated as unknown words. We need this to compute valid

mutations of the current word, and to recover the edge we are mutating to update the hash.

5 bits The word sense of the word we are mutating. We need this to compute valid mutations of the current word. The value 0 is reserved for "unknown sense" – e.g., for nearest neighbors mutations. Entries in WordNet with more than 30 senses have their remaining senses lumped into a last catch-all class.

24 bits The governor of the word we are mutating. We need this to recover the edge we are mutating to update the hash.

1 bit The presumed truth of the fact in the search so far. See the natural logic finite state diagram.

1 bit A marker for whether the search has mutated all quantifiers once already.

25 bits A backpointer to the search state this state was generated from. This limits NaturalLI to searching over at most 33M states. Note that this is somewhat unconventional: usually, the backpointer is stored as a pointer to the previous state (64 bits); however, managing the history explicitly in a flat array allows us to cut this space by over half. Since we add to the history array linearly, we maintain good caching properties.

64 bits The monotonicity signatures of the quantifiers in the sentence. This supports up to 8 quantifiers, each of which is 8 bits: 2 bits for monotonicity and 2 bits for additivity/multiplicativity for both the subject and the object of the quantifier.

Updating the hash of a fact given a mutation is then as easy as reconstructing the hash of the edge we are mutating, xoring that with the current hash, and xoring in the new edge's hash. Deleting an edge involves computing the subtree that was deleting, and xoring out each deleted edge. The root edge can be computed as above, and since we're searching through the tree in a breadth-first fashion, the children edges will be identical to the original tree.

Cycle Detection Since we are running a graph search, of course there are cycles in the search. Trivially, if we mutate a word to its synonym, we can mutate the synonym back to the original word. The natural solution to this problem is to keep a set of seen states around, and at each search tick look up whether we have already seen the node we are considering. On the other extreme, the problem could simply be ignored on the theory that a faster search computationally would be more valuable than a faster search theoretically. Empirically, we found that both options were slow, and the best solution was to implement an approximate cycle detection algorithm. At each search state, we search up the tree through the states' backpointers to a depth k (5 in our experiments), and ignore the state if we found an identical state in this k node *cycle memory*. Why this approach was empirically more effective remains an open question – in principle, keeping around a set of all seen nodes should not be substantially slower. Our theory is that this again has to do with caching behavior. The cycle memory of nodes pop'd at near the same time will be nearly the same; therefore, this simple cycle detection mechanism is not likely to make many main memory accesses. Nonetheless, it catches the most common cases of an already-visited node being re-visited.

6.5 Evaluation

We evaluate our entailment system on the Regents Science Exam portion of the Aristo dataset [29, 30]. The dataset consists of a collection of multiple-choice science questions from the New York Regents 4$^{\text{th}}$ Grade Science Exams [82]. Each multiple choice option is translated to a candidate hypotheses. A large corpus of text is given as a knowledge base; the task is to find support in this knowledge base for the hypothesis.

Our system is in many ways well-suited to the dataset. Although certainly many of the facts require complex reasoning (see Section 6.5.4), the majority can be answered from a single premise. Unlike FraCaS or the RTE challenges, however, the task does not have explicit premises to run inference from, but rather must infer the truth of the hypothesis from a large collection of supporting text. This makes NaturalLI a good candidate for an approach. However, the queries in the Aristo dataset are relatively longer, complete sentences than the common sense reasoning task in Angeli and Manning [3], necessitating

the improvements in Sections 6.2 and 6.3.

6.5.1 Data Processing

We make use of two collections of unlabeled corpora for our experiments. The first of these is the Barron's study guide (BARRON'S), consisting of 1200 sentences. This is the corpus used by Hixon et al. [50] for their conversational dialog engine Knowbot, and therefore constitutes a more fair comparison against their results. However, we also make use of the full SCITEXT corpus [31]. This corpus consists of 1 316 278 supporting sentences, including the Barron's study guide alongside simple Wikipedia, dictionaries, and a science textbook.

Since we lose all document context when searching over the corpus with NaturalLI, we first pre-process the corpus to resolve high-precision cases of pronominal coreference, via a set of simple high-precision sieves. Filtering to remove duplicate sentences and sentences containing non-ASCII characters yields a total of 822 748 facts in the supporting corpus.

These sentences were then indexed using Solr. The set of promising premises for the soft alignment in Section 6.3, as well as the Solr score feature in the lexical classifier (Section 6.3.1), were obtained by querying Solr using the default similarity metric and scoring function. On the query side, questions were converted to answers using the same methodology as Hixon et al. [50]. In cases where the question contained multiple sentences, only the last sentence was considered.

6.5.2 Training an Entailment Classifier

To train a soft entailment classifier, we needed a set of positive and negative entailment instances. These were collected on Mechanical Turk. In particular, for each true hypothesis in the training set and for each sentence in the Barron's study guide, we found the top 8 results from Solr and considered these to be candidate entailments. These were then shown to Turkers, who decided whether the premise entailed the hypothesis, the hypothesis entailed the premise, both, or neither. The data was augmented with additional negatives, collected by taking the top 10 Solr results for each false hypothesis in the training set. This yielded a total of 21 306 examples.

System	Barron's		SCITEXT	
	Train	Test	Train	Test
KNOWBOT (held-out)	45	–	–	–
KNOWBOT (oracle)	57	–	–	–
Solr Only	49	42	62	58
Classifier	53	52	68	60
+ Solr	53	48	66	64
Evaluation Function	52	**54**	61	63
+ Solr	50	45	62	58
NaturalLI	52	51	65	61
+ Solr	**55**	49	73	61
+ Solr + Classifier	**55**	49	**74**	**67**

Table 6.1: Accuracy of various systems on the Aristo science questions dataset. Results are reported using only the Barron's study guide as the supporting text, and using all of SCITEXT. KNOWBOT is the dialog system presented in Hixon et al. [50]. The held-out version uses additional facts from other question's dialogs; the oracle version made use of human input on the question it was answering. The test set did not exist at the time KNOWBOT was published.

The scores returned from NaturalLI incorporate negation in two ways: if NaturalLI finds a contradictory premise, the score is set to zero. If NaturalLI finds a soft negation (see Section 6.3.2), and did not find an explicit supporting premise, the score is discounted by 0.75 – a value tuned on the training set. For all systems, any premise which did not contain the candidate answer to the multiple choice query was discounted by a value tuned on the training set.

6.5.3 Experimental Results

We present results on the Aristo dataset in Table 6.1, alongside prior work and strong baselines. The test set for this corpus consists of only 68 examples, and therefore both perceived large differences in model scores and the apparent best system should be interpreted cautiously. We propose that NaturalLI consistently achieves the best training accuracy, and is

more stable between configurations on the test set. For instance, it may be consistently discarding lexically similar but actually contradictory premises that often confuse some subset of the baselines.

KNOWBOT is the dialog system presented in Hixon et al. [50]. We report numbers for two variants of the system: *held-out* is the system's performance when it is not allowed to use the dialog collected from humans for the example it is answering; *oracle* is the full system.

We additionally present three baselines. The first simply uses Solr's IR confidence to rank entailment (*Solr Only* in Table 6.1). The max IR score of any premise given a hypothesis is taken as the score for that hypothesis. Furthermore, we report results for the entailment classifier defined in Section 6.3.1 (*Classifier*), optionally including the Solr score as a feature. We also report performance of the evaluation function in NaturalLI applied directly to the premise and hypothesis, without any inference (*Evaluation Function*).

Last, we evaluate NaturalLI with the improvements presented in this chapter (*NaturalLI* in Table 6.1). We additionally tune weights for a simple model combination with (1) Solr (with weight 6:1 for NaturalLI) and (2) the standalone classifier (with weight 24:1 for NaturalLI). Empirically, both parameters were observed to be fairly robust.

6.5.4 Discussion

We analyze some common types of errors made by the system on the training set. The most common error can be attributed to the question requiring complex reasoning about multiple premises. 29 of 108 questions in the training set (26%) contain multiple premises. Some of these cases can be recovered from (e.g., *This happens because the smooth road has less friction.*), while others are trivially out of scope for our method (e.g., *The volume of water most likely decreased.*).

Another class of errors which deserves mention are cases where a system produces the same score for multiple answers. This occurs fairly frequently in the standalone classifier (7% of examples in training; 4% loss from random guesses), and especially often in NaturalLI (11%; 6% loss from random guesses). This offers some insight into why incorporating other models – even with low weight – can offer significant boosts in the performance

of NaturalLI. Both this and the previous class could be further mitigated by having a notion of a *process*; e.g., as in Berant et al. [12].

Other questions are simply not supported by any single sentence in the corpus. For example, *A human offspring can inherit blue eyes* has no support in the corpus that does not require significant multi-step inferences.

A remaining chunk of errors are of course classification errors. For example, *Water freezing is an example of a gas changing to a solid* is marked as the best hypothesis, supported incorrectly by *An ice cube is an example of matter that changes from a solid to a liquid to a gas*, which after mutating *water* to *ice cube* matches every keyword in the hypothesis.

Chapter 7

Conclusions

In this dissertation, I have explored methods to leverage natural logic for extracting open domain knowledge from large-scale text corpora. Unlike fixed-schema knowledge bases, this approach allows querying arbitrary facts. Unlike open-domain knowledge bases – such as Open Information Extraction approaches – this approach (1) does not limit the representation of facts to subject/relation/object triples; and (2) allows for rich inferences to be made so that we can find facts which are not only in the knowledge base, but also *inferred* by some known fact. From the other direction, unlike shallow information retrieval approaches, which also operate over large text corpora, the approach in this dissertation is robust to logical subtleties like negation and monotonicity. We have applied this method to three areas: we have shown that we can predict the truth of common-sense facts with high precision and substantially higher recall than using a fixed knowledge base. We have shown that we can segment complex sentences into short atomic propositions, and that this is effective for a practical downstream task of knowledge base population. Lastly, we have shown that we can incorporate an *evaluation function* encoding a simple entailment classifier, and that the hybrid of this evaluation function and our natural logic search is effective for question answering.

In Chapter 3 I reviewed the current theory behind natural logic as a logical formalism. We reviewed a theory of denotations over lexical items, a notion of monotonicity over arguments to quantifiers and other functional denotations, and then introduced Monotonicity Calculus as a logic to reason over these monotonic functions. We then introduced *exclusion*

to deal with antonomy and negation, and showed how we can extend Monotonicity Calculus to incorporate this additional expressive power. Lastly, I introduced a brief sketch of a propositional natural logic, which would allow for jointly reasoning about multiple natural language utterances (for instance, the disjunctive syllogism). I encourage future research into this propositional natural logic, and further research into the use of natural logic in place of conventional (e.g., first-order) logics for language tasks.

In Chapter 4, I introduce NaturalLI – a large-scale natural logic reasoning engine for common sense facts. I show that natural logic inference can be cast as a search problem, and that the *join table* of MacCartney and Manning [71] can be more elegantly represented as a finite state machine we transition through during search. I show that we can not only perform strictly warranted searches, but also learn a confidence for likely valid mutations; this allows the system to improve its recall by matching not only strictly valid premises, but also likely valid premises that it finds through the search. I show that our system improves recall by $4\times$ over lemmatized knowledge base lookup when assessing whether common-sense facts are true given a source corpus of 270 million unique short propositions.

In Chapter 5, I move from short propositions to longer sentences, and introduce a method for segmenting and trimming a complex sentence into the types of short utterances that NaturalLI can operate over. This is done in two steps: First, complex sentences are broken into clauses, where each clause expresses one of the main propositions of the sentence (alongside potentially many additional qualifiers). This is done by casting the problem as a search task: as we search down the dependency tree, each edge either corresponds to a split clause (possibly interpreting the subject / object of the governor as the subject of the dependent), or the search is told to stop, or the search is told to continue down that branch of the tree but not to split off that clause. These clauses are then maximally shortened according to valid natural logic mutations to yield maximally informative atomic propositions. These propositions can then either be used as propositions for a system like NaturalLI, or segmented further into OpenIE relation triples. Using segmented triples from this system outperforms prior OpenIE systems on a downstream relation extraction task by 3 F_1.

In Chapter 6, we extend NaturalLI to operate over dependency trees and incorporate the method for creating atomic propositions from Chapter 5 to allow NaturalLI to operate over a more complex premise set. In addition, we introduce a method for combining a

shallow entailment classifier with the more formal NaturalLI search. At each step of the search, this classifier is run against a set of candidate premises; if any of these search states get close enough to a candidate premise according to the classifier, the fact is taken to be possibly true. This behaves as a sort of evaluation function – akin to evaluation functions in gameplaying algorithms – and allows for both (1) improving the recall of NaturalLI, and (2) creating a reasonable confidence value for likely entailment or contradiction even when the query cannot be formally proven or disproven. I show that this method outperforms both strong IR baselines and prior work on answering multiple choice 4^{th} grade science exam questions.

There are a number of interesting and natural directions for future work in this area, which I will briefly discuss below:

Propositional Natural Logic Section 3.3 sketches a simplistic propositional natural logic, based around a simple proof theory. However, this is presented both without a formal proof of consistency or completeness, and without an associated model theory. Furthermore, I have skirted issues of proper quantification and other first-order phenomena. It would clearly be beneficial to the NLP community to have a natural logic which can operate over multiple premises, and more work in this area is I believe both useful and exciting.

Downstream Applications This dissertation has presented a means of inferring the truth or falsehood of common-sense facts, but has only scratched the surface of downstream applications which can make use of this information. There is I believe an interesting avenue of research which attempts to improve core NLP algorithms beyond what can be obtained with statistical methods by leveraging the common-sense knowledge acquired from large unsupervised text corpora. For example, perhaps a parser which is more aware of facts about the world could correctly disambiguate prepositional attachments (e.g., *I ate the cake with a fork / cherry*).

Natural Logic for Cross-Domain Question Answering The applications in this dissertation have focused on factoid-style true/false queries. However, much of question answering is either (1) non-factoid (e.g., procedural) questions, or (2) requires finding a textual

answer to the question (e.g., *Who is the president of the US?)*. Extending NaturalLI to handle these questions is a potential means of creating a truly cross-domain question-answering system. If all premises are encoded in text, and all questions are given in text, then there is no notion of a schema or domain-specific model / named entity tag set / etc. which would limit the scope of questions that could be asked of the system. For the first time, the same system could be asked both what color the sky is, and where Barack Obama was born.

I hope that this dissertation can inspire research in the direction of open-domain, broad coverage knowledge extraction, and encourage researchers to consider natural logic and its extensions as the foundation for storing and reasoning about this sort of knowledge. As humans, we have chosen language as the means we store and represent knowledge, and I believe intelligent computers should do the same.

Bibliography

[1] David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Müller, Maarten de Rijke, and Stefan Schlobach. Using wikipedia at the trec qa track. In *TREC*, 2004.

[2] Gabor Angeli and Christopher D. Manning. Philosophers are mortal: Inferring the truth of unseen facts. In *CoNLL*, 2013.

[3] Gabor Angeli and Christopher D. Manning. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*, 2014.

[4] Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. Combining distant and partial supervision for relation extraction. In *EMNLP*, 2014.

[5] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *ACL*, 2015.

[6] Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang*, 2009.

[7] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. *Proc. Linguistic Annotation Workshop*, 2013.

[8] DAILLE Béatrice. *Approche mixte pour l'extraction automatique de terminologie:*

statistique lexicale et filtres linguistiques. PhD thesis, Thèse de Doctorat. Université de Paris VII, 1994.

[9] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*, 2014.

[10] Jonathan Berant, Ido Dagan, and Jacob Goldberger. Global learning of typed entailment rules. In *Proceedings of ACL*, Portland, OR, 2011.

[11] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, 2013.

[12] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. Modeling biological processes for reading comprehension. In *EMNLP*, 2014.

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[14] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[15] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

[16] Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics, 2005.

[17] Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. Wide-coverage semantic representations from a CCG parser. In *Proceedings of Coling*, Geneva, Switzerland, 2004. COLING.

[18] Samuel Bowman. Can recursive neural tensor networks learn logical reasoning? *ICLR (arXiv:1312.6192)*, 2014.

[19] Samuel R Bowman, Christopher Potts, and Christopher D Manning. Recursive neural networks can learn logical semantics. *ACL-IJCNLP 2015*, 2015.

[20] Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. *Linguistic Data Consortium*, 2006.

[21] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, 2007.

[22] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[23] Angel X. Chang and Christopher D. Manning. TokensRegex: Defining cascaded regular expressions over tokens. Technical Report CSTR 2014-02, Department of Computer Science, Stanford University, 2014.

[24] Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*, 2013.

[25] Yi Chen, Ming Zhou, and Shilong Wang. Reranking answers for definitional qa using language modeling. In *ACL*, 2006.

[26] Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Matthew Snover, Javier Artiles, Marissa Passantino, and Heng Ji. CUNY-BLENDER. In *TAC-KBP*, 2010.

[27] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[28] Timothy Chklovski and Patrick Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*, volume 2004, 2004.

[29] Peter Clark. Elementary school science and math tests as a driver for ai: Take the aristo challenge! *AIII*, 2015.

[30] Peter Clark, Philip Harrison, and Niranjan Balasubramanian. A study of the knowledge base requirements for passing an elementary science test. In *AKBC*, 2013.

[31] Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. Automatic construction of inference-supporting knowledge bases. 2014.

[32] Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. Using the framework. Technical report, The FraCaS Consortium, 1996.

[33] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *AAAI*, 1999.

[34] Hoa Trang Dang, Diane Kelly, and Jimmy J. Lin. Overview of the trec 2007 question answering track. In *TREC*, 2007.

[35] Marie-Catherine de Marneffe and Christopher D. Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 2008.

[36] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, 2014.

[37] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ACE) program–tasks, data, and evaluation. In *LREC*, 2004.

[38] Oren Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, 2011.

[39] Stefan Evert. *The statistics of word cooccurrences: word pairs and collocations*. PhD thesis, Universit at Stuttgart, 2005.

[40] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[41] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *KDD*, 2014.

[42] David Ferrucci, E Brown, J Chu-Carroll, J Fan, D Gondek, A Kalyanpur, A Lally, J Murdock, E Hyberg, J Prager, et al. The AI behind Watson. *The AI Magazine*, 2010.

[43] Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1):176–210, 1935.

[44] Gerhard Gentzen. Untersuchungen über das logische schließen. ii. *Mathematische Zeitschrift*, 39(1):405–431, 1935.

[45] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL workshop on textual entailment and paraphrasing*. Association for Computational Linguistics, 2007.

[46] Yoav Goldberg and Jon Orwant. A dataset of syntactic-ngrams over time from a very large corpus of english books. In **SEM*, 2013.

[47] Ralph Grishman and Bonan Min. New York University KBP 2010 slot-filling system. In *Proc. TAC 2010 Workshop*, 2010.

[48] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *ACL*, 2005.

[49] Andrew Hickl. Using discourse commitments to recognize textual entailment. In *Coling*, 2008.

[50] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. *NAACL*, 2015.

[51] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. *ACL*, 2012.

[52] Thomas Icard, III. Inclusion and exclusion in natural language. *Studia Logica*, 2012.

[53] Thomas Icard, III and Lawrence Moss. Recent progress on monotonicity. *Linguistic Issues in Language Technology*, 2014.

[54] Stanisław Jaśkowski. *On the rules of suppositions in formal logic*. Nakładem Seminarjum Filozoficznego Wydziału Matematyczno-Przyrodniczego Uniwersytetu Warszawskiego, 1934.

[55] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *NIPS*, 2012.

[56] Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference*, 2010.

[57] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of the 10th International Conference on Research on Computational Linguistics*, 1997.

[58] Hans Kamp. Two theories about adjectives. 1975.

[59] Rohit J. Kate. Transforming meaning representation grammars to improve semantic parsing. In *CoNLL*, Manchester, UK, 2008.

[60] Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. Learning to transform natural to formal languages. In *AAAI*, Pittsburgh, PA, 2005.

[61] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. 2013.

[62] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *CoNLL Shared Task*, 2011.

[63] Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 1995.

[64] Mike Lewis and Mark Steedman. Combined distributional and logical semantics. *TACL*, 1:179–192, 2013.

[65] P. Liang, M. I. Jordan, and D. Klein. Learning dependency-based compositional semantics. In *ACL*, 2011.

[66] Percy Liang and Christopher Potts. Bringing machine learning and compositional semantics together. *Annual Review of Linguistics*, 1(1):355–376, 2015.

[67] Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *NAACL-HLT*, 2006.

[68] Thomas Lin, Oren Etzioni, et al. No noun phrase left behind: detecting and typing unlinkable entities. In *EMNLP-CoNLL*, 2012.

[69] Bill MacCartney. *Natural Language Inference*. PhD thesis, Stanford, 2009.

[70] Bill MacCartney and Christopher D Manning. Natural logic for textual inference. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

[71] Bill MacCartney and Christopher D Manning. Modeling semantic containment and exclusion in natural language inference. In *Coling*, 2008.

[72] Bill MacCartney and Christopher D Manning. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*, 2009.

[73] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, 2014.

[74] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *EMNLP*, 2012.

[75] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial intelligence*, 1980.

[76] Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *EMNLP*, 2013.

[77] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL-HLT*, 2013.

[78] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, 2009.

[79] Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. Cogex: A logic prover for question answering. In *NAACL*, 2003.

[80] Neha Nayak, Mark Kowarsky, Gabor Angeli, and Christopher D. Manning. A dictionary of nonsubsective adjectives. Technical Report CSTR 2014-04, Department of Computer Science, Stanford University, October 2014.

[81] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *VLDB*, 2011.

[82] NYSED. The grade 4 elementary-level science test. `http://www.nysedregents.org/Grade4/Science/home.html`, 2014.

[83] Judea Pearl. Probabilistic semantics for nonmonotonic reasoning: A survey. *Principles of Knowledge Representation and Reasoning*, 1989.

[84] Glen Pink, Joel Nothman, and R. James Curran. Analysing recall loss in named entity slot filling. In *EMNLP*, 2014.

[85] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL*, 2011.

[86] Raymond Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1):81–132, 1980.

[87] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[88] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*, 2013.

[89] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[90] Víctor Manuel Sánchez Sánchez Valencia. *Studies on natural logic and categorial grammar*. PhD thesis, University of Amsterdam, 1991.

[91] Peter Sanders. Fast priority queues for cached memory. *Journal of Experimental Algorithmics (JEA)*, 5:7, 2000.

[92] Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *EMNLP*, 2010.

[93] Lenhart Schubert. Can we derive general world knowledge from texts? In *HLT*, 2002.

[94] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, 2013.

[95] Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Oren Etzioni, et al. Adapting open information extraction to domain-specific relations. *AI Magazine*, 2010.

[96] Stephen Soderland, John Gilmer, Robert Bart, Oren Etzioni, and Daniel S. Weld. Open information extraction to KBP relations in 3 hours. In *Text Analysis Conference*, 2013.

[97] Stephen G Soderland. *Learning text analysis rules for domain-specific natural language processing*. PhD thesis, University of Massachusetts, 1997.

[98] Rohini K. Srihari and Wei Li. Information extraction supported question answering. In *TREC*, 1999.

[99] Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. New York University 2011 system for KBP slot filling. In *Proceedings of the Text Analytics Conference*, 2011.

[100] Mihai Surdeanu. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Sixth Text Analysis Conference*, 2013.

[101] Mihai Surdeanu and Massimiliano Ciaramita. Robust information extraction with perceptrons. In *ACE07 Proceedings*, 2007.

[102] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *EMNLP*, 2012.

[103] Niket Tandon, Gerard de Melo, and Gerhard Weikum. Deriving a web-scale common sense fact database. In *AAAI*, 2011.

[104] Johan van Benthem. *Essays in logical semantics*. Springer, 1986.

[105] Johan van Benthem. A brief history of natural logic. Technical Report PP-2008-05, University of Amsterdam, 2008.

[106] Benjamin Van Durme, Phillip Michalak, and Lenhart K Schubert. Deriving generalized knowledge from corpora using wordnet abstraction. In *EACL*, 2009.

[107] Ellen M Voorhees. Question answering in TREC. In *Proceedings of the tenth international conference on Information and knowledge management*, 2001.

[108] Ellen M. Voorhees. Overview of the trec 2006. In *TREC*, 2006.

[109] Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *SIGIR*, 2000.

[110] Yotaro Watanabe, Junta Mizuno, Eric Nichols, Naoaki Okazaki, and Kentaro Inui. A latent discriminative model for compositional entailment relation recognition using natural logic. In *COLING*, 2012.

[111] Fei Wu and Daniel S Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on information and knowledge management*. ACM, 2007.

[112] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *ACL*. Association for Computational Linguistics, 2010.

[113] Wei Xu, Le Zhao, Raphael Hoffman, and Ralph Grishman. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL*, 2013.

[114] Limin Yao, Sebastian Riedel, and Andrew McCallum. Probabilistic databases of universal schema. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, 2012.

[115] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. TextRunner: Open information extraction on the web. In *ACL-HLT*, 2007.

[116] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, Portland, OR, 1996.

[117] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*. AUAI Press, 2005.

[118] Luke S. Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*, 2007.

[119] Ce Zhang and Christopher Ré. Dimmwitted: A study of main-memory statistical analytics. *VLDB*, 2014.