MODELING NATURAL LANGUAGE SEMANTICS IN LEARNED REPRESENTATIONS

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

Samuel Ryan Bowman

July 2016

Abstract

The last few years have seen many striking successes from artificial neural network models on hard natural language processing tasks. These models replace complex hand-engineered systems for extracting and representing the meanings of sentences with learned functions that construct and use their own internal vector-based representations. Though these learned representations are effective in many domains, they aren't interpretable in familiar terms and their ability to capture the full range of meanings expressible in language is not yet well understood.

In this dissertation, I argue that neural network models are capable of learning to represent and reason with the meanings of sentences to a substantial extent. First, I use entailment experiments over artificial languages to show that existing models can learn to reason logically over clean language-like data. I then present a large new corpus of entailments in English and use experiments on that corpus to show that these abilities extend to natural language as well. Finally, I introduce a new model that uses the semantic principle of compositionality to more efficiently and more effectively learn language from large volumes of data.

Acknowledgments

It was a great privilege to work closely with both Chris Manning and Chris Potts as advisors. I aspire to be even half as kind, reflective, and effective in my work as they've been, and I aspire to be even half as bold as they've encouraged me to be. I couldn't have asked for a better committee, either: Without Percy Liang's research as an example, I would have found it hard to believe that work on machine learning for broad-domain natural language understanding could pay off. Without Thomas Icard's advice, I would have had a much harder time forming some of the basic questions of this dissertation, and an even harder time addressing them.

I'm also grateful to Chris Manning, Dan Jurafsky, and Percy Liang for organizing the Stanford NLP Group, and to all of my colleagues in that group. It was my main academic community for most of my time at Stanford, and it was a lively, collaborative, and generally well-functioning setting in which to work.

I spent the summers during my PhD interning at Google, and much of my education in machine learning took place there. I'm grateful to Yuli Gao for taking a chance on me as a novice NLP researcher my first year, I'm grateful to Georg Heigold for taking a chance on me as a novice neural networks researcher my second year, I'm grateful to Bill MacCartney for being a supportive mentor during my foray into semantic parsing my third year and for doing the work on applied natural logic that laid the foundation for much of this dissertation, and I'm grateful to Oriol Vinyals for helping me get my bearings in modern deep learning my third year and for inviting me to come back to continue doing so my fourth year. In addition, I'm grateful to Samy Bengio for being a reliable and patient source of both humor and machine learning advice during all four years, and to Jakob Uszkoreit and Mattheiu Devin for going beyond the call of duty in a variety of ways to help me succeed there.

I would have had a lot less fun and been a lot less productive without my many coauthors: Harshit Chopra, Ben Lokshin, Marie de Marneffe, Miram Connor, Natalia Silveira, Tim Dozat, John Bauer, Gabor Angeli, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. I'm also happy to have gotten to work with Kai Sheng Tai, Neha Nayak, Kelvin Guu, and Jeffrey Pennington, even though our projects together never made it to print.

Beth Levin managed a fantastic graduate program and, with help from Dan Jurafsky, guided me through the many trials of the academic job hunt. Paul Kiparsky and Arto Anttila were excellent mentors during my stint as an aspiring phonologist, and I would have happily continued working with them had Richard Socher and Chris Potts not compelled me away with exciting results and even more exciting opportunities in the line of research that led to this dissertation. Everyone in the department office deserves credit for making sure that what sleep I lost was lost over academic problems and not administrative ones. Jason Riggle, Karen Livescu, and John Goldsmith at the University of Chicago first sparked and nurtured my interest in computational linguistics, and for that, I owe them a great deal.

Nothing that I've done in my last year at Stanford would be possible without the creativity and wit of the many Mechanical Turk workers—many anonymous—who participated in the creation of SNLI (Chapter 5).

The Donna Schweers and Thomas Geiser Stanford Interdisciplinary Graduate Fellowship kept me fed and sent me to conferences while I conducted much of the research presented in this dissertation.

Thank you dear reader for venturing this far into the acknowledgments. If your fortitude carries you through the rest of this dissertation, I hope you find that it was worth your time.

Personal support Graduate school would have been quite a lot more stressful without the many people at Stanford and Google who kept me sane with kibitzing over lunch, coffee, beer, or climbing: especially James, Janneke, Dasha, Ed, Roey, Lelia, Prerna, Bonnie, and Tim at Stanford, and my Google intern comrades Karl, Te, Yonatan, Nal, Luke, Arvind, Ali, Laurent, and Shane.

Without Pinhole Coffee and Caltrain, I would have gotten far less reading done, and I'm grateful to everyone who kept both running.

I'm grateful that Sam, Max, and Elly stayed close to me through graduate school despite my increasing inability to think or talk about anything other than research, and that my family for put up with me even though they didn't have much of a choice. I'm also grateful to my almost-family in Japan for sending encouragement from afar.

Saving the best for last: Thanks for everything, Dilini.

For E.F., in memoriam.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation introduces natural language inference as a task for the development and evaluation of neural network techniques for sentence understanding in natural language processing, and uses that task both to advance our understanding of existing models and as a problem on which to develop new models.

Neural network models have recently become the most effective tools for a range of hard applied natural language processing problems, including translation (Luong et al. 2015b), sentiment analysis (Socher et al. 2011b), and text generation (Wen et al. 2015). These models succeed in large part because they can learn and use their own continuous numeric representational systems for sentence meaning. However, their representations need not correspond in any interpretable way with the logic-based representations typically used in linguistic semantics. These models' successes in learning to solve semantically difficult problems signal that they are a potentially valuable object of study for semantics, and drawing insights from semantics to improve these models could yield substantial progress across applied language understanding tasks. This dissertation pursues these goals.

In order to study the ability of neural network models to develop good semantic representations, it is first necessary to choose a concrete task that instantiates that ability. I argue that the task of NATURAL LANGUAGE INFERENCE—also called RECOGNIZING TEXTUAL ENTAILMENT—is ideal for this purpose. In this setting, a machine learning system is adequate if it can learn representations for sentences that allow it to judge whether any given sentence contradicts or logically follows from any other given sentence. In a typical instance, a model would be asked to label the pair (1.1) as an entailment, rather than as a contradiction or a logically independent example.

(1.1) PREMISE: a young man without protective gear is on a bike jumping over piles of sand
HYPOTHESIS: a cyclist with no helmet is navigating obstacles with his bicycle

Since the task is framed as a simple decision problem over sentence pairs, it fits easily with standard machine learning methods. However, a model can only fully succeed on this task if it grapples with the full complexity of compositional semantics, including quantification, coreference, pragmatic inference, and many other phenomena. My experimental work on neural networks is centered on this task.

There are three excellent reasons to pursue the development of high-quality neural network models for language understanding:

- Improving upon models that already constitute the state of the art in natural language processing is a straightforward way to produce practical technological advances.

- Neural networks represent a uniquely clear instance of a naïve model class that incorporates little prior knowledge of grammar but that can nonetheless learn to use and understand language effectively in some settings. Better understanding what makes this possible can tell us a great deal about how language is learned and represented in humans.

- Attempting to build and understand models that can use language in a general sense will force us—productively—to integrate the symbolic theories of language understanding that best capture logical reasoning and complex compositionality with distributional approaches that best capture similarity, analogy, and heuristic inference.

This dissertation makes the following concrete contributions:

- In this chapter and throughout the dissertation, I introduce and motivate the use of natural language inference experiments as a highly effective means of evaluating representation learning systems for natural language understanding.
- In Chapter 3, I use experiments on artificial data and heavily structured natural language data to show that existing tree-structured neural networks can efficiently learn to handle key building blocks of natural language inference—memorizing and deriving lexical entailments, learning to represent recursive functions, and modeling the ways that semantic composition impacts entailment—without the need for special architectures or training procedures.
- In Chapter 4, I use additional experiments on artificial data to show that sequence-based recurrent neural networks—the most widely used style of neural network architecture for sentence understanding—are substantially less effective than tree-structured models at learning these same behaviors.
- In Chapter 5, I demonstrate the need for a large human-annotated dataset for NLI, and go on to present SNLI, the largest such dataset by almost two orders of magnitude.
- In Chapter 6, I demonstrate that SNLI makes it newly possible for neural network models to match the performance of conventional symbolic models on NLI.
- in Chapter 7, I use SNLI as a testing ground to develop SPINN, a new neural network architecture for sentence understanding, which incorporates the strengths of both tree-structured and recurrent models and yields a new state of the art.

This chapter lays out the key issues and motivations of this dissertation. Section 1.1 discusses the challenges of setting criteria for success in work on sentence meaning and presents the primary formal approaches to sentence understanding. Section 1.2 then discusses how sentence meaning is treated in applied work on computational semantics. Finally, Section 1.3 lays out the structure of the rest of this dissertation in more detail.

1.1 Defining sentence meaning

This dissertation studies neural networks as a means of extracting representations of meaning from the texts of sentences. In order to measure their ability to do this, it is necessary to have a clear picture of what success looks like: in this case, what constitutes an effective representation of sentence meaning. In this section, I briefly discuss the issue of what constitutes *sentence meaning* under two different views of semantics and go on to motivate my use of a RELATIONAL view of meaning. In this discussion, I focus primarily on literal meaning, before going on to briefly introduce issues of pragmatic enrichment.

There is currently no single approach to the representation of meaning that is suitable for all purposes. The question of what constitutes meaning is largely a question of definition. What's worse, it is a question of definition that is shared across a diverse set of research communities spanning philosophy, psychology, linguistics, and computer science, each with its own goals, making the question nearly impossible to answer decisively.

In light of this, I attempt to make my pursuit of *meaning* in this dissertation more precise by following the advice of Lewis (1970):

> In order to say what a meaning is, we may first ask what a meaning does, and then find something that does that.

To study meaning, then, I pick out some of the things that meanings *do* and stipulate that any system for representing sentence meaning that can reproduce those behaviors has an adequate command of sentence meaning. This section introduces two broad families of views about sentence meaning which differ in what activities they would have meanings do. The TRUTH-CONDITIONAL view, which is most prominent in formal semantics within linguistics, defines the meaning of a sentence primarily as the subset of possible situations in which that sentence is true. The RELATIONAL view, which I focus on in this dissertation, defines the meaning of a sentence primarily as the set of logical relationships that it enters into with all other possible sentences. The two views each present ways of thinking of the same semantic phenomena, but they lead to very different conclusions about how to conduct the research goals that I pursue in this dissertation.

1.1.1 The truth-conditional view of meaning

The mainstream approach to formal semantics represents the meaning of a sentence primarily through its truth conditions (Lewis 1970, Montague 1973, Heim & Kratzer 1998, i.a.). Under this approach, the meaning of a sentence can be seen as a function that can be applied to some representation of a situation (i.e. the state of the world, the time, the identity of the speaker, etc.) to determine whether the sentence is true in that situation. The primary goal of semantic theory in this view is the production of an effective formal system that can predict and describe the truth conditions of novel sentences. This goal is typically pursued using two key tools:

- Model theory: Model theory formalizes the means by which a sentence meaning can be checked against a situation and verified by introducing the notion of rich set-theoretic models of the world.
- Logical forms: The functional representations of meanings in this approach are generally logical forms. These are expressions that can be evaluated alongside a model (and possibly additional contextual information) to yield a truth value ($\mathtt{T}, \mathtt{F}$). Logical forms are generally expressed using modal logic or a similar logic and assembled from constituent parts at interpretation time.

As an example of what a meaning looks like in this tradition, Montague (1973) transcribes one possible reading of the sentence (1.2) as (1.3). I use slightly updated notation.

(1.2) A woman loves every man.

(1.3) $\forall x[\text{man}'(x) \rightarrow \exists y[\text{woman}'(y) \wedge \text{love}'(y, x)]]$

A model, in this case, would correspond to an assignment of members to the sets man$'$, woman$'$, and love$'$. If the model makes this assignment in such a way that the logical condition above is satisfied, then the sentence is true.

This approach to meaning captures the reasonable intuition that, if one knows the meaning of a sentence, then one knows what the world would have to be like for the sentence to be true. The particular choice of logical forms and model theory is a straightforward way to formalize a semantics that takes this view, but it is not the only possible way. There are many possible ways of formally representing the state of the world, and many possible ways of expressing functions over those representations, and no fundamental obstacles prevent neural network models from filling these roles in the construction of a truth-conditional semantics.

Interpretation in truth-conditional semantics

In the standard formal approach to truth-conditional semantics, the meaning (logical form) for a sentence is constructed from the sentence itself incrementally following the PRINCIPLE OF COMPOSITIONALITY. The principle, attributed to Gottlob Frege though never stated explicitly in his work, is summarized as follows by Partee (1984).

> The meaning of an expression is a function of the meanings of its parts and the way they are syntactically combined.

The parts in this statement are the smallest meaningful units of language: words with no decomposable parts (e.g. *dog*, *France*), morphemes (meaningful word parts, e.g. *un-*, *afford*, *-able*), and multi-word idioms (e.g. *kick the bucket*).

The process of interpretation in this view of language understanding is generally broken into three components: lexical semantics, compositional semantics, and pragmatics. Lexical semantics characterizes the meanings of the smallest meaningful units of language. Compositional semantics describes the formal procedures by which these minimal units are assembled together into sentence meanings, yielding logical forms like (1.3) above. Pragmatics describes the systematic reasoning processes by which listeners infer aspects of sentence meaning that aren't supported by the literal content of a sentence.

Much of the work that goes into specifying a full truth-conditional semantics involves constructing complex functional forms for words (and other basic units) within the lexical semantics such that those words can combine with other words in a way that yields correct sentence meanings. The word *everyone*, for example, might be represented by a function over predicates which selects those predicates that are true in any case where their argument is a person. Using a simplified lambda notation, that could be expressed as in (1.4).

(1.4) $[\![everyone]\!] = \lambda P.\forall x.\text{person}'(x) \rightarrow P(x)$

This task is quite challenging, since it is necessary to assign meanings to linguistic forms such that each meaning makes the precisely correct contribution to the meaning of every possible sentence it could be used in, without adding extraneous information, leaving out necessary information, or creating a type mismatch that would prevent the resulting sentence meaning from being checked against a model. A complete truth-conditional semantics has not yet been completed for any language.

1.1.2 The relational view of meaning

In this dissertation, I focus on an alternative framing of sentence meaning that selects inferential relationships, rather than truth conditions, as the primary element of meaning. In this view, rather than evaluating the truth of a sentence–situation pair, the semantics evaluates the relation holding between a sentence–sentence pair.

This framing is based in the INFERENTIALIST or PROOF-THEORETIC (in the sense of Hallnäs & Schroeder-Heister 1990) approach to formal semantics, which follows a substantial tradition in logic (Prawitz 1965, Dummett 1991, Francez et al. 2010). In natural language semantics, this tradition is instantiated most clearly in work on NATURAL LOGIC (Sánchez-Valencia 1991, van Benthem 2008, MacCartney & Manning 2009, Icard III 2012, Icard III & Moss 2013b), the formal characterization of a sound subset of intuitive human reasoning. Contemporary natural logic presents a rich formal system that is capable of determining, for example, that the relationship between (1.5) and (1.6) is that of ENTAILMENT—that is, whenever the first is true, the second must also be true—rather than CONTRADICTION or some other relation.

(1.5) Jimmy Dean refused to move without blue jeans.

(1.6) James Dean didn't dance without pants.

This framing surfaces in natural language processing as the task of NATURAL LANGUAGE INFERENCE (NLI; MacCartney 2009) or RECOGNIZING TEXTUAL ENTAILMENT (RTE; Dagan et al. 2006), in which computational models are evaluated on their ability to judge the relationships between pairs of novel sentences. While NLI is relatively immature as a research area within applied NLP, it has been used both as a proving ground for novel approaches to computational semantics and as a component in the design of larger NLP systems.

Natural logics use relatively simple representations for sentence meaning: they perform inference over the raw texts of sentences augmented with their parse trees. Using these surface forms as representations of meaning complicates the operation of the logic, but removes the need for a more sophisticated interpretation mechanism like those typical of truth-conditional semantics.

While natural logics operate over surface form, the choice of a relational view of meaning in general does not force one to choose this kind of superficial representational system. Neural network models for relational semantics like those presented in this dissertation cannot operate directly on text. Instead, they incorporate a component that translates sentences into DISTRIBUTED REPRESENTATIONS, as discussed below.

1.1.3 Choosing the relational view

Relational semantics and truth-conditional semantics both must account for many of the same natural language phenomena, including lexical and syntactic ambiguity, coreference, quantification, and many others. However, the two approaches highlight different aspects of language understanding, and a formal system that is adequate for one may not be adequate for the other. Even if a relational system can successfully recognize valid inferences, it may not necessarily have explicit access to any kind of world model that would allow it to evaluate the truth of a single statement. On the other hand, even if a truth-conditional system can successfully judge the truth of novel sentences in novel situations, it may not necessarily use representations that allow it to reason about inferences, at least not without the prohibitively expensive operation of enumerating all possible situations and models.

In this dissertation, I pursue the relational view. I build and evaluate neural network models for natural language inference as a means of evaluating their ability to interpret sentence meaning. This is because the relational view makes it possible to precisely implement semantic interpretation without the need for a model or any other explicit representation of situations in the outside world. A system for NLI thus needs to be able to handle only one type of input—the sentence—rather than both sentences and situations as in the truth-conditional view. This use of a single data type makes the interpretation of experimental results fairly straightforward. If a model succeeds at performing inference, it thereby demonstrates that it can effectively represent sentence meaning. If a model fails to perform inference, it has few other possible points of failure, suggesting that it lacks an effective representational system for sentence meaning.

While the neural network models that I build do not draw directly on techniques from natural logic, I do use it as a point of reference. In Chapters 3 and 4, testing models on their ability to reproduce the behavior of natural logic on aspects of reasoning that natural logic captures well.

1.1.4 Pragmatics and sentence understanding

Pragmatic inference is the process by which language users systematically apply information about social convention and conversational context to infer aspects of speaker intention that are not reflected in literal sentence meaning. Pragmatic reasoning is responsible, for example, for the inference that S2 below does not know where in Europe Bob is.

(1.7) a. **S1:** What country is Bob in right now?

b. **S2:** Bob is somewhere in Europe.

Pragmatic inference plays a substantial role in almost any instance of human language understanding, and it is relevant to any reasonable view of sentence meaning. In the context of applied natural language inference, it has been studied in work like that of de Marneffe et al. (2012), but it is not a major focus of current research. It is not highlighted in any of the evaluation tasks that I use in this dissertation, and I choose leave it as a background issue in much of what follows.

1.1.5 The role of representations in semantic theory

Semantics is, in large part, the study of the correspondence between linguistic form and meaning. Even though much of the body of semantic theory is presented from a formal and truth-conditional perspective, any empirically sound result about that correspondence is a contribution to semantics, no matter what representation or framing it is expressed under. In this section, I argue that both results about inferences and supervised learning results can inform semantic theory.

Inference results are semantic results

Claims about what inferences a sentence does or does not support are also claims about the truth conditions of that sentence. If some sentence A entails another sentence B, then the truth conditions of sentence A must be strictly less restrictive than the truth conditions of B: any situation that validates A must validate B. If A contradicts B, then their truth conditions must be disjoint: no possible situation can satisfy both.

Conversely, claims about the truth conditions of sentences are also claims about entailments. If the truth conditions for two sentences are such that the situations that validate A are a subset of those that validate B, than A entails B. Similar rules justify predictions of contradiction and other semantic relations. Although much of semantic theory is built in truth-conditional terms, it is often straightforward to recast existing claims in terms of inference. For example, attempts to specify the behavior of generics like *dogs* in (1.8) can be framed as attempts to predict the circumstances under which generic sentences entail related sentences like the simplified examples shown in (1.9).

(1.8) Dogs bark.

(1.9) a. All dogs bark.
b. 51% of dogs bark.
c. The stereotypical dog barks.

Similarly, attempts to predict the behavior of factive verbs like *forget* can be framed as questions of the entailments of full sentences, for example whether (1.10) or its negation (1.11) entails the simpler form (1.12).

(1.10) John forgot to eat breakfast.

(1.11) John did not forget to eat breakfast.

(1.12) John ate breakfast.

Machine learning results are semantic results

Much of linguistic theory takes the form of claims about what aspects of language are universal, and there is a long history of debate about the extent to which discovered universal properties reflect specific innate properties of human cognition. Machine learning results can provide evidence on this point: any aspect of language that a low-bias machine learning system (one that has no specific prior knowledge of language) can learn to reproduce from a type and amount of data that human children also have access to is an aspect of language that need not be innate. In practice, evidence of this kind is only indirect, since large-scale machine learning experiments like the ones in this dissertation never come especially close to reproducing the conditions of human language learning. However, this evidence can still be informative, especially in settings where it is possible to argue that the machine learning system is learning in strictly less advantageous circumstances, or from strictly less data, than the human learner.

1.2 Computational semantics: Building systems

In this section, I briefly outline the approaches to sentence meaning that are seen most often in applied computational semantics research within NLP. I focus particularly on distributed representations, which I use in this dissertation, and which do not have a close parallel with any approach used in formal semantics.

1.2.1 Truth-conditional semantics

Logical form-based representations of meaning, and their corresponding interpretation strategies, have been under study in computational settings for years (Blackburn & Bos 2005, Van Eijck & Unger 2010, i.a.). The most application-oriented line of work to emerge from this is that of semantic parsing for question answering (Woods et al. 1972, Zettlemoyer & Collins 2005, Liang et al. 2013), in which questions are translated to logical forms that can be checked against a knowledge base (which is effectively a simple world model) to yield answers. Systems based on logical forms are a natural fit for this kind of question answering, and are the best available systems for the task, but their performance is still far from perfect except on heavily constrained task domains due to the need for rich lexical representations that are difficult to acquire without expensive expert annotation.

1.2.2 Natural logic

The task of NLI has been studied relatively thoroughly within NLP and a range of methods for it have been proposed, usually applying machine learning to some combination of simple matching features and the outputs of local symbolic inferences. However, the history of formal approaches to applied NLI like natural logic is relatively brief. MacCartney (2009) finds that a direct implementation of natural logic using existing resources is relatively weak, failing to outperform simpler baselines on its own. MacCartney (2009) and Watanabe et al. (2012) both show that hybrid systems that also incorporate machine learning components can perform somewhat better.

1.2.3 Distributed representations

Artificial neural networks are built around distributed representations. Distributed representations for natural language (sometimes also called EMBEDDINGS) encode the

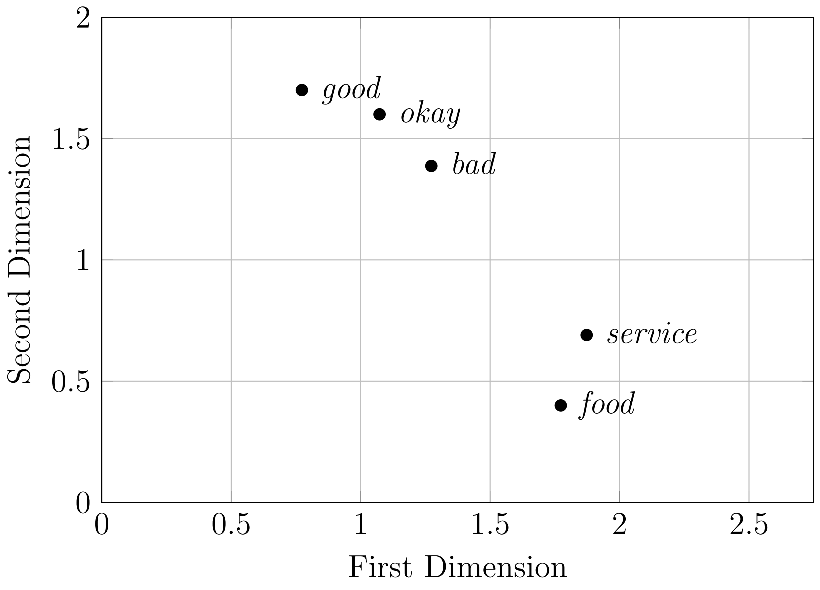


Figure 1.1: An example 2D Cartesian embedding space with the embeddings for a few words.

meaning of a piece of text not symbolically, but rather as a dense real-valued vector in a space of tens, hundreds, or thousands of dimensions. No individual dimension of this vector space is meant to encode any specific semantic feature, but all semantic information is instead distributed throughout the dimensions, in a rough analogy to the way information is represented in the brain. Generally, all possible natural language expressions will be represented as vectors of the same dimension in the same vector space, allowing for arbitrary pairwise comparisons between them. In particular, proximity in this vector space tends to reflect semantic similarity, and, at least within local regions, directions in this vector space tend to capture particular aspects of semantic variation (Mikolov et al. 2013b). This ability to integrate nuanced and continuously-variable measures of similarity directly into the representational system is tremendously helpful in enabling models based on distributed representations to generalize intelligently to unseen phenomena. Figure 1.1 shows an example of words in a simple two-dimensional embedding space.

Most successful applications of distributed representations to natural language sentences have relied extensively on machine learning, and for good reason. There is not currently any effective non-learning-based framework that is reliably able to condense sentences into fixed length representations without either sacrificing necessary information or expanding the size of those representations to an impractical degree. In contrast, machine learning-based methods, especially those involving artificial neural networks, have so far proven effective at negotiating this trade-off.

Distributed representations for meaning stem from a line of research into meaning representation that is historically aligned more with natural language processing than with either of the two views of semantics discussed above. As such, their development has not been directly motivated by either view of what meanings should do. Instead, a range of different practical goals have informed the development of a corresponding variety of different systems for distributed representation. However, there is no fundamental obstacle to using distributed representations within either truth-conditional or relational semantics, and in this dissertation I model relational semantics using these representations.

Word vectors and distributional semantics

Much of the research that has been conducted on distributed representations for text falls within the DISTRIBUTIONAL semantics paradigm. The aim of distributional semantics is to discover information about the meanings of words (and occasionally larger structures) using only distributional information, such as the words that a given word cooccurs with in a sentence, the types of syntactic configurations it appears in, or the documents it appears in. This often takes the form of algorithms for constructing distributed[1] representations for those objects. Baroni et al. (2014) provide a thorough history of this literature.

Distributional semantics methods are generally UNSUPERVISED LEARNING methods: they learn to assign vectors to words in a way that allows contextual information

[1] *Distributed* and *distributional* are distinct terms describing potentially similar objects. Distributed representations are dense numeric representations (vectors or tensors) of information of any kind. Distributional semantics is the approach to representing the meanings of words or phrases as collections of statistical information about the distribution of those phrases across documents and contexts. Distributional semantic representations often take the form of distributed representations generated by some function applied to a set of cooccurrence statistics.

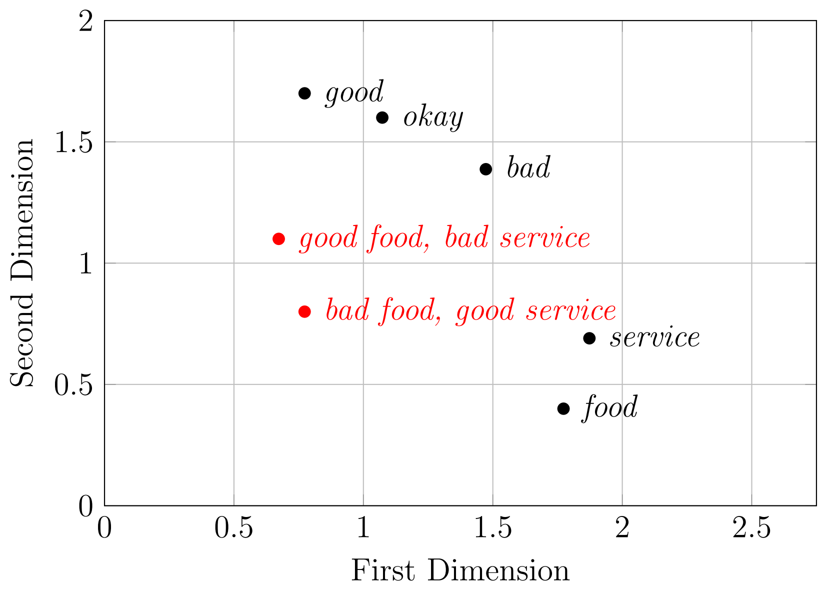


Figure 1.2: An example 2D Cartesian embedding space with some sentence fragment encodings shown.

of some kind to be recovered from those vectors, rather than learning to specifically tune those vectors to provide useful semantic signals for any specific applied task. Nonetheless, much of the extant work on distributional semantics aims to produce representations that are broadly useful as inputs to task-specific NLP models. They have largely succeeded at this aim, as distributed representations generated from large text corpora using methods like SkipGram (Mikolov et al. 2013a), CBOW (ibid), and GloVe (Pennington et al. 2014) are ubiquitous in modern NLP.

Sentence vectors and sentence encoding

Interpretation in the context of distributed representations is the process of translating the text of a sentence into a vector representation of that sentence, called a SENTENCE ENCODING or SENTENCE EMBEDDING. While word representations from distributional semantics approaches are commonly used as inputs to systems for sentence encoding, distributional statistics alone are not especially effective above the word level. This is because sentence-encoding systems need to be able to produce encodings for sentences which have never been uttered before, and for which there are thus no statistics available. Instead, sentence encoding is almost invariably done using neural network models which are trained to combine the distributed representations for the words (or word parts) in a sentence into a new distributed representation for that sentence. Figure 1.2 shows an example of sentence encodings for two short sentence fragments and the word embeddings that they are built from.

Instead of attempting to predict or reconstruct distributional statistics, these neural network sentence encoders are generally trained to produce vectors that solve some specific language understanding task, such as sentiment evaluation, translation, or paraphrase identification. They are almost invariably trained in a SUPERVISED LEARNING setting, wherein the model is encouraged to match the known correct output for each input from some human-annotated training corpus.

1.3 This dissertation

Machine learning experiments on natural language inference data constitute the primary mode of research in this dissertation. My goal is not to describe the properties of ideal or formally elegant distributed representation models for sentence meaning, as has been explored in research like that of Coecke et al. (2011), but it is rather to understand and improve neural network based sentence-encoding models that have been empirically shown to solve hard problems.

After Chapter 2 introduces the key technical machinery used in this dissertation, Chapter 3 presents four experiments on artificial data derived from implementations of natural logic and on heavily structured natural language data. These experiments establish a basic picture of the learning ability of the tree-structured models that I study. I find that existing tree-structured neural network models can learn lexica of word embeddings that support natural language inference, learn to use a compositional grammar to interpret novel structures, and learn to encode the complex functional meanings of quantifiers. These models can, for example, conclude that *all reptiles swim* contradicts *some turtles do not move.* Chapter 4 extends these experiments to show that tree-structured models are substantially more effective than simpler sequence-based models at reasoning over compositional structures, and discusses the trade-offs between the two approaches.

Next, Chapter 5 presents the data necessary to extend these artificial language results to real English. While there are preexisting public corpora of natural language inference data available, none of them contains more than a few thousand examples. Since neural network models make up for their lack of prior knowledge by demanding vast amounts of training data, they cannot learn well in this context. After presenting an experiment that demonstrates the seriousness of this problem, I present the Stanford Natural Language Inference (SNLI) corpus, which I collected. It consists of 570,000 sentence pairs constructed and labeled for natural language inference by human annotators through a crowdsourcing task.

In order to collect labeled sentence pairs in a way that is intuitive and consistent across individuals, I also develop a novel approach to NLI data collection based around scene descriptions. In it, workers are given a description of a scene and asked first to re-describe the events in that scene in a new sentence (yielding an entailment between the initial description and the newly created one), then to describe the events of a similar but different scene (yielding a contradiction), and then to write a third description that is possibly, but not definitely, true of the original scene (yielding a semantically independent example). A subsequent validation study shows that the collected sentence pairs are generally understandable and that their relation labels can be reliably reproduced by other annotators.

Next, Chapter 6 presents a set experiments that establish performance baselines for SNLI with a variety of model types. I find that while some neural network models are competitive with the state of the art in conventional non-neural network NLI models, neural network models which incorporate explicit syntactic parse information into their representations—and which have been shown to perform at the state of the art elsewhere—cannot scale to the size and difficulty of SNLI.

Following this, Chapter 7 introduces SPINN, a novel model architecture for sentence encoding that makes it possible to use syntactic structure in large-scale natural language learning tasks like SNLI. This architecture integrates the computations of a tree structured sentence encoding into the structure of a natural language parser, yielding dramatic gains in training and test speed, the ability to operate on unparsed data, and a new state of the art on sentence encoding for SNLI.

Finally, Chapter 8 concludes and lays out promising directions for future work.

Chapter 2

Background and technical foundations

This chapter introduces the key technical concepts of this dissertation, and provides references to relevant prior work. Section 2.1 introduces natural language inference in more detail and describes the machinery of natural logic. Section 2.2 introduces the principles of artificial neural networks, and Section 2.3 discusses their use in sentence encoding. Finally, 2.4 discusses some additional relevant prior work.

2.1 Natural language inference and natural logic

The experiments in this dissertation are built around the task of natural language inference, in which the goal is to determine the core inferential relationship between two sentences. The semantic concepts of entailment and contradiction are central to all aspects of natural language meaning (Katz 1972), from the lexicon to the content of entire texts. Thus, NLI—characterizing and using these relations in computational systems—is essential in tasks ranging from information retrieval to semantic parsing to common-sense reasoning (Fyodorov et al. 2000, Condoravdi et al. 2003, Bos & Markert 2005, Dagan et al. 2006, MacCartney & Manning 2009).

In recent years, NLI has become an important testing ground for approaches employing distributed word and phrase representations. Distributed representations excel at capturing relations based in similarity, and have proven effective at modeling simple dimensions of meaning like evaluative sentiment (e.g. Socher et al. 2013), but it is less clear from existing literature that they can be trained to support the full range of logical and common-sense inferences required for NLI (Weston et al. 2015, 2016, and Chapter 3). In a SemEval 2014 task aimed at evaluating distributed representations for NLI, the best-performing systems relied heavily on additional features and reasoning capabilities (Marelli et al. 2014a).

Much of the theoretical work on this task involves NATURAL LOGIC, a family of formal systems that define rules of inference between natural language words, phrases, and sentences without the need of intermediate representations in an artificial logical language. This dissertation uses the natural logic developed by MacCartney & Manning (2009) as a formal point of comparison. This logic defines seven core relations of synonymy, entailment, contradiction, and mutual consistency, as summarized in Table 2.1, and it provides rules of semantic combination for projecting these relations from the lexicon up to complex phrases.

Several applied models also draw methods from natural logic. MacCartney (2009) himself presents an implementation of his natural logic that specifies a straightforward pipeline of tools to produce the alignment and lexical relation information that the core logic requires for any given example. MacCartney finds mixed results with this implementation, with the model tending to favor precision when identifying entailments and contradictions at the expense of low recall. Watanabe et al. (2012) improve upon MacCartney's implementation of the logic by treating some portion of the alignment and lexical relation information as latent. Angeli & Manning (2014) use a simple implementation of MacCartney's logic in a model that is designed to search efficiently over a large space of possible premises to judge the truth of a single target hypothesis.

Name	**Symbol**	**Set-theoretic definition**	**Example**
entailment	$x \sqsubset y$	$x \subset y$	*turtle* $\sqsubset$ *reptile*
reverse entailment	$x \sqsupset y$	$x \supset y$	*reptile* $\sqsupset$ *turtle*
equivalence	$x \equiv y$	$x = y$	*couch* $\equiv$ *sofa*
alternation	$x \mid y$	$x \cap y = \emptyset \wedge x \cup y \neq \mathcal{D}$	*turtle* $\mid$ *warthog*
negation	$x \wedge y$	$x \cap y = \emptyset \wedge x \cup y = \mathcal{D}$	*able* $\wedge$ *unable*
cover	$x \smile y$	$x \cap y \neq \emptyset \wedge x \cup y = \mathcal{D}$	*animal* $\smile$ *non-turtle*
independence	$x \# y$	(else)	*turtle* $\#$ *pet*

Table 2.1: The seven natural logic relations of MacCartney & Manning (2009). $\mathcal{D}$ is the universe of possible objects of the same type as those being compared. The relations are mutually exclusive and the relation # applies whenever none of the other six do.

2.1.1 Projectivity

Much of the complexity of natural logic is in the piece of the logic responsible for determining how word–word relations are PROJECTED onto corresponding sentence–sentence relations. Determining, for example, that the relation in (2.1) is reversed when placed in the sentential context shown in (2.2), and that it becomes the minimally informative # relation when its two sides are placed in the differing contexts shown in (2.3).

(2.1) cat $\sqsubset$ animal

(2.2) Stimpy is not a cat $\sqsupset$ Stimpy is not an animal

(2.3) Stimpy is not a cat # Stimpy is an animal

The machinery responsible for this inference is that of PROJECTIVITY, an extension of the MONOTONICITY CALCULUS of earlier work on natural logic (Sánchez-Valencia 1991, van Benthem 2008). I refer the reader to MacCartney (2009) for a full treatment of this aspect of natural logic.

2.1.2 Soundness and NLI

The formal properties and inferential strength of contemporary natural logic are now well understood (Icard III & Moss 2013a,b), and it is known to be sound. If it derives a relation between two expressions of natural language, then that relation is correct under standard assumptions about natural language meaning. However, there is no guarantee that this logic will be able to derive a relation between any two comparable expressions, and in fact inference in the logic often halts with an output state that reflects uncertainty between several or even all of the seven possible relations.

This means that if some formal or computational system is to be able to accurately perform natural language inference across a maximally broad range of natural language expressions, that system will have to be able to fully reproduce the inferences of natural logic, but it will also need the power to make additional inferences as well. The structure of this dissertation mirrors that observation, first studying the ability of neural network sentence-encoding models to reproduce the behaviors of natural logic, then studying their ability to learn to do open domain inference by example.

2.1.3 Data for natural language inference

The quality of a machine learning model depends heavily on the quality and quantity of the data on which it is trained, and this dependence is especially strong for models like neural networks that incorporate little prior domain knowledge. As such, building effective machine learning systems for natural language inference requires access to large high-quality corpora of example inferences. This dissertation uses artificially generated corpora, preexisting human-annotated corpora, and a novel human-annotated corpus.

Artificial language data

One of the aims of this dissertation is to use artificial language learning results to make claims about the abilities of existing neural network models to learn to represent particular aspects of natural language meaning. Artificial language data provides two key advantages that make it ideal for studying representation learning systems like these. First, it is possible to create datasets from well-understood formal models, such that the data are guaranteed to be internally consistent and error-free, and such that the challenges posed by each experiment can be tightly constrained. In addition,

Corpus	**Examples**	**NL**	**Validated**
FraCaS (Cooper et al. 1996)	**346**	✓	✓
RTE 1–5 (Dagan et al. 2006, et seq.)	**7K**	✓	✓
SICK (Marelli et al. 2014b)	**10K**	✓	✓
SNLI (Bowman et al. 2015a, Ch. 5)	**570K**	✓	✓
DenotationGraph (Young et al. 2014)	**728K**	✗	✗
Entailment graphs (Levy et al. 2014)	**1,500K**	✗	✗
Paraphrase Database 2.0 (Pavlick et al. 2015)	**100,000K**	✗	✗

Table 2.2: A comparison of available NLI corpora. *Examples* indicates the number of labeled sentence pairs in the corpus. *NL* indicates whether a corpus contains pairs of full natural language sentences. *Validated* indicates whether the labels in a corpus were assigned or validated by human annotators.

it is possible to scale datasets to arbitrarily large sizes with minimal effort, making it possible to ensure that learning systems do not fail to model any aspect of a dataset solely for lack of evidence.

In Chapters 3 and 4 I train neural network models on data from artificial languages. To generate these datasets, I randomly generate individual expressions from these languages, and then label pairs of these expressions with a purpose-built Python implementation of MacCartney's natural logic.

Natural language data

Training neural network models to do NLI on real natural language requires the use of annotated corpora. While it is possible to automatically annotate inference data for simple artificial languages, this luxury is not available for natural language.

Table 2.2 surveys existing NLI corpora and compares them with the Stanford Natural Language Inference (SNLI) corpus, which is presented in this dissertation. Of the six existing datasets, three contain full natural language sentence pairs, but are too small to be effective in training neural networks, and three others are large enough to be effective, but are not structured as sentence pairs, and were generated automatically, yielding substantial amounts of biased error in their labels.

FraCaS (Cooper et al. 1996) is the earliest corpus for NLI to be widely available,

Section 1: Quantifiers		
No delegate finished the report.	CONTRADICTION	Some delegate finished the report on time.
Section 2: Plurals		
Either Smith, Jones or Anderson signed the contract.	NEUTRAL	Jones signed the contract.

Table 2.3: Examples of data from FraCaS.

and was created manually by experts in order to highlight a diverse range of inference patterns. As such, it is not entirely natural, but it is nonetheless is an excellent task for revealing the ability of models to do several kinds hard formal inference effectively. FraCaS contains 346 question and answer pairs which can be converted into an entailment form using a straightforward method from MacCartney (2009). It is broken down into sections covering semantic issues like quantification, anaphora, and propositional attitudes. There are three labels covering most of the data, YES (forward entailment), NO (contradiction), and UNK (independence, including reverse entailment). Table 2.3 shows a couple of examples. The small size makes it effectively useless as a training set, but its high quality and cleanly defined subsets make it a useful benchmark for testing, and I use it in this way in Section 7.4.3.

The Recognizing Textual Entailment datasets (RTE; Dagan et al. 2006, et seq.), produced in a series of competitions, comprise about seven thousand entailment examples, divided between two- and three-class classification. (Only the later datasets mark a contrast between contradiction and independence.) These problems are drawn at least in part from naturally occurring sentences, and tend to be fairly complex and to rely on a considerable amount of world knowledge. Table 2.4 shows some examples from RTE-3, a task from this series. The RTE tasks have served as useful benchmarks for machine learning models for NLI, but their small training sets limit their value for this purpose for neural network models. Chapter 6 briefly compares the performance of several non-neural models on RTE-3 with the performance of those same models on SNLI.

The Sentences Involving Compositional Knowledge dataset (SICK, released as

As leaders gather in Argentina ahead of this weekends regional talks, Hugo Chávez, Venezuelas populist president is using an energy windfall to win friends and promote his vision of 21st-century socialism.	ENTAILMENT	Hugo Chávez acts as Venezuela's president.
Mr. Fitzgerald revealed he was one of several top officials who told Mr. Libby in June 2003 that Valerie Plame, wife of the former ambassador Joseph Wilson, worked for the CIA.	NON-ENTAILMENT	Joseph Wilson worked for CIA.

Table 2.4: Examples of training data from RTE-3, adapted from MacCartney (2009).

The player is dunking the basketball into the net and a crowd is in background.	NEUTRAL	A man with a jersey is dunking the ball at a basketball game.
Two people are kickboxing and spectators are not watching.	CONTRADICTION	Two people are kickboxing and spectators are watching.

Table 2.5: Examples of training data from SICK.

SemEval 2014 Task 1; Marelli et al. 2014b) was the human-annotated corpus for NLI until the release of SNLI, at about ten thousand examples. It consists of entailment examples derived from image and video captions and labeled for three-class classification. The data collection process involved significant pruning and editing, and drew on only a few hundred source sentences, leaving the examples in the corpus somewhat artificially short and self-similar. However, the result is a set of examples with minimal reliance on idiom or world knowledge, and the only preexisting corpus large enough to even come close to supporting the training of a neural network-based model. A few examples are shown in Table 2.5.

The Denotation Graph entailment set (Young et al. 2014) contains millions of examples of entailments between sentences and artificially constructed short phrases, but it was constructed using fully automatic methods, and is noisy enough that it is

A man editing a black and white photo at a computer with a pencil in his ear.	ENTAILMENT	man sit
A grown lady is snuggling on the couch with a young girl and the lady has a frightened look.	NON-ENTAILMENT	man sit

Table 2.6: Examples of Denotation Graph data.

suitable only as a source of supplementary training data. A couple of typical examples are shown in Table 2.6.

Outside the domain of sentence-level entailment, Levy et al. (2014) introduce a large corpus of semiautomatically annotated entailment examples between subject–verb–object relation triples, and the second release of the Paraphrase Database (Pavlick et al. 2015) includes automatically generated entailment annotations over a large corpus of pairs of words and short phrases.

Chapter 5 briefly discusses the kinds of neural network learning work that can be done with existing data, and then presents a new corpus, SNLI, which is meant to be both large enough and of high enough quality to support training low-bias models like neural networks to do NLI.

2.2 Artificial neural networks

ARTIFICIAL NEURAL NETWORK models (also called *neural networks* or NNs) are machine learning models consisting of chains of composed highly-parametric functions called layers. Most applied machine learning systems that create or manipulate distributed representations either are neural networks themselves or have significant NN components. Neural network parameters can be initialized randomly and trained from data—with or without corresponding labels, depending on the specific model architecture—and they can learn to approximate potentially arbitrarily complex functions (Hornik et al. 1989).

Figure 2.1 illustrates a simple neural network model for a toy single word classification task that illustrates some simple key components that occur across many neural network models. Some key components of neural networks are discussed in the

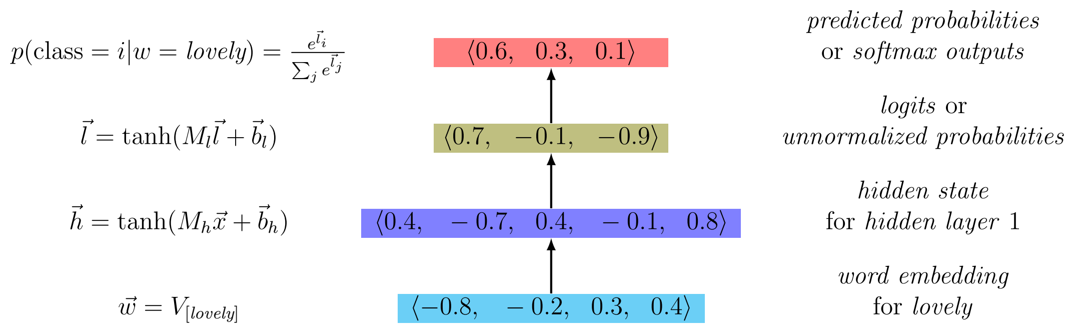

$p(\text{class} = i | w = \textit{lovely}) = \frac{e^{\vec{l}_i}}{\sum_j e^{\vec{l}_j}}$
⟨0.6, 0.3, 0.1⟩
predicted probabilities or *softmax outputs*
$\vec{l} = \tanh(M_l \vec{l} + \vec{b}_l)$
⟨0.7, −0.1, −0.9⟩
logits or *unnormalized probabilities*
$\vec{h} = \tanh(M_h \vec{x} + \vec{b}_h)$
⟨0.4, −0.7, 0.4, −0.1, 0.8⟩
hidden state for *hidden layer* 1
$\vec{w} = V_{[lovely]}$
⟨−0.8, −0.2, 0.3, 0.4⟩
word embedding for *lovely*

Figure 2.1: A simple example neural network designed to predict which of three classes (e.g. *positive*, *neutral*, *negative*) a word belongs to. The model has five sets of parameters: the word embedding matrix V, the matrix parameters of the two tanh layers M_h and M_l, and the bias parameters of the two tanh layers $\vec{b}_h$ and $\vec{b}_l$. When the model is being run, a word embedding vector w is first looked up from the appropriate row of V, then passed through one layer to yield the hidden state $\vec{h}$, then passed through a second layer to yield the logits (unnormalized probabilities) $\vec{l}$, then finally passed through the softmax function to produce a distribution over the three classes. The top two rows of this diagram and the parameters M_h and b_h together represent a *softmax layer*. The third row and the parameters M_l and b_l together represent a *hidden layer*.

caption and below, but a full presentation is far beyond the scope of this dissertation. For a thorough introduction, I refer the reader to the recent textbook by Goodfellow et al. (2016).

Neural network models use distributed representations of some fixed prespecified dimension as their sole means of representing information. Correspondingly, one of the major challenges in designing a neural network model architecture for some task is finding an appropriate way of transforming the task data into a fixed-size distributed form for input into the network. For some tasks, this is relatively straightforward: if the task involves images of a fixed size, for example, each pixel of each image can be treated as one dimension in a distributed representation. For text, though, there are two challenges: discrete symbols (generally words) must be transformed into continuous representations, and this transformation must be able to produce representations of a consistent dimension from sequences that vary in length. The former problem is almost universally solved by the technique of word embedding, but the latter problem—that of sentence encoding—remains an open issue in NLP research, and one that I investigate in this dissertation.

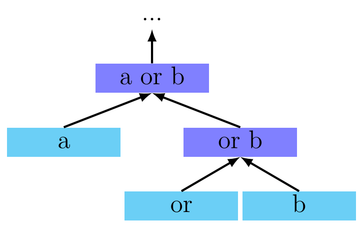
2.2.1 Word embeddings in neural networks

Nearly all neural network models in NLP store a vocabulary in which each word (or word part) is mapped to a vector of some preselected length. Whenever a word is used as an input to the NN, its embedding vector is looked up and used to represent it. In many cases these embedding vectors are created (or PRETRAINED) using distributional methods, as discussed in Section 1.2.3, but when embeddings are used within neural network models to solve a specific task, an additional approach becomes available: the full set of embeddings vectors V can be treated as a parameter of a neural network model, and can be initialized randomly at the start of model training and learned alongside the other parameters. In this dissertation I use pretrained word embeddings for most natural language experiments and train task-specific word embeddings for artificial language experiments (where distributional methods are less applicable) and elsewhere where noted.

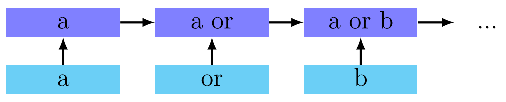
The supervised learning of task-specific representations of discrete symbols can be traced back at least to the birth of modern neural networks with the introduction of the backpropagation learning algorithm by Rumelhart et al. (1986), and forms a significant part of the recent successes of neural network models.

Key prior work Bengio et al. (2006) introduce the use of word embeddings within the supervised training of neural network models for NLP, using embeddings as inputs to a feedforward neural network language model and training them through backpropagation as part of the broader language model training procedure.

Collobert et al. (2011) provide an important demonstration of the power of learned vector representations in applied work that requires semantic representations, and in particular show that a single set of learned vector representations can suffice for a diverse set of tasks. They used partially labeled data, with labels from a number of different tasks, to learn a general-purpose set of vector representations for their vocabulary of English. Their model was structured as a set of neural networks with shared word representations, and was capable of outputting part of speech (POS)



(a) The architecture for tree-structured sentence models. Terminal nodes (light blue) are learned embeddings and non-terminal nodes (indigo) are NN, NTN, or TreeLSTM layers.



(b) The architecture for the sequence-based sentence models. Nodes in the lower row are learned embeddings and nodes in the upper row are RNN, LSTM, or GRU layers.

Figure 2.2: Examples of tree- and sequence-based neural network sentence-encoding models, instantiated for the simple sentence *a or b*.

tags, named entity tags, shallow parses, language model scores, and semantic role labels (SRL), all with performance at or near the state of the art for that time.

Countless subsequent studies on neural networks for natural language processing at the word, sentence, and document levels have gone on to build on this line of work.

2.3 Neural networks for sentence encoding

Individual neural network layers require fixed length inputs. Thus, if a neural network is to handle sentences of varying lengths, it needs to be accompanied by a component for converting these sentences into suitable fixed-length vectors. A simple such technique is to look up the embedding vectors for each of the words in a sequence and sum or average them, yielding a single vector of the same dimension. This strategy, often called CONTINUOUS BAG OF WORDS (CBOW) is effective in some simple tasks, but it is impossible for a network to recover any word order information from representations that are produced in this way.

Instead, most modern neural network models that operate over word sequences include a special learned neural network component called a SENTENCE ENCODER. This component has repeating parts that can be added or removed to fit the structure and size of the input sentence, and generally takes one of three basic forms:

- In RECURRENT or SEQUENCE-BASED networks (Figure 2.2b; Elman 1991, Sutskever et al. 2014), including long short-term memory networks (LSTMs; Hochreiter & Schmidhuber 1997) and gated recurrent unit networks (GRUs; Cho et al. 2014a), the input is fed into the network in sequential order (from left to right or vice versa for text) with the network updating a hidden state after each input is processed.
- In CONVOLUTIONAL networks (LeCun et al. 1998, Kalchbrenner et al. 2014, Zhang et al. 2015, Conneau et al. 2016), information from all parts of a sentence are processed in parallel using a set of filters that look at fixed-size subsequences of words.
- In RECURSIVE or TREE-STRUCTURED networks (Figure 2.2a; Goller & Küchler 1996, Socher et al. 2011a), the input is fed in according to a tree structure (generally produced by a parser), in which hidden representations are formed for increasingly large sub-spans of the input data following the principle of compositionality.

Of these, sequence-based models have been the most widely used in practice and tree-structured models have an independently compelling theoretical motivation through compositionality and a strong track record in their limited applications so far. Convolutional models have not been as widely used, and I will not be focusing on them in this dissertation. The references above provide further discussion.

While sentence encoders are very common in work on natural language understanding with neural networks, their roles can vary. In recent work on soft attention (see Section 2.3.5), a sentence encoder is used to encode a sentence, but some of its internal state information is used to directly supply information to downstream model components, shortcutting the encoder's explicit output representation. In addition, some models like that of Parikh et al. (2016) avoid sentence encoding entirely, and build representations of multiple-sentence inputs without ever constructing representations for the constituent sentences.

2.3.1 Recurrent neural networks

The simplest sequence-based model, the plain recurrent neural network, computes representations for sentences by applying the following function repeatedly, consuming input words $x^{(t)}$ one by one from left to right before producing an output $\vec{y}^{(T)}_{\text{RNN}}$ at the final step T.

$$\vec{y}^{(t)}_{\text{RNN}} = f(\mathbf{M}\begin{bmatrix}\vec{y}^{(t-1)}_{RNN}\\ \vec{x^{(t)}}\end{bmatrix} + \vec{b}) \tag{2.4}$$

Unlike a plain RNN, an LSTM has several hidden activation vectors at each time step with different purposes, two of which ($\vec{h}$ and $\vec{c}$) are transmitted across steps. The below equations define an LSTM, with $\odot$ designating element-wise (Hadamard) product.

$$\begin{bmatrix}\vec{i}\\ \vec{f}\\ \vec{o}\\ \vec{g}\end{bmatrix} = \begin{bmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{bmatrix}\left(W\begin{bmatrix}\vec{h}^{(t-1)}\\ \vec{x}^{(t)}\end{bmatrix} + \vec{b}\right) \tag{2.5}$$

$$\vec{c}^{(t)} = \vec{f} \odot \vec{c}^{(t-1)} + \vec{i} \odot \vec{g} \tag{2.6}$$

$$\vec{h}^{(t)} = \vec{o} \odot \vec{c}^{(t)} \tag{2.7}$$

The $\vec{c}$ states are meant to serve as a long-term memory and are not passed through full multiplicative neural network layers between steps, but are instead updated additively. The $\vec{c}$ states can thus reliably propagate a gradient signal from each time step to the step before it, which enables LSTMs to learn much more effectively than plain RNNs on long sequences. The $\vec{h}$ state serves as the primary output of the network, with the final state $\vec{h}^{(T)}$ acting as the input to a downstream task model.

The non-recurrent internal variables $\vec{i}$, $\vec{f}$, $\vec{o}$, and $\vec{g}$ are computed anew at each step and govern the flow of information in and out of $\vec{c}$ through soft continuous gating operations: the input gate $\vec{i}$ (in $(0, 1)$) indicates which elements of $\vec{c}$ are to be written to, the forget gate $\vec{f}$ (in $(0,1)$) indicates which elements of $\vec{c}$ are to be zeroed out, the output gate $\vec{o}$ (in $(0,1)$) indicates which elements of $\vec{c}$ are to be passed to $\vec{h}$ for use in the current step's computation, and $\vec{g}$ (in $(-1,1)$) supplies new values to be written to $\vec{c}$.

Both RNNs and LSTMs can be stacked (see, for instance, Sutskever et al. 2014) to create a more powerful network for a given sequence length. In this case, the hidden activation ($\vec{h}$ or $\vec{y}$) from each time step of one network is used as the input ($\vec{x}$) to a corresponding time step in another network, which is jointly trained but separately parameterized.

Key prior work Sequence based models have been used extensively in recent years as probabilistic language models (Mikolov et al. 2010, Jozefowicz et al. 2016), in which a sequence of words (generally representing a partial sentence) is fed into a network which, after each word, generates a latent representation which can be used to predict a distribution over possible next words.

Concurrently-authored papers by Sutskever et al. (2014) and Bahdanau et al. (2015) show that sequence-based NNs can also be used to encode the meanings of full sentences in a way that can be subsequently decoded in a different language to form a machine translation system. The latter work achieved state-of-the-art results, beating translation-specific machine learning systems that have been developed and tuned over many years, suggesting that distributed representations from neural networks are among the best tools available for the representation of natural language meaning in applied settings. A great deal of subsequent work has built on these successes.

Beyond these two core application areas, sequence-based models have been used for a wide range of other NLP tasks, including parsing (Dyer et al. 2015, Vinyals et al. 2015), text generation (Wen et al. 2015), and sentiment analysis (Tai et al. 2015) among many others.

2.3.2 Tree-structured neural networks

Tree-structured recursive neural networks (abbreviated as TreeRNNs here, referred to ambiguously as RNNs, RcNNs, or RecNNs in some earlier work) represent an alternative approach to sentence encoding. Unlike sequence-based networks, which compose meanings from left to right, TreeRNNs build on the observation that the composition of meaning in natural language follows a tree structure that closely resembles the syntactic structure of sentences (Lambek 1958, Lewis 1970, Montague 1973, Szabolcsi 2009).

TreeRNNs have a principled foundation in compositionality and have succeeded at tasks involving at least some aspect of natural language meaning, so they are a natural object of study in this dissertation. Artificial data experiments in Chapter 3 show that they can efficiently learn key aspects of NLI in controlled settings, and further such experiments Chapter 4 show that they are much more effective than plain sequential RNNs on data generated by a tree-structured grammar. Chapter 7 presents technical innovations that make it possible to apply TreeRNN-style compositionality to large-scale NLP learning problems like the one posed by SNLI.

In a TreeRNN, the composition function, a neural network layer, maps a pair of word or phrase vectors of length D to a single vector of length D, which can then be merged again with another word or phrase vector to represent a more complex phrase. Once the sentence-level vector representation—the vector corresponding to the root of the parse tree—has been derived in this way, it serves as a fixed-length input for some subsequent function.

The plain TreeRNN applies a classic NN layer as a composition function to the concatenated vectors for two children, as in (2.8). The TreeRNTN layer function adds an additional term, shown in (2.9), following the version of Chen et al. (2013).

$$\text{(2.8)} \qquad \vec{y}_{\text{TreeRNN}} = f(\mathbf{M} \begin{bmatrix} \vec{x}_{\text{left}} \\ \vec{x}_{\text{right}} \end{bmatrix} + \vec{b}\,)$$

$$\text{(2.9)} \qquad \vec{y}_{\text{TreeRNTN}} = \vec{y}_{\text{TreeRNN}} + f(\vec{x}_{\text{left}}^{T} \mathbf{T}^{[1\ldots n]} \vec{x}_{\text{right}})$$

Here, $\vec{x}_{\text{left}}$ and $\vec{x}_{\text{right}}$ are the column vector representations for the left and right children of the node, and $\vec{y}$ is the node's output. The TreeRNN concatenates them, multiplies them by an $n \times 2n$ matrix of learned weights, and adds a bias $\vec{b}$. The TreeRNTN adds a learned third-order tensor $\mathbf{T}$, dimension $n \times n \times n$, modeling multiplicative interactions between the child vectors. Both layer functions are wrapped with the elementwise nonlinearity $f(x) = \tanh(x)$.

An alternative, TreeLSTM composition layer function (Tai et al. 2015), generalizes the LSTM neural network layer to tree- rather than sequence-based inputs. It shares with the LSTM layer the idea of representing intermediate states as a pair of an active state representation $\vec{h}$ and a memory representation $\vec{c}$. I use the following formulation:

$$\begin{bmatrix} \vec{i} \\ \vec{f_l} \\ \vec{f_r} \\ \vec{o} \\ \vec{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W_{\text{composition}} \begin{bmatrix} \vec{h}_{\text{left}} \\ \vec{h}_{\text{right}} \end{bmatrix} + \vec{b}_{\text{composition}} \right) \tag{2.10}$$

$$\vec{c} = \vec{f_l} \odot \vec{c}_{\text{left}} + \vec{f_r} \odot \vec{c}_{\text{right}} + \vec{i} \odot \vec{g} \tag{2.11}$$

$$\vec{h} = \vec{o} \odot \vec{c} \tag{2.12}$$

where σ is the sigmoid activation function, $\odot$ is the element-wise product, and the pairs $\langle \vec{h}_{\text{left}}, \vec{c}_{\text{left}} \rangle$ and $\langle \vec{h}_{\text{right}}, \vec{c}_{\text{right}} \rangle$ come from the two children of the node being computed. The result of this function, the pair $\langle \vec{h}, \vec{c} \rangle$, is the representation of the current tree node. As in the LSTM RNN, though, the $\vec{c}$ portion of this state is meant to be isolated from the rest of the network, and only the $\vec{h}$ portion of this state is used to represent a phrase or sentence in the downstream task model. The intuition behind the five gates $\vec{i}$, $\vec{f_l}$, $\vec{f_r}$, $\vec{o}$, and $\vec{g}$ are the same as those behind the four gates of the LSTM described in Section 2.3.1, with two forget gates instead of one to correspond to the two different inputs. Each vector-valued variable listed is of dimension D.

There has been little work done to date on the precise types of syntactic information that TreeRNNs find most valuable in large-scale language learning experiments (and, correspondingly, on what approaches to parsing will yield the best TreeRNN performance). This remains a valuable direction for future work.

Key prior work TreeRNNs were introduced in Socher et al. (2011a) based on earlier work by Goller & Küchler (1996), and subsequently applied successfully to a range of tasks involving natural language meaning. Several versions of the tree-structured neural network have been proposed, including the recursive neural tensor network and TreeLSTM discussed above, deep recursive NNs (Irsoy & Cardie 2014), which learn several layers to extract increasingly abstract information at each tree node, dependency tree NNs (Socher et al. 2014), which operate over the non-binary trees produced by dependency parsers, and global belief or upward-downward TreeRNNs (Paulus et al. 2014, Ji & Eisenstein 2015), which use information from the entire tree to compute a final representation at each node.

TreeRNNs have performed well on tasks like sentiment analysis (Socher et al. 2013, Tai et al. 2015) which can be learned from moderate amounts of data. However, the use of a different tree structure for each sentence can cause these models to be dramatically slower to train than sequence-based models, limiting their application elsewhere in NLP. Chapter 7 discusses this issue in more detail and offers a solution.

2.3.3 A note on learning

The models that I study and develop, like most modern neural network models, are trained using minibatch stochastic gradient descent (SGD). In minibatch SGD, training takes the form of a sequence of steps. At each step, backpropagation is used to compute the gradient of the trainable parameters of the model with respect to some objective function (some continuous measure of task success) aggregated over a set of randomly sampled examples from the training set (the size B of this set is fixed as a hyperparameter), which indicates the direction in which to adjust these parameters to improve that objective. The parameters are then adjusted slightly in the direction of the gradient, and another step is taken with a new batch of examples. This process repeats until training converges (i.e., the parameters stop changing) or until some predetermined early stopping criterion indicates that the model's actual task performance is not likely to improve further.

Gradient descent algorithms, including minibatch SGD, require the specification of an update rule which determines how much each parameter will be adjusted in the appropriate direction at each step. This choice is at least somewhat independent of the choice of the type of network chosen, and new optimization functions are being introduced frequently. The non-convex nature of neural network models makes this choice an important one: using an inappropriate or inappropriately tuned optimization function can lead to a model that converges early on a poor solution, or one that never converges to a solution at all.

There is no obvious theoretical reason to choose any one update rule for the research described here. As such, I use the method that appears to be best supported at the time that I start each experiment. I note in each chapter which rule I use.

2.3.4 From sentence encoding to inference

This dissertation incorporates work that is, to the best of my knowledge, the first application of deep learning methods to entailment classification.[1] Because of this, even though my aim in studying NLI is to understand and improve existing sentence-encoding architectures, it is necessary to propose a novel model structure to adapt those model components to this task. This architecture should be as simple as possible while still serving as a viable setting in which to train usable models.

I propose the structure shown in Figure 2.3, which makes it possible to use the sentence-pair classification task of NLI to train sentence-encoding models. In it, each sentence is embedded separately using some encoding sub-network such as a tree or sequence model, a comparison layer (or stack of layers) generates a set of features reflecting the relationship between the two representations, and a softmax classifier uses those features to predict a class label. This model does not allow context from either sentence to be used in computing the representation of the other sentence, as attention-based models (see Section 2.3.5). This limits model performance in practice, but it is helpful in allowing the sentence-encoding sub-networks to be reused without modification and in creating a bottleneck to force the sub-networks to encode as much

[1]Section 6.4.1 surveys concurrent work on deep learning for entailment.

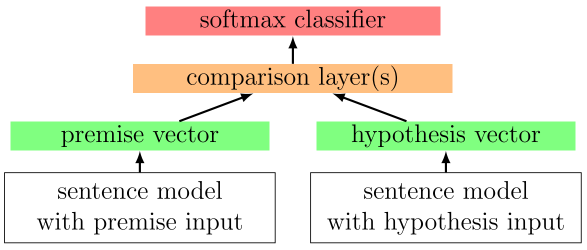

softmax classifier
comparison layer(s)
premise vector
hypothesis vector
sentence model with premise input
sentence model with hypothesis input

Figure 2.3: In the experiments in this dissertation, two copies of a sentence model—like a tree or sequence model as in Figure 2.2—encode the two input sentences. A multilayer classifier component then uses the resulting vectors to predict a label that reflects the logical relationship between the two sentences.

information as possible into the representations of interest. Every experiment in this dissertation uses a version of this model structure.

2.3.5 Other approaches to sentence encoding

Functional semantic types and higher-order tensors

Another line of research (Coecke et al. 2011, Clarke 2012, Clark et al. 2013, Baroni et al. 2014, Grefenstette 2013, Grefenstette et al. 2014, Fried et al. 2015, Bankova et al. 2016) attempts to translate formal notions of function application in compositional semantic structure into a distributional semantics-oriented framework. Clarke (2012) in particular defines a notion of entailment within this framework. None of the models in this framework are designed to be directly implementable as machine learning systems on real data, but rather attempt to define vector and tensor encodings of semantics that are in some sense human-interpretable or formally correct.

It is likely that further developments in this line of research could come to address the key questions of this dissertation by clarifying some of the limits on what semantic information can be efficiently represented in learned distributed representations. However, my focus here is on probing the capabilities of distributed representations by experiment in a learning setting, and there are considerable obstacles to taking that approach with these models. Crucially, these models allow for words to be represented by tensors of arbitrarily high order, and in particular require some function words to have order as high as five. A tensor in this context is a generalization of a matrix: a first order tensor is a vector, a second order tensor is a matrix, and an N-th order tensor is a collection of numeric values arranged along N dimensions. Attempting to learn fifth order tensors directly even for one such word would add a parameter to the model with hundreds of millions of scalar components given a conservative vector dimension of 50. Grefenstette & Sadrzadeh (2013) and Fried et al. (2015) have taken some steps towards showing the possibility of learning within this framework, but much work remains to be done before these models can be implemented in NLP.

Soft attention

Soft attention (Bahdanau et al. 2015, Luong et al. 2015a) is a technique which gives the downstream model (here, the classifier) the potential to access each input token individually through a soft content addressing system. This removes the sentence encoding as a bottleneck through which all semantic information must flow. While this change simplifies the problem of learning complex correspondences between input and output, it also makes it more difficult to build learning experiments that focus on the behavior of the encoder. Because of this, I chose to set aside these models and focus on models that represent sentences a single fixed length vectors.

2.4 Additional prior work

2.4.1 Reasoning in distributed representations

While the work presented in this dissertation is likely the first to apply neural network models to NLI, it is not the first to attempt to use neural networks or distributed representations on practical reasoning problems.

Widdows (2004) presents a system for constructing query vectors for use in information retrieval and builds a practical quasilogic around it which is capable of representing conjunction (*rock* AND *blues*) using vector addition. Subtraction (*rock* NOT *blues*) is represented through the operation of finding the orthogonal vector, expressed as:

(2.13) $$a \text{ NOT } b = a - (a \cdot b)b$$

Widdows & Cohen (2014) present a set of extensions to this system to enable it to support the approximate encoding and retrieval of relation triples from open relation domains (e.g. ⟨insulin, treats, diabetes⟩) and a form of analogical reasoning (similar to that of Mikolov et al. 2013b, below). This work is primarily focused on its target domain of information retrieval, and does not lend itself to direct adaptation to the variety of logical relation types that would be needed for NLI.

Socher et al. (2012) provide a brief proof of concept to show that Boolean negation and conjunction can be precisely implemented in a minimal recursive neural network model with one-dimensional (scalar) binary word representations.

Mikolov et al. (2013b) introduces a simple and strikingly effective arithmetic method for predicting the answers to analogy problems using modern distributional word embeddings. They show, for example, that the representations for *king*, *queen*, *man*, and *woman* roughly validate the following equation.

(2.14) $$\textit{king} - \textit{man} + \textit{woman} = \textit{queen}$$

This work and related subsequent work has provided a valuable window into the reasons for the success of distributional word vectors, but it has so far proved difficult to find or create any kind of similar interpretable structure in general-purpose sentence encodings.

Grefenstette (2013) proposes a way of implementing a simple logic within a vector space model with tensors of different orders representing functions (as in Coecke et al. 2011and subsequent work discussed in Section 2.3.5), and vectors of two different dimensions representing the logical primitives of entities and truth values. This yields a system that combines a form of distributional semantics with some of the logical foundations of formal semantics. He proves that in this implementation, quantifiers are not linear operators, and must be represented using custom nonlinear functions, in an entirely different style from the rest of the terms of the logic.

Weston et al. (2016) introduce a set of question-answering tasks that are meant to test the abilities of neural network question-answering models to do several kinds of domain-specific reasoning involving things like spatial relations, timelines, and transfers of possession. These tasks are presented as natural language, but are generated from simple artificial grammars, making answer selection, rather than question interpretation, the main point of difficulty in the task, and making them therefore largely complementary to the work on sentence interpretation that this dissertation presents.

Vendrov et al. (2016) propose a novel objective function for embedding vectors, including word and sentence encodings, which explicitly manifests the notion of entailment as a simple geometric relation between points in space. This represents a valuable step towards creating interpretable embeddings. However, the new objective only supports relations that induce partial orders, which makes it incapable of explicitly encoding exclusion relations like $|$ or $^\wedge$, and makes it only partially applicable to the view of semantics that I pursue in this dissertation.

In very recent work, Rocktäschel & Riedel (2016) introduce the neural theorem prover and the accompanying method of differentiable backward chaining. This model acts as a theorem prover that uses distributed representations of symbols to search for symbolic proofs of simple expressions. While it operates over directly specified symbols and rules, rather than expressions of natural language, and it is sharply limited in the number and complexity of the rules that it can use, this model is a clear demonstration of the value of distributed representations and their corresponding learned similarity metrics in logical reasoning over language-like domains.

2.4.2 Analyzing learned representations for language

This dissertation focuses on a behavioral approach to understanding the abilities of neural network models. I evaluate models only by running them from end to end on the kinds of data that they were meant to operate over. I should note, though, that there has been a good deal of prior work on the substantially more difficult—but also potentially quite informative—approach of directly inspecting and analyzing the internal state vectors of neural networks.

In some of the earliest work on recurrent neural networks in the modern sense, Elman (1989) presents a simple and intuitive method for analyzing the behaviors of RNNs: projecting hidden states into two dimensional space using principal component analysis (PCA) and then plotting those states as points in space to compare activations for different sequences, or for the same sequence at different time points. This approach has been widely used since (e.g. Vendrov et al. 2016, Narasimhan et al. 2015), often with the substitution of t-SNE (Van der Maaten & Hinton 2008) for PCA.

In other key early work on RNNs, Elman (1990) investigates the memory capacities of recurrent models trained on next word prediction (language models) over simple rule-based artificial languages. The conclusions in that work draw both on patterns in model errors and on the results of a *hierarchical* clustering analysis over model hidden state vectors. Giles et al. (1992) study similar models, and introduce a method for extracting discrete representations of deterministic regular grammars from them after training. A related line of work, exemplified by Garcez et al. (2001), attempts to find algorithms for extracting logical rules from feedforward neural networks trained on nonlinguistic reasoning tasks.

In more recent work, Karpathy et al. (2015) proposes a family of techniques (broadly building on this earlier work) for visualizing modern recurrent neural network language models. While these techniques are helpful aids for analysis, they have not yet yielded clear insights into the internal structure of sentence representations. Li et al. (2016) apply similar techniques, as well as a novel derivative-based method, to sentence-encoding models for sentiment analysis and autoencoding. They find that LSTM-based models, unlike simpler RNNs, tend to show reasonable and interpretable behaviors within the scope of the study. Kádár et al. (2016) introduce some additional techniques for the analysis of these models, with a focus on elucidating the differences between the representations learned by language models and models trained to predict visual features from image captions. This work also provides a helpful survey of work on analyzing neural networks in computer vision.

In a more model-specific approach to analysis, most recent neural attention models (Bahdanau et al. 2015, Rocktäschel et al. 2016, i.a.) have been presented with explicit visualizations of the model-internal lexical alignments that they induce while computing sentence pair relationships.

While this recent work represents a recent surge of progress into the analysis of learned representations, substantial opportunities for future work remain. Of particular relevance to this dissertation, there is still no technology that can come close to revealing the correspondences (or lack thereof) between learned representations for language and discrete representations like logical forms or the structured distributed representations of Coecke et al. (2011).

Chapter 3

Logical reasoning in tree-structured models

This chapter presents work that was published as Bowman, Potts & Manning (2015c) and Bowman, Potts & Manning (2015d). My only coauthors on these papers were my advisors, who were primarily involved in an advisory role rather than in a direct collaboration. However, both contributed edits and a small amount of writing, and Christopher Potts was largely responsible for the design and creation of the artificial dataset used in Section 3.5.

3.1 Introduction

This chapter describes four machine learning experiments that directly evaluate the abilities of tree-structured neural network models to learn representations that support specific semantic behaviors. These tasks follow the format of natural language inference, in which the goal of the model is to determine the relationship between the meanings of two expressions.

In order to train models on this task, I use a version of the architecture introduced in Section 2.3.4. In this architecture, the model uses two TreeRNN or TreeRNTN sentence-encoding components to build representations of two input sentences, and then uses a classifier to produce a judgment for the input sentence pair using only these representations. Using separate representations in this way makes it possible to directly evaluate the ability of the tree-structured sentence-encoding models to learn to represent all of the necessary semantic information in their output representations.

In three of the four experiments in this chapter, I use artificial data to test the models' ability to learn to perform specific patterns of inference that appear to be prevalent in natural language reasoning. I do this by training them to reproduce the judgments of the natural logic of MacCartney & Manning (2009). This logic defines seven mutually-exclusive relations of synonymy, entailment, contradiction, and mutual consistency (introduced previously in Table 2.1), and it provides rules of semantic combination for projecting these relations from the lexicon up to complex phrases.

The goal of these artificial data experiments is to study behaviors that are likely to be difficult for neural network sentence encoders to learn, but are nonetheless likely to be necessary for successful NLI. These experiments involve training models to reproduce the behaviors of a particular logical system in a constrained setting, but I do not claim that the models that I study are effective at learning to do logical inference in any more general sense. I leave the study of the limits of the logical abilities of these models to future work.

The first two experiments in this chapter cover reasoning with logical relations between atomic lexical items (Section 3.3), both on artificial data based on natural logic and on similar natural data extracted from WordNet (Miller 1995, Fellbaum 2010). The next experiment (Section 3.4) covers reasoning with statements constructed compositionally from recursive functions, and the final experiment (Section 3.5) extends this to cover the additional complexity that results from introducing quantifiers.

Though the performance of the plain TreeRNN model is somewhat poor in one experiment, I find that the stronger TreeRNTN model generalizes well in every case, suggesting that it can learn to simulate the target logical concepts. Overall, these experiments fail to find any fundamental obstacle that tree-structured neural networks face in learning semantic representations for natural language, and indeed find that these models can extract complex generalizations in difficult settings without explicit prompting to do so. This suggests that they should be able to learn well and efficiently

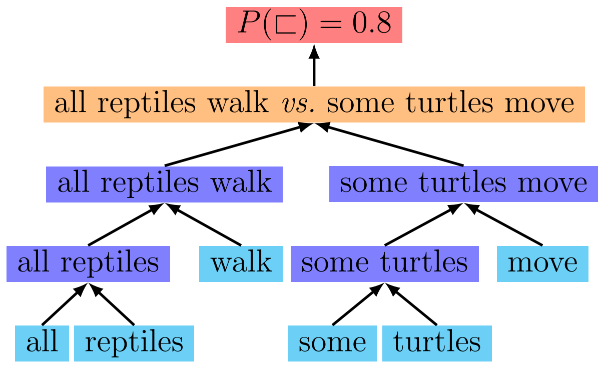


Figure 3.1: In the model evaluated in this chapter, based on the architecture in Section 2.3.4, two separate tree-structured networks (blue and lavender) build up vector representations for each of two sentences using either NN or NTN layer functions (lavender). A comparison layer (orange) then uses the resulting vectors to produce features for a classifier (red).

in the real natural language settings that I examine in the subsequent chapters of this dissertation.

3.2 Methods

This chapter focuses on the evaluation of TreeRNN sentence encoders. They have a clear theoretical motivation from compositionality, and they are the among the most effective preexisting sentence encoding models available.

To apply these tree-structured models to this task, I use the sentence pair model architecture introduced in Section 2.3.4 and depicted in Figure 3.1. In it, the two expressions being compared are processed separately using a pair of tree-structured sentence encoders that share a single set of parameters. The resulting vectors are fed into a separate comparison layer that is meant to generate a feature vector capturing the relation between the two phrases. The output of this layer is then given to a softmax classifier, which produces a distribution over the seven natural logic relations introduced in Table 2.1.

For the sentence-encoding portions of the network, I evaluate TreeRNN models with the standard NN layer function (2.8) and with the more powerful neural tensor network layer function (2.9), as well as a simple sum-of-words baseline. While the TreeRNN layer functions that I study are not the only ones available, I use just these two here as a case study, and hypothesize that positive results are likely to generalize well to tree-structured models with other strong layer functions.

The comparison layer uses the same layer function as the composition layers for the TreeRNNs, and uses a plain TreeRNN layer for the sum-of-words model. In all cases, the comparison layer has an independently learned set of parameters and a separate nonlinearity function. Rather than use a tanh nonlinearity here, I found better results with the leaky rectified linear function of Maas et al. (2013), depicted in 3.1.

$$f(x) = \max(x, 0) + 0.01 \min(x, 0) \tag{3.1}$$

To run the model forward, I assemble the two tree-structured networks so as to match the structure of each sentence. These structures are included in the source data for the experiments in this chapter, but can also be produced using a parser in natural language settings. The word vectors are then looked up from the vocabulary embedding matrix V (one of the learned model parameters), and the composition and comparison functions are used to pass information up the tree and into the classifier. For an objective function, I use the negative log likelihood of the correct label with tuned L2 regularization.

I initialize parameters randomly from a uniform distribution, using the heuristically chosen range $(-0.05, 0.05)$ for layer parameters and $(-0.01, 0.01)$ for embeddings, and train the model using minibatch stochastic gradient descent (SGD) with learning rates computed using AdaDelta (Zeiler 2012). The classifier feature vector is fixed at 75 dimensions and the dimensions of the recursive layers are tuned manually. All models are trained on CPU, with training times varying from hours to days across experiments. (Issues of speed and GPU use are revisited more extensively in Chapter 7.) On the experiments which use artificial data, I report mean results over five fold cross-validation, where variance across runs is typically no more than two percentage points. In addition, because the distribution of labels is not necessarily well balanced in the artificial datasets used in this chapter, I report both accuracy and macroaveraged F1.[1]

3.3 Reasoning about lexical relations

Since natural logic is set in the inferentialist view of semantics, the meanings for expressions are given by their inferential connections with other expressions. For instance, *turtle* is analyzed, not primarily by its extension in the world, but rather by its lexical network: it entails *reptile*, excludes *chair*, is entailed by *sea turtle*, and so forth. With generalized notions of entailment and contradiction, these relationships can be defined for all lexical categories as well as complex phrases, sentences, and even whole texts. The resulting theories of meaning offer valuable new analytic tools for tasks involving database inference, relation extraction, and textual entailment.

Natural logic aligns well with distributed (e.g. vector-based) approaches to representing words, which also naturally model meaning relationally. However, it remains an open question whether it is possible to train such representations to support the kinds of logical reasoning captured by natural logic. In the experiments in this section, I evaluate these models' ability to learn the basic algebra of natural logic relations both from simulated natural logic data (Section 3.3.1) and from the WordNet noun graph (Section 3.3.2). The simulated data offer analytic insights into what the models learn and the WordNet data show how they fare with a real natural language vocabulary. I find that only the NTN is able to fully learn the underlying algebra, but that both models excel in the WordNet experiment.

3.3.1 Learning to represent and infer natural logic relations

The simplest kinds of deduction in natural logic involve atomic statements over pairs of individual lexical items using the seven relations in Table 2.1. For instance, from the relation $p_1 \sqsubset p_2$ between two lexical items, one can infer the relation $p_2 \sqsupset p_1$

[1]I compute macroaveraged F1 as the harmonic mean of average precision and average recall, both computed for all classes for which there is test data, setting precision to 0 where it is not defined.

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

Table 3.1: In Section 3.3.1, I assess the models' ability to learn to do inference over pairs of relations using the rules represented in this figure (adapted from MacCartney 2009), which are derived from the definitions of the relations in Table 2.1. As an example, given that $p_1 \sqsubset p_2$ and $p_2 \wedge p_3$, the entry in the $\sqsubset$ row and the $\wedge$ column indicates that we can conclude $p_1 \mid p_3$. Cells containing a dot correspond to situations for which no valid inference can be drawn with certainty.

by applying the definitions of the relations directly. If one is also given the relation $p_2 \sqsubset p_3$ one can conclude that $p_1 \sqsubset p_3$ by basic set-theoretic reasoning (transitivity of $\sqsubset$). It is also possible to make inferences like these from two pairs that share an argument but not a relation type. For example, given that $p_1 \sqsubset p_2$ and $p_2 \wedge p_4$, one can conclude that $p_1 \mid p_4$. The full set of sound such inferences on pairs of premise relations is depicted in the JOIN TABLE shown as Table 3.1. Though these basic inferences do not involve compositional sentence representations, any successful reasoning using compositional representations will rely on the ability to perform sound inferences of this kind in order to be able to use unseen relational facts within larger derivations. This first experiment studies how well each model can learn to perform them them in isolation.

This problem is of limited formal and computational complexity. It can be implemented relatively simply in a discrete system, and it is plausible that some style of feedforward neural network model (i.e. some neural network model limited to performing a fixed amount of computation per example) could be configured so as to simulate it effectively. It is not at all clear, though, whether an embedding-based model trained using conventional methods is up to the task.

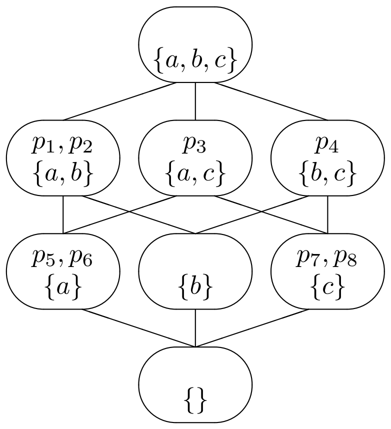


Figure 3.2: Example boolean structure, shown with edges indicating inclusion. The terms p_1–p_8 name the sets. Not all sets have names, and some sets have multiple names, so that learning $\equiv$ is non-trivial.

Experiments

I begin by creating a world model on which to base the statements in the train and test sets. This takes the form of a small Boolean structure in which terms denote sets of entities from a small domain. Figure 3.2 depicts a structure of this form with three entities (a, b, and c) and eight proposition terms (p_1–p_8). I then generate a relational statement for each pair of terms in the model, as shown in Table 3.2. I divide these statements evenly into train and test sets, and delete the test set examples which cannot be proven from the train examples—those for which there is not enough information for even an ideal system to choose a correct label. In each experimental run, I create a model with 80 terms over a domain of 7 elements, yielding a training set of 3200 examples and a test set of 2960 examples.

I train models with both the NN and NTN comparison functions on these datasets. In both cases, the models are implemented as described in Section 3.2. Since the items being compared are single terms rather than full tree structures, the composition layer is not used, and the two models are not recursive. Because of this, the plain TreeRNN is equivalent to the sum-of-words model in this setting. I simply present the each model with the (randomly initialized) embedding vectors for each of two terms, ensuring that the models have no information about the terms being compared except

Train	Test
$p_1 \equiv p_2$	$p_2 \wedge p_7$
$p_1 \sqsupset p_5$	$p_2 \sqsupset p_5$
$p_4 \sqsupset p_8$	~~$p_5 \equiv p_6$~~
$p_5 \mid p_7$	~~$p_7 \sqsubset p_4$~~
$p_7 \wedge p_1$	$p_8 \sqsubset p_4$

Table 3.2: A few examples of atomic statements about the model depicted above. Test statements that are not provable from the training data shown are crossed out.

	Train		Test	
# only	53.8	(10.5)	53.8	(10.5)
15D NN	99.8	(99.0)	94.0	(87.0)
15D NTN	**100**	**(100)**	**99.6**	**(95.5)**

Table 3.3: Performance on the semantic relation experiments. These results and all other results on artificial data are reported as mean accuracy scores over five runs followed by mean macroaveraged F1 scores in parentheses. The # *only* entries reflect the frequency of the most frequent class.

for the relations between them that appear in the training data.

Results

The results (Table 3.3) show that NTN is able to accurately encode the relations between the terms in the geometric relations between their vectors, and is able to then use that information to recover relations that are not overtly included in the training data. The NN also generalizes fairly well, but makes enough errors that it remains an open question whether it is capable of learning representations with these properties. It is still possible that different optimization techniques or finer-grained hyperparameter tuning could lead an NN model to succeed.

As an example from the test data, both models correctly labeled $p_1 \sqsubset p_3$, potentially learning from the training examples $\{p_1 \sqsubset p_{51},\ p_3 \sqsupset p_{51}\}$ or $\{p_1 \sqsubset p_{65},\ p_3 \sqsupset p_{65}\}$. On another example involving comparably frequent relations, the NTN correctly labeled $p_6 \sqsupset p_{24}$, likely on the basis of the training examples $\{p_6 \smallsmile p_{28},\ p_{28} \wedge p_{24}\}$, while the NN incorrectly assigned it #.

These results are quite promising for the ability of these models to learn broadly useful representational systems for natural language. Both models succeeded in inferring a complex logic (the logic described by Table 3.1) from a set of training data, despite being trained to merely memorize that data with no explicit incentive to make high-level generalizations of this kind. The next section shows that this type of behavior is not limited to small artificial lexica, and that both models can learn to generalize in a similar fashion over a large vocabulary of English words as well.

3.3.2 Reasoning about lexical relations in WordNet

Using simulated data as above is reassuring about what the models learn and why, but it is also important to know how they perform with a real natural language vocabulary. Unfortunately, as far as I am aware, there are no available resources that annotate such a vocabulary with the relations from Table 2.1. However, the relations in WordNet (Miller 1995, Fellbaum 2010) come close and pose the same substantive challenges within a somewhat easier classification problem.

I extract three types of relation from WordNet. Hypernym and hyponym are represented explicitly in the WordNet graph structure, and correspond closely to the $\sqsupset$ and $\sqsubset$ relations from natural logic. As in natural logic, these relations are mirror images of one another: if *dog* is a hyponym of *animal* (perhaps indirectly by way of *canid*, *mammal*, etc.), then *animal* is a hypernym of *dog*. I also extract *coordinate* terms, which share a direct hypernym, like *dalmatian*, *pug*, and *puppy*, which are all direct hyponyms of *dog*. Coordinate terms tend to exclude one another, thereby providing a loose approximation of the natural logic exclusion relation $|$. WordNet's antonym relation, however, is both too narrowly defined and too rare to use for this purpose. WordNet defines its relations over sets of synonyms, rather than over individual terms, so I do not include a synonym or equivalent relation, but rather consider only one member of each set of synonyms. Word pairs which do not fall into these three relations are not included in the dataset.

To limit the size of the vocabulary without otherwise simplifying the learning

underachiever	HYPONYM	*enrollee*
agnate	COORDINATE	*enate*
organism	HYPERNYM	*planula*
psittacosaur	HYPONYM	*vertebrate*
organism	HYPERNYM	*celt*

Table 3.4: Randomly selected examples of lexical relations extracted from WordNet.

problem, I extract all of the instances of these three relations for single word nouns in WordNet that are hyponyms of the node `organism.n.01`. In order to balance the distribution of the classes, I slightly down-sample instances of the COORDINATE relation, yielding a total of 36,772 relations among 3,217 terms. Examples of the final result of the extraction process are shown in Table 3.4.

The model and training methods used in this experiment differ slightly from those used elsewhere in this chapter, both in the use of a different label set and in other ways. Embeddings are fixed at 25 dimensions and are initialized either randomly or using distributional word vectors pretrained using GloVe (Pennington et al. 2014).[2] In order to allow the model to reason over a vector space that is structured differently from that of GloVe, I introduce an additional tanh neural network layer between the embedding input and the comparison function. The feature vector produced by the comparison layer is fixed at 80 dimensions. Finally, this experiment was conducted before the other experiments in this chapter, and uses the AdaGrad (Duchi et al. 2011) parameter update rule during training rather than the AdaDelta rule used elsewhere. While this may have contributed to some instability during learning for this experiment, the final results are strong enough that a better update rule could not plausibly offer any substantial improvements.

I report results below using cross-validation, choosing a disjoint 10% test sample for each of five runs. Unlike in the previous experiment, it is not straightforward here to determine in advance how much data should be required to train an accurate model, so I performed training runs with various fractions of the remaining data.

[2]I use a 25-dimensional version of the standard released word vectors which was provided to us directly by the authors.

Portion of	NN		NTN		MFC
training data	w/ GloVe	w/o GloVe	w/ GloVe	w/o GloVe	
100%	99.73 (0.04)	99.37† (0.14)	99.61 (0.02)	**99.95** (0.03)	37.05
33%	95.96 (0.20)	95.30 (0.12)	95.83 (0.35)	95.45† (0.31)	37.05
11%	91.11 (0.24)	90.81† (0.20)	91.27 (0.27)	90.90† (0.13)	37.05

Table 3.5: Mean test % accuracy scores (with standard error) on the WordNet data over five-fold crossvalidation. *MFC* denotes the frequency of the most frequent class (HYPERNYM) as a baseline.

Results

The results are shown in Table 3.5. The structured WordNet data contains substantially more redundancy than the random artificial data above, and I find that the NTN performs perfectly with random initialization. The plain NN performs almost as well, providing a point of contrast with the results of Section 3.3.1. I also find that initialization with GloVe is only marginally helpful, and only shows a detectable impact on performance with smaller amounts of training data, suggesting that the information in these vectors is neither necessary nor sufficient for the modeling of lexical relations in a fully supervised setting like this one. GloVe may be more helpful, though, in more naturalistic settings in which many words are seen rarely or not at all at training time, but are needed at test time.

Some of the randomly initialized model runs failed to learn usable representations at all and labeled all examples with the most frequent labels. I excluded these runs from the statistics, but marked settings for which this occurred with the symbol †. For all of the remaining runs, training accuracy was consistently above 99%.

3.3.3 Discussion

In this section, I evaluate two neural models on the task of learning natural logic relations between distributed word representations. The results suggest that at least the neural tensor network has the capacity to meet this challenge with reasonably-sized training sets, learning both to embed a vocabulary in a way that encodes a diverse set of relations, and to subsequently use those embeddings to infer new relations. In the remainder of this chapter, I extend these results to include complex expressions involving logical connectives and quantifiers, with similar conclusions about (tree-structured recursive versions of) these models.

3.4 Reasoning over recursive structure

A successful natural language inference system must reason about relations not just over familiar atomic symbols, but also over novel structures built up recursively from these symbols. This section shows that recursive models can learn a compositional semantics over such structures. I focus my evaluation of these models on an infinite logical language, which makes it possible to test the model on strings that are longer and more complex than any seen in training.

3.4.1 Experiments

As in Section 3.3.1, I generate artificial data from a formal system, but I now replace the unanalyzed symbols from that experiment with complex formulae. These formulae represent a complete classical propositional logic: each atomic symbol is a variable over the domain {`T`, `F`}, and the only operators are truth-functional ones. Table 3.6 defines this logic, and Table 3.7 gives some short examples of relational statements from this data. To compute the relations between statements, I exhaustively enumerate the sets of assignments of truth values to propositional variables that would satisfy each of the statements, and then I convert the set-theoretic relation between those assignments into one of the seven relations in Table 2.1. As a result, each relational statement represents a valid theorem of the propositional logic, so to succeed, the models must learn a function that identifies valid theorems over this small domain.

Socher et al. (2012) show that a matrix-vector TreeRNN model somewhat similar to the TreeRNTN seen here can learn boolean logic, a logic where the atomic symbols are simply the values `T` and `F`. While learning the operators of that logic is not trivial, the outputs of each operator can be represented accurately by a single bit. In the much more demanding task presented here, the atomic symbols are variables over

Formula	Interpretation
p_1, p_2, p_3, p_4, p_5, p_6	$[\![x]\!] \in \{\mathtt{T}, \mathtt{F}\}$
$not\ \varphi$	$\mathtt{T}$ iff $[\![\varphi]\!] = \mathtt{F}$
$(\varphi\ and\ \psi)$	$\mathtt{T}$ iff $\mathtt{F} \notin \{[\![\varphi]\!], [\![\psi]\!]\}$
$(\varphi\ or\ \psi)$	$\mathtt{T}$ iff $\mathtt{T} \in \{[\![\varphi]\!], [\![\psi]\!]\}$

Table 3.6: Well-formed formulae. φ and ψ range over all well-formed formulae, and $[\![\cdot]\!]$ is the interpretation function mapping formulae into $\{\mathtt{T}, \mathtt{F}\}$.

$not\ p_3$	$\wedge$	p_3
p_3	$\sqsubset$	$p_3\ or\ p_2$
$(not\ p_2)\ and\ p_6$	$\mid$	$not\ (p_6\ or\ (p_5\ or\ p_3))$
$p_4\ or\ (not\ ((p_1\ or\ p_6)\ or\ p_4))$	$\sqsubset$	$not\ ((((not\ p_6)\ or\ (not\ p_4))\ and\ (not\ p_5))\ and\ (p_6\ and\ p_6))$

Table 3.7: Short examples of the type of statements used for training and testing. These are relations between well-formed formulae, computed in terms of sets of satisfying interpretation functions $[\![\cdot]\!]$. Parentheses are used here to disambiguate scope, but examples are supplied to the models as full binary parse trees, and do not include explicit parentheses.

these values, and the sentence vectors must thus be able to distinguish up to 2^{2^6} distinct conditions on valuations.

For these experiments, I randomly generate unique pairs of formulae containing up to 12 tokens of the logical operators (*and*, *or*, *not*) each and compute the relation that holds for each pair. I discard pairs in which either statement is either a tautology or a contradiction (true for all interpretation functions or for no interpretation functions), which are outside the scope of the MacCartney-style natural logic on which I am training these models. The resulting set of formula pairs is then partitioned into 12 bins according the number of operators in the larger of the two formulae. I finally sample 20% of each bin for a held-out test set.

If no constraint is imposed that requires that the two statements being compared are similar in any way, then the generated data are dominated by statements in which the two formulae refer to largely separate subsets of the six variables, which means that the # relation is almost always correct. In an effort to balance the distribution of relation labels without departing from the basic task of modeling propositional logic, I disallow individual pairs of statements from referring to more than four of the six propositional variables.

In order to test the model's generalization to unseen structures, I discard training examples with more than 4 logical operators, yielding 60K short training examples, and 21K test examples across all 12 bins. I train TreeRNN models of both types as well as the sum-of-words (SumNN) baseline. Unlike the two tree models, the baseline does not use word order, and is as such guaranteed to ignore some information that it would need in order to succeed perfectly.

More than the other two experiments in this chapter, this experiment directly tests the ability of the tree-structured models under study to learn the kinds of infinitely recursive compositional grammars that they were designed around. Because of this, it would be especially valuable to use a simpler sequence-based model as a baseline in order to discover whether a given model's success at this task is due specifically to its use of tree structures that provide valuable information about the input expressions, or to some more general property that would apply to neural networks with differently structured composition techniques. There are substantial complications to setting up this comparison effectively, and I defer it to Chapter 4 for more extensive coverage.

3.4.2 Results

Figure 3.3 shows the relationship between test accuracy and statement size. While the summing baseline model performed poorly across the board, both tree-structured models were able to perform well on unseen small test examples, with TreeRNN accuracy above 98% and TreeRNTN accuracy above 99% on formulae below length five, indicating that they learned correct approximations of the underlying logic. Training accuracy was 66.6% for the SumNN, 99.4% for the TreeRNN, and 99.8% for the TreeRNTN.

After the size four training cutoff, performance gradually decays with expression size for both tree models, suggesting that the learned approximations were accurate but lossy. Despite the TreeRNTN's stronger performance on short sentences, its performance decays more quickly than the TreeRNN's. This suggests that it learns

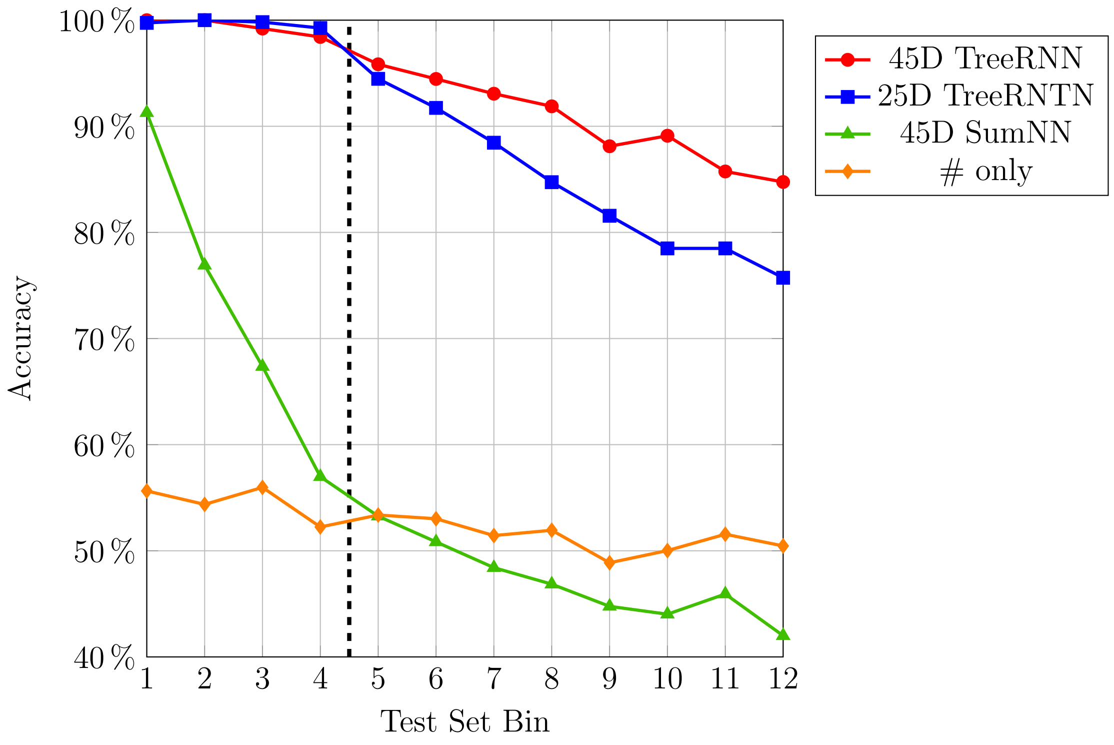


Figure 3.3: Results on recursive structure. The horizontal axis divides the test set into bins by the number of logical connectives in each sentence, and the vertical axis reflects test accuracy. The vertical dotted line marks the size of the longest training examples. # *only* reflects the performance that can be reached by always guessing #, the most frequent relation.

to interpret many specific fixed-size tree structures directly, allowing it to get away without learning as robust generalizations about how to compose terms in the general case. Two factors may make the TreeRNTN prone to learning these more complex generalizations: even with the lower dimension, the TreeRNTN composition function has about eight times as many parameters as the TreeRNN, and the TreeRNTN works best with weaker L2 regularization than the TreeRNN ($\lambda = 0.0003$ vs. 0.001). However, even in the most complex set of test examples, the TreeRNTN classifies true examples of every class but $\equiv$ (which is rare in long examples, and occurs only once here) correctly the majority of the time, and the performance of both models on those examples indicates that both learn reasonable approximations of the underlying logical reasoning task over recursive structure.

3.5 Reasoning with quantifiers and negation

We have seen that tree-structured models can learn to represent and reason with short expressions of propositional logic. However, natural languages can express functional meanings of considerably greater complexity than this. As a test of whether these models can capture this complexity, I now investigate the degree to which they are able to develop suitable representations for the semantics of natural language quantifiers like *most* and *all* as they interact with negation and lexical entailments. Quantification and negation are far from the only place in natural language where complex functional meanings are found, but they are natural focus, since they have formed a standard case study in prior formal work on natural language inference (e.g. Icard III & Moss 2013b).

3.5.1 Experiments

The data consist of pairs of sentences generated from a grammar for a simple English-like artificial language. Each sentence contains a quantifier, a noun which may be negated, and an intransitive verb which may be negated. I use the quantifiers *some*, *most*, *all*, *two*, and *three*, and their negations *no*, *not-all*, *not-most*, *less-than-two*, and *less-than-three*, and also include five nouns, four intransitive verbs, and the negation symbol *not*. In order to be able to define relations between sentences with differing lexical items, I define the lexical relations for each noun–noun pair, each verb–verb pair, and each quantifier–quantifier pair. The grammar then generates pairs of sentences and calculates the relations between them.[3] For instance, the models might then see pairs like 3.2 and 3.3 in training and be required to then label 3.4.

(3.2) (most turtle) swim $|$ (no turtle) move

(3.3) (all lizard) reptile $\sqsubset$ (some lizard) animal

[3]The relations are assigned by a task-specific implementation of MacCartney and Manning's logic. This logic is sound, but it is not complete even over this small fixed fragment of English, and there are some generated pairs to which the logic does not assign a relation label. These include those that require inferences based on De Morgan's laws, such as *(all pets) growl* $^\wedge$ *(some pet) (not growl)*. Pairs which are not assigned labels are omitted from the corpus.

	Train		Test	
# only	35.4	(7.5)	35.4	(7.5)
25D SumNN	96.9	(97.7)	93.9	(95.0)
25D TreeRNN	99.6	(99.6)	99.2	(99.3)
25D TreeRNTN	**100**	(**100**)	**99.7**	**(99.5)**

Table 3.8: Performance on the quantifier experiments, given as % correct and macroaveraged F1.

(3.4) (most turtle) reptile | (all turtle) (not animal)

In each run, I randomly partition the set of valid *single sentences* into train and test, and then label all of the pairs from within each set to generate a training set of 27K pairs and a test set of 7K pairs. Because the model doesn't see the test sentences at training time, it cannot directly use the kind of reasoning described in Section 3.3 at the sentence level (by treating sentences as unanalyzed symbols), and must instead jointly learn the word-level relations and a complete reasoning system over them for the logic.

I use the same summing baseline as in Section 3.4. The highly consistent sentence structure in this experiment means that this model is not as disadvantaged by the lack of word order information as it is in the previous experiment, but the variable placement of *not* nonetheless introduces potential uncertainty in the 58.8% of examples that contain a sentence with a single token of it.

3.5.2 Results

The results (Table 3.8) show that both tree models are able to learn to generalize the underlying logic almost perfectly. The baseline summing model can largely memorize the training data, but does not generalize as well. I do not find any consistent pattern in the handful of errors made by either tree model, and no errors were consistent across model restarts, suggesting that there is no fundamental obstacle to learning a perfect model for this problem.

3.6 Discussion and conclusion

This chapter evaluates two tree-structured recursive neural network models on four inference tasks over clean data, covering the core capacities of natural logic with entailment and exclusion, recursive structure, and quantification. The results suggest that TreeRNTNs, and potentially also TreeRNNs, can learn to faithfully reproduce logical inference behaviors from reasonably-sized training sets. These positive results are promising for the future of learned representation models in the applied modeling of compositional semantics.

Some questions about the abilities of these models remain open. Even the TreeRNTN falls short of perfection in the recursion experiment, with performance falling off steadily as the size of the expressions grows. It remains to be seen whether these deficiencies are limiting in practice, and whether they can be overcome with stronger models or better optimization techniques. In addition, interesting analytical questions remain about the particular type and structure of the internal vector representations that these models use to encode logical primitives and composed expressions. Neither the underlying logical theories nor any straightforward parameter inspection technique provides much insight on this point, but further research may reveal structure in the learned parameters or the representations they produce.

The next chapter explores the degree to which simpler *recurrent* neural network models (RNNs) can show some of these same symbolic reasoning abilities, and finds that they are substantially less efficient learners in this setting. Following that, the remainder of this dissertation shows that both recurrent and recursive models are effective at learning from real natural language data, and that a novel model that serves as a hybrid between the two is even more effective.

Chapter 4

Logical reasoning in sequence models

This chapter presents work that was published as Bowman, Manning & Potts (2015b). My only coauthors on this paper were my advisors, who were primarily involved in an advisory role rather than in a direct collaboration.

4.1 Introduction

While both tree and sequence-based sentence-encoding models perform well on many tasks, and both are under active development, tree models are often presented as the more principled choice, since they align with standard linguistic assumptions about constituent structure and the compositional derivation of complex meanings. Nevertheless, tree models have not shown the kinds of dramatic performance improvements over sequence models that their billing would lead one to expect: head-to-head comparisons with sequence models show either modest improvements (Tai et al. 2015) or none at all (Li et al. 2015).

In this chapter, I investigate a hypothesis that could explain these results: standard sequence models may be able to learn to exploit recursive syntactic structure in generating representations of sentence meaning, thereby covertly using the same structure that tree models are explicitly designed around. This would require that sequence models be able to identify syntactic structure in natural language. This appears to be plausible on the basis of other recent research (Vinyals et al. 2015, Karpathy et al. 2015). In this chapter, I use experiments based on those in the previous chapter to evaluate whether LSTM sequence models are able to use such structure to guide interpretation, focusing on cases where syntactic structure is clearly indicated in the data.

I compare standard tree and sequence models on their handling of recursive structure by training the models on sentences whose length and recursion depth are limited, and then testing them on longer and more complex sentences, such that only models that exploit the recursive structure will be able to generalize in a way that yields correct interpretations for these test sentences. The methods extend those of Section 3.4, which introduces an experiment and corresponding artificial dataset to test this ability in two tree models. I adapt that experiment to sequence models by decorating the statements with an explicit bracketing, and I use this design to compare an LSTM sequence model with three tree models, with a focus on what data each model needs in order to generalize well.

Building on the results in Section 3.4, I first show that standard tree neural networks are able to make the necessary generalizations, with their performance decaying gradually as the structures in the test set grow in size. I additionally find that extending the training set to include larger structures mitigates this decay. Then considering sequence models, I find that a single-layer LSTM RNN can also generalize to unseen large structures to some extent, but that it does this only when trained on a much larger and more complex training set than is needed by the tree models to reach the same generalization performance.

These results engage with those of Vinyals et al. (2015) and Dyer et al. (2015), who find that sequence models can learn to extract syntactic structure from natural language sentences, at least when trained on explicitly syntactic tasks. The simplest model presented in Vinyals et al. (2015) uses an LSTM sequence model to encode each sentence as a vector, and then generates a linearized parse (a sequence of brackets and constituent labels) with high accuracy using only the information present in the vector. This shows that the LSTM is able to identify the correct syntactic structures

$not\, p_3$	$\wedge$	p_3
p_3	$\sqsubset$	$p_3\ or\, p_2$
$(not\, p_2)\ and\, p_6$	$\mid$	$not\,(p_6\ or\,(p_5\ or\, p_3))$
$p_4\ or\,(not\,((p_1\ or\, p_6)\ or\, p_4))$	$\sqsubset$	$not\,((((not\, p_6)\ or\,(not\, p_4))\ and\,(not\, p_5))\ and\,(p_6\ and\, p_6))$

Table 4.1: Examples of short to moderate length pairs from the artificial data introduced in Chapter 3. I only show the parentheses that are needed to disambiguate the sentences rather than the full binary bracketings that the models use.

and also hints that it is able to develop a generalizable method for encoding these structures in vectors.

The experiments in this chapter show that LSTM sequence models can learn to understand tree structures in a generalizable way when given enough data. I also find, though, that sequence models lag behind tree models dramatically across the board, even on training corpora that are quite large relative to the complexity of the underlying grammar, suggesting that tree models can play a valuable role in modeling natural language understanding.

4.2 Recursive structure in artificial data

The artificial language The language described in Section 3.4, which I use here, is designed to highlight the use of recursive structure with minimal additional complexity. Its vocabulary consists only of six unanalyzed word types ($p_1, p_2, p_3, p_4, p_5, p_6$), *and*, *or*, and *not*. Sentences of the language can be straightforwardly interpreted as statements of propositional logic (where the six unanalyzed words types are variable names), and labeled sentence pairs can be interpreted as theorems of that logic. Some example pairs are provided in Table 4.1.

Crucially, the language is defined such that any sentence can be embedded under negation or conjunction to create a new sentence, allowing for arbitrary-depth recursion, and such that the scope of negation and conjunction are determined only by bracketing with parentheses (rather than bare word order). The compositional structure of each sentence can thus be an arbitrary tree, and interpreting a sentence correctly requires using that structure.

The data come with parentheses representing a complete binary bracketing. The models use this information in two ways. For the tree models, the parentheses are not word tokens, but rather are used in the expected way to build the tree. For the sequence model, the parentheses are word tokens with associated learned embeddings. This approach provides the models with equivalent data, so their ability to handle unseen structures can be reasonably compared.

The data The sentence pairs are divided into thirteen bins according to the number of logical connectives (*and, or, not*) in the longer of the two sentences in each pair. I test each model on each bin separately (58K total examples, using an 80/20% train/test split) in order to evaluate how each model's performance depends on the complexity of the sentences. In four experiments, I train the models on the training portions of bins 0–3 (62K examples), 0–4 (90K), 0–6 (160K), and 0–10 (230K), and test on every bin but the trivial bin 0. Capping the size of the training sentences allows us to evaluate how the models interpret the sentences: if a model's performance falls off abruptly above the cutoff, it is reasonable to conclude that it relies heavily on specific sentence structures and cannot generalize to new structures. If a model's performance decays gradually with no such abrupt change, then it must have learned a more generally valid interpretation function for the language which respects its recursive structure.

When evaluating any sentence-encoding model on sentences longer than those seen at training time, it is inevitable to see at least some decay in accuracy—this experiment only attempts to compare the magnitude of this decay across models. Since sentences are fixed-dimensional vectors of fixed-precision floating point numbers, all models will make errors on sentences above some length, and L2 regularization (which empirically helps performance by preventing the model from overfitting on the finite training data) exacerbates this by discouraging the model from using the kind of numerically precise, nonlinearity-saturating functions that would ultimately generalize best.

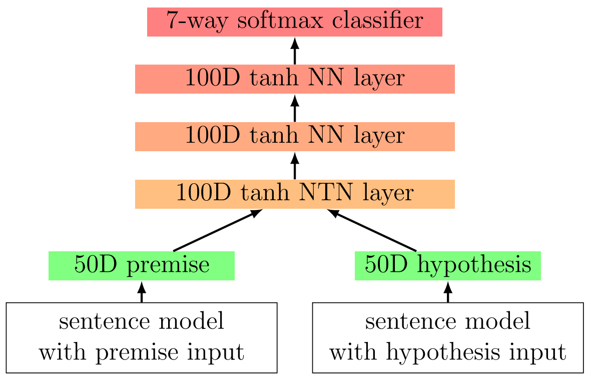
4.3 Testing sentence models on entailment

I use the architecture depicted in Figure 4.1a, which builds on the one used in Chapter 3. The model architecture uses two copies of a single sentence model (a tree or sequence model) to encode the premise and hypothesis (left and right side) expressions, and then uses those encodings as the features for a multilayer classifier which predicts one of the seven relations. Since the encodings are computed separately, the sentence models must encode complete representations of the meanings of the two sentences if the downstream model is to succeed.

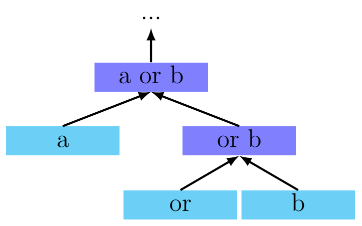
Two points must be considered when comparing tree-structured models to recurrent ones. Most importantly, the two do not take the same structures as input: tree-structured models must be provided with a parse structure, which is generally not provided when using sequence models. In order to ensure that any performance difference between the two approaches is not strictly due to this difference in available information, it is necessary to experiment with sequence models that have access to linearized parse structure information. Here, this takes the form of explicit open- and close-bracket tokens. In addition, it is important to distinguish between differences in model structure and differences in layer function. While the relatively weak plain RNNs and plain TreeRNNs use essentially the same layer function, the more effective LSTMs, TreeLSTMs, and TreeRNTNs use distinctive layer functions that are to some extent entangled with specific network architectures, such that the choice of architecture and the choice of layer function are tied.

Classifier The classifier component of the model consists of a combining layer which takes the two sentence representations as inputs, followed by two neural network layers, then a softmax classifier.

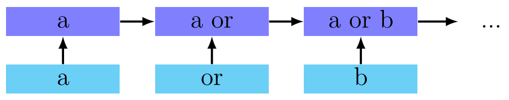
This model is largely identical to the model from Chapter 3, but adds the two additional tanh NN layers, which I found help performance across the board, and also uses the NTN combination layer when evaluating all four models, rather than just the TreeRNTN model, so as to ensure that the sentence models are compared in as similar a setting as possible.



(a) The general architecture shared across models.



(b) The architecture for the tree-structured sentence models. Terminal nodes are learned embeddings and non-terminal nodes are NN, NTN, or Tree-LSTM layers.



(c) The architecture for the sequence sentence model. Nodes in the lower row are learned embeddings and nodes in the upper row are LSTM layers.

Figure 4.1: In this model, which follows the basic structure introduced in Figure 2.3, two copies of a sentence model—based on either tree (b) or sequence (c) models—encode the two input sentences. A multilayer classifier component (a) then uses the resulting vectors to predict a label that reflects the logical relationship between the two sentences.

Sentence models The sentence-encoding component of the model transforms the (learned) embeddings of the input words for each sentence into a single vector representing that sentence. I experiment with tree-structured models (Figure 4.1b) with TreeRNN (2.8), TreeRNTN (2.9), and TreeLSTM (2.10–2.12) activation functions. In addition, I use a sequence model (Figure 4.1c) with an LSTM activation function (2.5–2.7). In experiments with a simpler non-LSTM RNN sequence model (2.4), the model tended to badly underfit the training data, and those results are not included here.

Initialization All parameters are initialized with samples from a uniform distribution, using a strategy that roughly follows Glorot & Bengio (2010). For most matrix-valued parameters, this distribution is over $\pm\frac{1}{\sqrt{D_{\text{in}}}}$, where D_{in} is the combined dimensions of the inputs to the relevant layer. For the RNTN layer, the matrix parameter is initialized in $\pm\frac{0.1}{\sqrt{D_{\text{in}}}}$ and the tensor parameter is initialized in $\pm\frac{0.9}{\sqrt{D_{\text{in}}}}$. All bias vectors are initialized to zero.

Training I train all four models using minibatch SGD with AdaDelta (Zeiler 2012) learning rates. The objective is the standard negative log likelihood classification objective with L2 regularization. The regularization constant λ was tuned manually on a separate train-test split, and was set to 1×10^{-4} for the plain TreeRNN and 3×10^{-4} for the other three models. All models are trained for 100 epochs, after which all have largely converged without significantly declining from their peak performances.

4.4 Results and discussion

The results are shown in Figure 4.2. The tree models fit the training data well, reaching 99.2, 99.0, and 98.8% overall accuracy respectively in the $\leq$6 setting, with the LSTM underfitting at 94.2%. In that setting, all models generalized well to relatively short test examples, with the tree models all surpassing 98% on examples in bin 4, and the LSTM reaching 95.6%. On longer test sentences, including those before the training cutoff, the tree models decay smoothly in performance across the

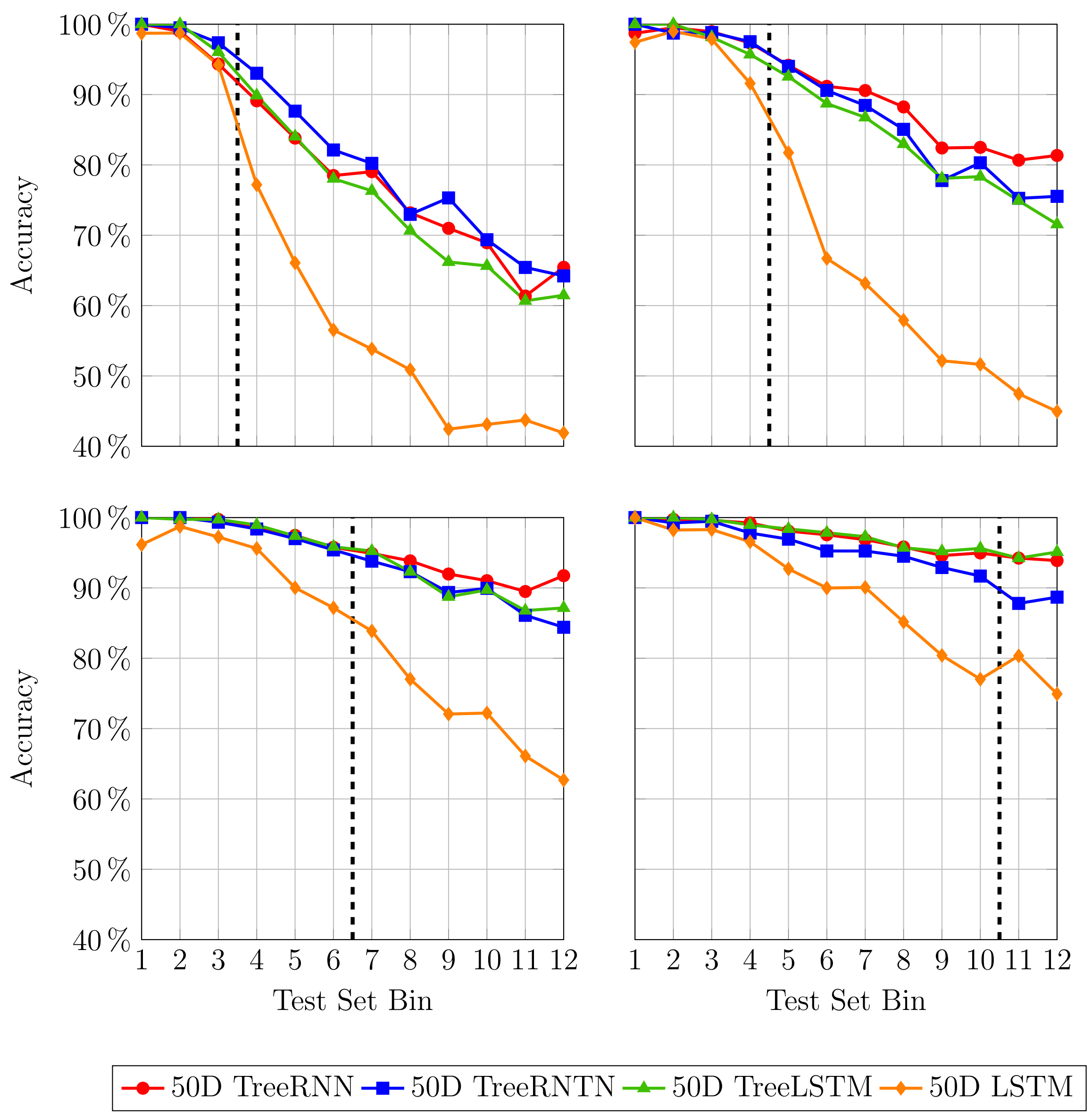


Figure 4.2: Test accuracy on four experiments using bins 0–3, 0–4, 0–6, and 0–10 for training. The horizontal axis on each graph divides the test set expression pairs into bins by the number of logical operators in the more complex of the two expressions in the pair. The dashed line shows the size of the largest examples in the training set in each experiment.

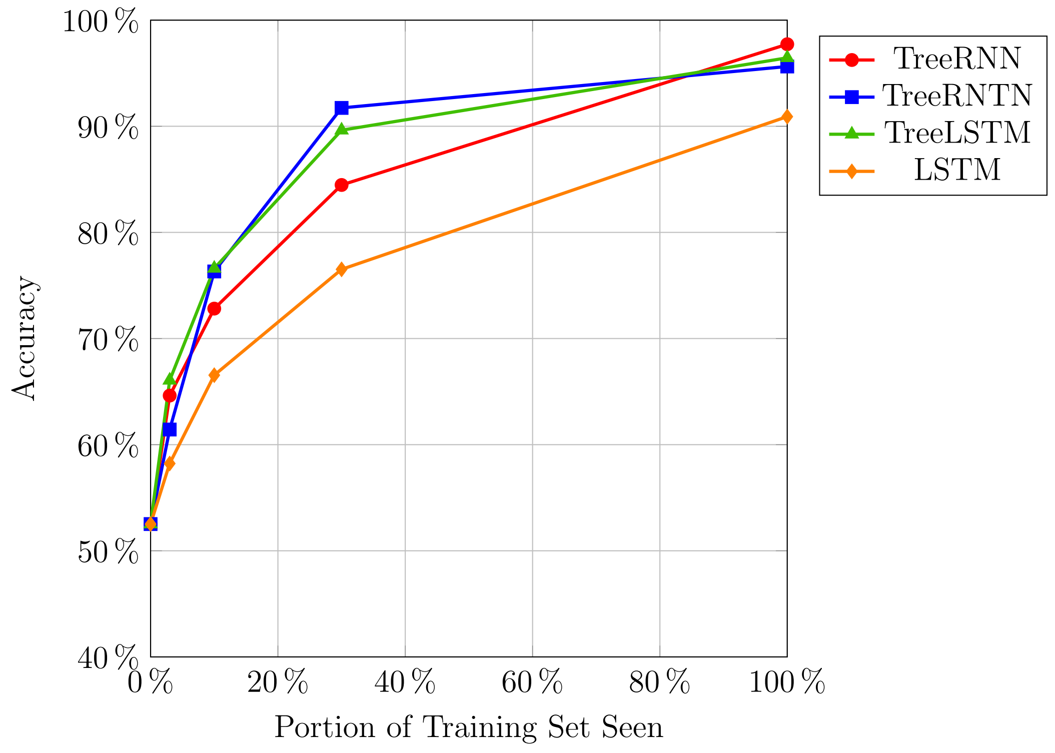


Figure 4.3: Learning curve for the ≤6 experiment.

board, while the LSTM decays more quickly and much more abruptly, with a striking difference in the ≤4 setting, where LSTM performance falls about 10% from bin 4 to bin 5, compared to 3% for the next worse model. However, the LSTM improves considerably with more ample training data in the ≤6 setting, showing only a 3% drop and generalization results comparable to the best model's in the ≤3 setting. In the ≤10 setting, the LSTM improves further, roughly matching the performance of the best model in the dramatically harder ≤4 setting.

All four models robustly beat the simple baselines reported in Figure 3.3.

The learning curve (Figure 4.3) suggests that additional data is unlikely to change these basic results. The LSTM lags behind the tree models by a substantial margin across the curve, but appears to gain accuracy at a similar rate.

4.5 Conclusion

I find that all four models are able to learn to reason over novel sentences of a recursive language to some extent. Further, I find that tree models' biases allow them to do this with much greater efficiency, outperforming sequence-based models substantially in every experiment. However, the sequence model is nonetheless able to generalize smoothly from seen sentence structures to unseen ones, showing that its lack of explicit recursive structure does not categorically prevent it from recognizing recursive structure in the artificial language.

I interpret these results as evidence that both tree and sequence architectures can play valuable roles in the construction of sentence models over data with recursive syntactic structure. Tree architectures provide an explicit bias that makes it possible to efficiently learn to compositional interpretation, which is difficult for sequence models. Sequence models, on the other hand, lack this bias, but have other advantages. Since they use a consistent graph structure across examples, it is easy to accelerate minibatch training in ways that yield substantially faster training times than are possible with tree models, especially with GPUs. In addition, when sequence models integrate each word into a partial sentence representation, they have access to the entire sentence representation up to that point, which may provide valuable cues for the resolution of lexical ambiguity, which is not present in the artificial language, but is a serious concern in natural language text.

Finally, I suggest that, because of the well-supported linguistic claim that the kind of recursive structure that I study here is key to the understanding of real natural languages, there is likely to be value in developing sequence models that can more efficiently exploit this structure without fully sacrificing the flexibility that makes them succeed. The remainder of the dissertation works towards this hypothesis. After Chapters 5 and 6 introduce the necessary data, Chapter 7 introduces a novel model that builds on the strengths of both tree and sequence models, and shows that this new hybrid model is a fast and uniquely effective sentence encoder.

Chapter 5

A new corpus for NLI

This chapter presents work that was published as Bowman, Angeli, Potts & Manning (2015a). I was primarily responsible for the design and creation of the corpus described in this paper, though Gabor Angeli was an active collaborator throughout its development. The other two co-authors, my advisors, contributed to this paper primarily in an advisory role. In addition, Section 5.2 in this chapter presents work that is excerpted from Bowman, Potts & Manning (2015d), which also forms the basis for much of Chapter 3. That work is my own. Finally, Section 5.4 was composed for this dissertation and is entirely my own.

5.1 Introduction

My objective in this dissertation is to provide an empirical evaluation of representation learning approaches to NLI and to advance the case for NLI as a tool for the evaluation of domain-general approaches to semantic representation. The experiments presented so far in Chapters 3 and 4 have done this using experiments over artificial data, an approach which is informative, but which cannot satisfy this objective alone. In this chapter, I move to the use of natural language. While in previous chapters, it was possible to create valid inference data by simulating the behaviors of sound logics, this option is unavailable here.

Existing NLI corpora do not permit the thorough assessment of low-bias machine

A man inspects the uniform of a figure in some East Asian country.	**contradiction** C C C C C	The man is sleeping
An older and younger man smiling.	**neutral** N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	**contradiction** C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	**entailment** E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	**neutral** N N E C N	A happy woman in a fairy costume holds an umbrella.

Table 5.1: Randomly chosen examples from the development section of the new corpus, shown with both the selected gold labels and the full set of labels (abbreviated) from the individual annotators, including (in the first position) the label used by the initial author of the pair.

learning models like neural networks. With only hundreds or thousands of training examples, these corpora are generally too small for training low-bias models like neural networks. In addition, many contain sentences that are algorithmically generated, and many suffer from issues of inconsistent handling of event and entity coreference in labeling, both of which significantly impact the interpretability of any results.

To address this, this chapter presents the Stanford Natural Language Inference (SNLI) corpus, a collection of sentence pairs labeled for entailment, contradiction, and semantic independence. At 570,152 sentence pairs, SNLI is two orders of magnitude larger than all other resources of its type. And, in contrast to many such resources, all of its sentences and labels were written by humans in a grounded, naturalistic context. In a separate validation phase, we collected four additional judgments for each label for 56,941 of the examples. Of these, 98% of cases emerge with a three-annotator consensus, and 58% see a unanimous consensus from all five annotators. Table 5.1 shows a few examples.

5.1.1 Why not some other task?

There is a sound intellectual case to be made for NLI as an evaluation task (advanced in Chapter 1), but since the collection of a new corpus is a large and expensive endeavor, it is worth pausing here to consider alternative tasks for which high-quality corpora are already available.

The task of machine translation can also be used to train and evaluate neural network sentence encoders, and that task also demands some degree of sensitivity to compositional syntactic and semantic structure. In a practical sense, translation is the more appealing task, since translation data is generated in the course of business for many organizations, leaving a great deal of data available. I argue, though, that NLI is the better benchmark for developing NNs for language understanding. This is for two reasons:

- Typical translation tasks require natural language generation, which is a separate difficult problem that must be learned in parallel with the semantic encoding task of interest, making results harder to interpret. For NLI, the only tool that is needed beyond the sentence encoder is a well-understood conventional classifier.

- Contradiction vs. entailment decisions in particular specifically target the abilities of NN models to learn lexical and phrasal representations (like alternation) that don't resemble similarity, either in their correlation with distributional information or their transitivity behavior. There is no close parallel to this in machine translation. Since modeling similarity is almost the only aspect of NN behavior in NLP that's reasonably well understood and essentially known to work, using a benchmark that explicitly demands something more sophisticated than this is likely to pay off by better exposing the weaknesses of current standard models.

Sentiment evaluation has also been used in recent work to evaluate sentence-encoding models, and online review sites provide an easy way to collect naturally-occurring sentiment data, making sentiment another practical alternative to inference. However, sentiment analysis evaluates only one narrow aspect of language meaning, rather than the broad spectrum of phenomena that an NLI system must understand in order to succeed.

In addition, several question-answering-based tasks have been used to evaluate neural networks in the recent literature (Iyyer et al. 2014, Bordes et al. 2016). For a model to succeed at almost any version of the question-answering task, it needs to show sophisticated language understanding. However, existing question-answering task definitions and datasets reliably present models with substantial additional challenges that can mask their performance at understanding, including the challenges of efficiently searching through large knowledge sources, integrating potentially conflicting sources of information, and phrasing answers.

5.1.2 Corpora in semantics

Beyond evaluating neural network models, there is also a second reason to pursue the creation of a large human-annotated NLI corpus. Corpus methods on such a dataset have the potential to offer a range of fast reproducible experimental designs to researchers in semantics. These methods have so far been limited by a lack of large corpora with any kind of semantic grounding. Because of this, it has only been possible to use corpus methods to study the usage and distribution of constructions with known semantics rather than to study the semantics of novel constructions directly.

An ideal corpus for truth-conditional semantics (as discussed in Section 1.1.1) would pair utterances in context with either a truth value or, even better, a set of truth conditions expressed in a logical form of some kind. Doing this requires specifying a representational system to describe the contexts of utterances, and in the latter case, the contents of those utterances as well. How best to represent this kind of information, though, is very much an open problem, and in fact it is the very problem that such a corpus would be meant to help solve.

Creating a natural language inference corpus instead makes it possible to collect a representation-agnostic corpus. In this framing, natural language is treated as its own representation language, and the semantic annotation for a sentence consists simply of examples of sentences with respect to which it is entailing, contradictory, or semantically neutral. High quality corpora of this kind like RTE and SICK exist, but at a few thousand examples they have so far been too small for corpus methods to offer much insight, and the fact that they were constructed using detailed annotation guidelines limits their value as examples of real human language use.

Crucially, as I discussed in Section 1.1.5, even semantic theories built up entirely around model-theoretic representations make clear, testable predictions about entailments. I hope that SNLI will make it possible to quantitatively evaluate semantic theories that bear on phenomena found in captions, and that it will pave the way for informative corpus semantic work on future corpora.

5.2 Training neural networks with existing data

In this section, we investigate what can be accomplished in NLI using neural networks over existing training data. While none of the existing datasets discussed in Section 2.1.3 is ideal for neural network training, the SICK entailment corpus's 4500 training examples may be large enough to support at least some limited learning.

The model that we use for this experiment is based on the one used in Chapter 3, with a few additions. In order to better handle rare words, we initialize the word embeddings using 200 dimensional vectors trained with GloVe (Pennington et al. 2014) on data from Wikipedia. Since 200 dimensional vectors are too large to be practical in a TreeRNTN on a small dataset due to the TreeRNTN's D^3-element tensor parameter, a new embedding transformation layer is needed to make it possible to scale the embeddings down to a manageable size. Before any embedding is used as an input to a recursive layer, it is passed through an additional tanh neural network layer with the same output dimension as the recursive layer. This new layer allows the model to choose which aspects of the 200 dimensional representations from the unsupervised source it most values, rather than relying on GloVe—which has no knowledge of the task—to do so, as would be the case were GloVe asked to directly produce vectors of the lower dimensionality. An identical layer is added to the sum-of-words model between the word vectors and the comparison layer.

We also supplement the SICK training data[1] (4500 examples) with 600K examples

[1]We tuned the model using performance on a held out development set, but report performance

The patient is being helped by the doctor	ENTAILMENT	The doctor is helping the patient (PASSIVE)
A little girl is playing the violin on a beach	CONTRADICTION	There is no girl playing the violin on a beach (NEG)
The yellow dog is drinking water from a bottle	CONTRADICTION	The yellow dog is drinking water from a pot (SUBST)
A woman is breaking two eggs in a bowl	NEUTRAL	A man is mixing a few ingredients in a bowl (MULTIED)
Dough is being spread by a man	NEUTRAL	A woman is slicing meat with a knife (DIFF)

Table 5.2: Examples of each category used in error analysis from the SICK test data.

of approximate entailment data from the Denotation Graph project (DG; Young et al. 2014), a corpus of noisy automatically labeled entailment examples over image captions—the same genre of text from which SICK was drawn—which was also used by the winning SICK submission. We trained a single model on data from both sources, but used a separate set of softmax parameters for classifying into the labels from each source, and forced the model to sample SICK examples and DG examples about equally often during training.

We parse the data from both sources with the Stanford PCFG Parser v. 3.3.1 (Klein & Manning 2003). We also find that training an effective model with such limited data requires an additional technique: we collapse subtrees that were identical across both sentences in a pair by replacing them with a single head word. The training and test data on which we report performance are collapsed in this way, and both collapsed and uncollapsed copies of the training data are used in training. Finally, in order to improve regularization on the noisier data, we add tuned dropout (Srivastava et al. 2014) at the input to the comparison layer (90% keep rate) and at the output from the embedding transform layer (25% keep rate).

here for a version of the model trained on both the training and development data and tested on the 4,928 example SICK test set. We also report training accuracy on a small sample from each data source.

	NEUTRAL only	30D SumNN	30D TreeRNN	50D TreeRNTN
DG Train	50.0	68.0	67.0	**74.0**
SICK Train	56.7	96.6	95.4	**97.8**
SICK Test	56.7	73.4	74.9	**76.9**
PASSIVE (4%)	0	76	68	**88**
NEG (7%)	0	96	**100**	**100**
SUBST (24%)	28	**72**	64	**72**
MULTIED (39%)	**68**	61	66	64
DIFF (26%)	**96**	68	79	**96**
SHORT (47%)	50.0	73.9	73.5	**77.3**

Table 5.3: Classification accuracy, including a category breakdown for SICK test data. Categories are shown with their frequencies.

5.2.1 Results

Despite the small amount of high quality training data available and the lack of resources for learning lexical relationships, the results (Table 5.3) show that the tree-structured models perform competitively on textual entailment, beating a strong baseline. Neither model reaches the performance of the winning system in the SemEval competition (84.6%), but the TreeRNTN did exceed that of eight out of 18 submitted systems, including several which used sophisticated hand-engineered features and lexical resources specific to the version of the entailment task at hand.

To better understand the results, I manually annotated a fraction of the SICK test set, using mutually exclusive categories for passive/active alternation pairs (PASSIVE), pairs differing only by the presence of negation (NEG), pairs differing by a single word or phrase substitution (SUBST), pairs differing by multiple edits (MULTIED), and pairs with little or no content word overlap (DIFF). Examples of each are in Table 5.2. We annotated 100 random examples to judge the frequency of each category, and continued selectively annotating until each category contained at least 25. I also use the category SHORT for pairs in which neither sentence contains more than ten words.

The results (Table 5.3) show that the TreeRNTN performs especially strongly in the two categories which pick out specific syntactic configurations, PASSIVE and

NEG, suggesting that that model has learned to encode the relevant structures well. It also performs fairly on SUBST, which most closely parallels the lexical entailment inferences addressed in Section 3.5. In addition, none of the models perform dramatically better on the SHORT pairs than on the rest of the data, suggesting that the performance decay observed in Chapter 4 may not impact models trained on typical natural language text, or that short sentences in this corpus tend to be more difficult, possibly because they lack useful information that would be spelled out in longer sentences.

It is known that a model can perform well on SICK (like other natural language inference corpora) without taking advantage of compositional syntactic or semantic structure (Marelli et al. 2014a), and the summing baseline model is powerful enough to do this. The tree models nonetheless perform substantially better, suggesting that given sufficient data it should be possible for the tree models, and not the summing model, to learn a truly high-quality solution, a possibility that I explore in Chapter 7.

5.3 A new corpus for NLI

5.3.1 Formulating a task definition

Existing resources suffer from a subtler issue that impacts even projects using only human-provided annotations: indeterminacies of event and entity coreference lead to insurmountable indeterminacy concerning the correct semantic label (de Marneffe et al. 2008, Marelli et al. 2014b). For an example of the pitfalls surrounding entity coreference, consider the sentence pair

(5.1) PREMISE: A boat sank in the Pacific Ocean
HYPOTHESIS: A boat sank in the Atlantic Ocean

The pair could be labeled CONTRADICTION if one assumes that the two sentences refer to the same single event, but could also be reasonably labeled NEUTRAL if that assumption is not made. In order to ensure that the labeling scheme assigns a single correct label to every pair, it is necessary to select one of these approaches across the board, but both choices present problems. If we opt not to assume that events are coreferent, then we will only ever find contradictions between sentences that make broad universal assertions, but if we opt to assume coreference, new counterintuitive predictions emerge. For example, the below pair would unintuitively be labeled CONTRADICTION, rather than NEUTRAL, under this assumption, since the two sentences can't plausibly describe the same event.

(5.2) PREMISE: Ruth Bader Ginsburg was appointed to the US Supreme Court.
HYPOTHESIS: I had a sandwich for lunch today.

Entity coreference presents a similar kind of indeterminacy, as in the following pair.

(5.3) PREMISE: A tourist visited New York.
HYPOTHESIS: A tourist visited the city.

Assuming coreference between *New York* and *the city* justifies labeling the pair ENTAILMENT, but without that assumption *the city* could be taken to refer to a specific unknown city, leaving the pair NEUTRAL. This kind of indeterminacy of label can be resolved only once the questions of coreference are resolved.

Marelli et al. (2014b) observed a similar issue in constructing SICK and proposed a solution of roughly the same flavor as the one we propose below.

> Not unreasonably, subjects found that, say, *A woman is wearing an Egyptian headdress* does not contradict *A woman is wearing an Indian headdress*, since one could easily imagine both sentences truthfully uttered to refer to a single scene where two different women are wearing different headdresses. In the future, a higher proportion of CONTRADICTION labels could be elicited by using grammatical and possibly visual cues (pictures) encouraging co-indexing of the entities in the two sentences.

With SNLI, we sought to address the issues of size, quality, and indeterminacy. To do this, we employed a crowdsourcing framework with the following crucial innovations:

- The examples were grounded in specific scenarios, and the premise and hypothesis sentences in each example were constrained to describe that scenario from the same perspective, which helps greatly in controlling event and entity coreference.[2]

- The prompt gave participants the freedom to produce entirely novel sentences within the task setting, which led to richer examples than we see with the more proscribed string-editing techniques of earlier approaches, without sacrificing consistency.

- A subset of the resulting sentences were sent to a validation task aimed at providing a highly reliable set of annotations over the same data, and at identifying areas of inferential uncertainty.

5.3.2 Data collection

We used Amazon Mechanical Turk for data collection. In each individual task (each *HIT*, in Amazon's terms), a worker was presented with premise scene descriptions from a preexisting corpus, and asked to supply hypotheses for each of the three labels—ENTAILMENT, NEUTRAL, and CONTRADICTION—forcing the data to be balanced among these classes.

The instructions that we provided to the workers are shown in Figure 5.1. Below the instructions were three fields for each of three requested sentences, corresponding to the ENTAILMENT, NEUTRAL, and CONTRADICTION labels, a fourth field (marked optional) for reporting problems, and a link to an FAQ page. That FAQ was constructed gradually over the course of data collection. It warns about disallowed techniques (e.g. reusing the same sentence for many different prompts, which we observed in a few cases), provides guidance concerning sentence length and complexity (we did not enforce a minimum length, and we allowed bare NPs as well as full sentences), and reviews logistical issues around payment timing.

[2] Issues of coreference are not completely solved, but are greatly mitigated. For example, with the premise sentence *A dog is lying in the grass*, a worker could safely assume that the dog is the most prominent thing in the photo, and very likely the only dog, and build contradicting sentences assuming reference to the same dog.

We will show you the caption for a photo. We will not show you the photo. Using only the caption and what you know about the world:

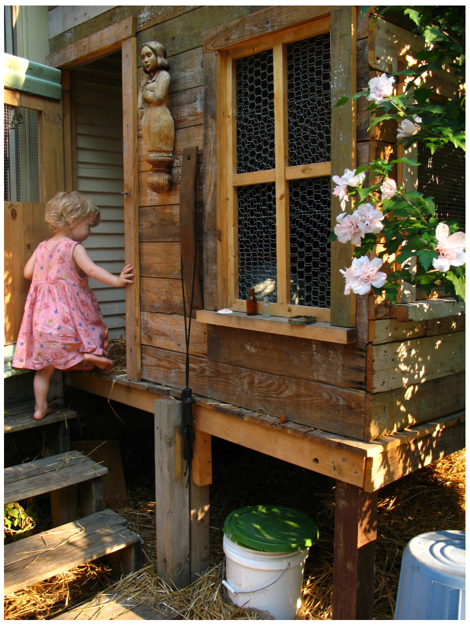
- Write one alternate caption that is **definitely** a **true** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "There are animals outdoors."*
- Write one alternate caption that **might be** a **true** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "Some puppies are running to catch a stick."*
- Write one alternate caption that is **definitely** a **false** description of the photo. *Example: For the caption "Two dogs are running through a field." you could write "The pets are sitting on a couch." This is different from the* maybe correct *category because it's impossible for the dogs to be both running and sitting.*

Figure 5.1: The instructions used on Mechanical Turk for data collection.

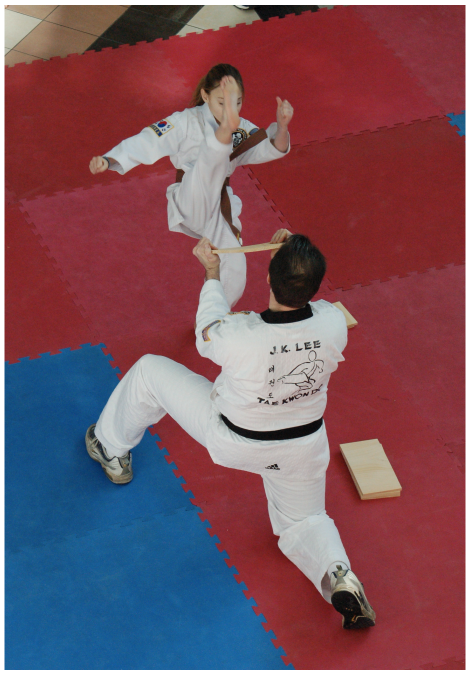
About 2,500 workers contributed. We adjusted the compensation strategy several times, but it averaged about \$0.16 per HIT. We could not accurately measure how long workers took to complete each HIT, but users on Mechanical Turk worker forums described both the compensation and the task completion experience in reliably positive terms.

For the premises, we use captions from the Flickr30K corpus (Young et al. 2014), a collection of approximately 160K captions (corresponding to about 30K images) collected in an earlier crowdsourced effort.[3] The images are sampled from personal photos posted to the sharing site Flickr, and cover a diverse range of scenes, with an emphasis on people and outdoor scenes. Figure 5.2 shows two examples. The captions were not authored by the photographers who took the source images, and they tend to contain relatively literal scene descriptions that are suited to this approach, rather than those typically associated with personal photographs (as in their example: *Our trip to the Olympic Peninsula*). In order to ensure that the label for each sentence

[3]We additionally include about 4K sentence pairs from a pilot study in which the premise sentences were instead drawn from the VisualGenome corpus (Krishna et al. 2016). These examples appear only in the training set, and have pair identifiers prefixed with `vg` in the corpus.



(a) Flickr photo available at `http://flickr.com/photos/studiobeerhorst/1000268201`



(b) Flickr photo available at `http://flickr.com/photos/scottfeldstein/101362133`

Figure 5.2: Two examples of images included in the Flickr30K corpus. Note that while these images were used to generate the Flickr30k sentences, which are used as premises in SNLI, our annotators did not see images when constructing or labeling hypotheses for SNLI. Both images are released under the Creative Commons CC BY 2.0 license: `http://creativecommons.org/licenses/by/2.0`

Dataset sizes:	
Training pairs	550,152
Development pairs	10,000
Test pairs	10,000
Sentence length:	
Premise mean token count	14.1
Hypothesis mean token count	8.3
Parser output:	
Premises with 'S' root tag	74.0%
Hypotheses with 'S' root tag	88.9%
S–S pairs	67.1%
NP–S pairs	21.2%
S–NP pairs	6.2%
NP–NP pairs	3.9%
Pairs with other root tags	1.6%
Distinct words (ignoring case)	37,026

Table 5.4: Key statistics for the raw sentence pairs in SNLI. Since the two halves of each pair were collected separately, we report some statistics for both.

pair can be recovered solely based on the available text, we do not use the images at all during corpus collection.

Table 5.4 reports some key statistics about the collected corpus, and Figure 5.3 shows the distributions of sentence lengths for both the source hypotheses and the newly collected premises. We observe that while premise sentences varied considerably in length, hypothesis sentences tended to be as short as possible while still providing enough information to yield a clear judgment, clustering at around seven words. We also observe that the bulk of the captions from both sources were syntactically complete sentences rather than bare noun phrase fragments, and the output of the parser attests to this.

5.3.3 Data validation

In order to measure the quality of the corpus, and in order to construct maximally useful testing and development sets, we perform an additional round of validation

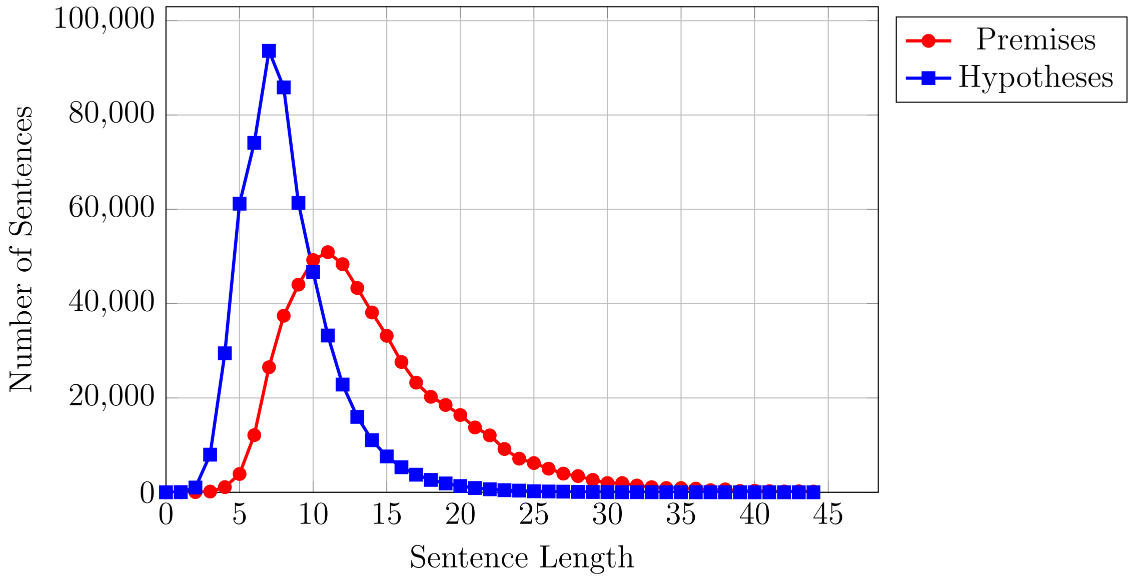


Figure 5.3: The distribution of sentence length.

for about 10% of the data. This validation phase follows the same basic form as the Mechanical Turk labeling task used to label the SICK entailment data: we present workers with pairs of sentences in batches of five, and ask them to choose a single label for each pair. We supply each pair to four annotators, yielding five labels per pair including the label used by the original author. The instructions are shown in Figure 5.4, and the annotator web interface includes a link to an FAQ similar to the one used during data collection. Though we initially used a very restrictive qualification (based on past approval rate) to select workers for the validation task, we nonetheless discovered (and deleted) some instances of random guessing in an early batch of work, and subsequently instituted a fully closed qualification restricted to about 30 trusted workers.

We assign a gold label to each validated pair. If any one of the three labels is chosen by at least three of the five annotators, it is used as the gold label. If there is no such consensus, which occurs in about 2% of cases, we assign the placeholder label '-'. While these unlabeled examples are included in the corpus distribution, they are unlikely to be helpful for the standard NLI classification task, and we do not include them in either training or evaluation in the experiments that I discuss in the following

The Stanford University NLP Group is collecting data for use in research on computer understanding of English. We appreciate your help!
Your job is to figure out, based on the correct caption for a photo, if another caption is also correct:

- Choose **definitely correct** if any photo that was captioned with the caption on the left would also fit the caption on the right. Example: "*A kitten with spots is playing with yarn.*"/"*A cat is playing.*"
- Choose **maybe correct** if the second caption could describe photos that fit the first caption, but could also describe sentences that don't fit the first caption. Example: "*A kitten with spots is playing with yarn.*"/"*kitten is playing with yarn on a sofa.*"
- Choose **definitely incorrect** if any photo that could possibly be captioned with the caption on the left would not fit the caption on the right. Example: "*A kitten with spots is playing with yarn.*"/"*A puppy is playing with yarn.*"

We have already labeled one out of every 250 HITs. Completing one of these HITs yields a bonus of $1 for each response that matches our label for up to $5.

Figure 5.4: The instructions used on Mechanical Turk for data validation.

General:	
Validated pairs	56,951
Pairs w/ unanimous gold label	58.3%
Individual annotator label agreement:	
Individual label = gold label	89.0%
Individual label = author's label	85.8%
Gold label–author's label agreement:	
Gold label = author's label	91.2%
Gold label ≠ author's label	6.8%
No gold label (no 3 labels match)	2.0%
Fleiss κ:	
CONTRADICTION	0.77
ENTAILMENT	0.72
NEUTRAL	0.60
Overall	0.70

Table 5.5: Statistics for the validated pairs. The AUTHOR'S LABEL is the label used by the worker who wrote the premise to create the sentence pair. A GOLD LABEL reflects a consensus of three votes from among the author and the four annotators.

two chapters.

The results of this validation process are summarized in Table 5.5. Nearly all of the examples receive a majority label, indicating broad consensus about the nature of the data and categories. The gold-labeled examples are very nearly evenly distributed across the three labels. The Fleiss κ scores (computed over every example with a full five annotations) are likely to be conservative given the large and unevenly distributed pool of annotators, but they still provide insights about the levels of disagreement across the three semantic classes. This disagreement likely reflects not just the limitations of large crowdsourcing efforts but also the uncertainty inherent in naturalistic NLI. Regardless, the overall rate of agreement is extremely high, suggesting that the corpus is sufficiently high quality to pose a challenging but realistic machine learning task.

5.3.4 The distributed corpus

Table 5.1 shows a set of randomly chosen validated examples from the development set with their labels. Qualitatively, we find the data draws fairly extensively on commonsense knowledge, and that hypothesis and premise sentences often differ structurally in significant ways, suggesting that there is room for improvement beyond superficial word alignment models. We also find the sentences that we collected to be largely fluent, correctly spelled English, with a mix of full sentences and caption-style noun phrase fragments, though punctuation and capitalization are often omitted.

The corpus is available under a CreativeCommons Attribution-ShareAlike license, the same license used for the Flickr30K source captions. It can be downloaded at: `http://nlp.stanford.edu/projects/snli/`

Partition We distribute the corpus with a prespecified train/test/development split. The test and development sets contain 10K examples each. Each original ImageFlickr caption occurs in only one of the three sets, and all of the examples in the test and development sets have been validated.

Parses The distributed corpus includes parses produced by the Stanford PCFG Parser 3.5.2 (Klein & Manning 2003), trained on the standard training set as well as on the Brown Corpus (Francis & Kucera 1979), which we find to improve the parse quality of the descriptive sentences and noun phrases found in the descriptions. Parses are available in both the standard PTB format and in the binarized unlabeled format used by tree-structured neural network models.

General evaluation standards While we hope that the corpus will be valuable in a variety of ways, we encourage researchers working on tools for semantic representation and inference to evaluate on the data in a uniform way: training on only the (parsed and/or unparsed) sentences included in the training set and doing final evaluations on only the subset of the test set for which there are single gold labels. Chapters 6 and 7 contain evaluations of this kind.

5.4 Understanding SNLI

In this section I offer examples and summary statistics that are meant to provide a clearer picture of the contents of the SNLI corpus and of the kinds of language understanding that it tests.

5.4.1 Common patterns and possible shortcuts

If SNLI is to accomplish its stated goals, it should be impossible for a model to do well at SNLI entailment classification without incorporating a fairly sophisticated approximation of human language understanding. This appears to be the case: I have been unable to identify any significant properties of the data that would allow a model to cheat by exploiting trivial correspondences between the contents of sentence pairs and their labels. This section discusses some key properties of the data that could potentially have yielded such undesirable regularities.

There is/are A handful of annotators chose to begin most or all of their ENTAILMENT hypotheses with *there is*, *there are*, or a contracted version of one of those two, followed by some entity from the premise, as here.

(5.4) PREMISE: A white duck is spreading its wings while sitting on the water.
HYPOTHESIS: **There is** an animal in the water.
LABEL: ENTAILMENT

This tendency in the data is noticeable, but neither frequent enough nor regular enough to provide a substantial aid in entailment classification. Only 3.8% of the test hypotheses follow this template, and of those only 64.4% are labeled ENTAILMENT.

Insertion of negation It would be possible for annotators to systematically create CONTRADICTION hypotheses by simply copying the premise and negating some part of it, as in this invented hypothetical example.

(5.5) PREMISE: A white duck is spreading its wings while sitting on the water.
HYPOTHESIS: A white duck is **not** spreading its wings while sitting on the

water.
LABEL: CONTRADICTION

This style of contradiction is reasonably well attested in SICK, and is easy to create, but we find it to be rare in SNLI. Only 1.4% of test examples contain inserted negation of any kind (tokens of *not* or *n't* that appear in the hypothesis but not the premise), and few of those follow this simple template. Only 45.4% of this broader set of inserted negation examples are labeled CONTRADICTION, and these often involve more complex alignments between premise and hypothesis than are seen in (5.5), as below.

(5.6) PREMISE: A young boy, wearing a jesters hat is enjoying himself sledding.
HYPOTHESIS: A young boy is just sitting down and **not** sledding.
LABEL: CONTRADICTION

Many more examples use inserted negation in a variety of other ways that correspond to all three possible labels, as in (5.7), making the presence of inserted negation only a very weak cue to the overall label.

(5.7) PREMISE: A shirtless man with cropped hair smokes a cigarette and ties a plastic bag.
HYPOTHESIS: The man is **not** wearing a shirt.
LABEL: ENTAILMENT

Pure insertion/deletion A small minority of the hypotheses in SNLI were created either by only deleting words from the premise as in (5.8), or by preserving the premise in its entirety and adding additional words as in (5.9).

(5.8) PREMISE: A man interviews a boy **in a gathering of people**.
HYPOTHESIS: A man interviews a boy.
LABEL: ENTAILMENT

(5.9) PREMISE: A soldier using binoculars in a desert.
HYPOTHESIS: A soldier using binoculars **searches for the enemy** in a

desert.
LABEL: NEUTRAL

If a model can reliably detect instances of pure insertion or deletion, it will benefit from that ability: most hypotheses built by pure deletion are entailments (since the truth conditions of the premise are likely being relaxed with the removal of content) and most hypotheses built by pure insertion are neutral. There are exceptions to both patterns, like the two examples below.

(5.10) PREMISE: **A** Man **is** eating **food next to a** child **on a bench**.
HYPOTHESIS: Man eating child.
LABEL: CONTRADICTION

(5.11) PREMISE: A woman wearing sunglasses.
HYPOTHESIS: A woman **is** wearing **a pair of** sunglasses.
LABEL: ENTAILMENT

Of the 2.5% of test examples built by pure deletion, 98% are labeled ENTAILMENT, and of the 0.6% of test examples built by pure insertion, 81% are labeled NEUTRAL.

Unrelated sentences Under SNLI's definition of CONTRADICTION, it is possible for annotators to create CONTRADICTION hypotheses by simply making up novel sentences that have nothing to do with the premise in either form or content, as here.

(5.12) PREMISE: A man is playing the saxophone in the street and some people are sitting on the curb next to the street by him.
HYPOTHESIS: A woman is flying crosscountry.
LABEL: CONTRADICTION

Examples like this are fairly rare. Out of a random sample of 100 test examples, I was able to identify three that contained hypotheses with arguably no relation to their premises. All three were labeled CONTRADICTION.

Typos and grammatical errors Typos and instances of non-standard grammar (relative to my intuitions as a first-language US English speaker) are relatively common, likely due to a combination of non-native English speaker annotators, annotators using nonstandard dialects of English, or simply fast or sloppy work. Out of the 100 example sample of the test set, I identified nine instances of non-standard grammar and three typos, with both phenomena relatively evenly distributed across the three label classes.

Both typos and grammatical variation will likely make it harder for machine learning systems to model the SNLI task, and typos in particular will pose a major obstacle to systems that are based on word-level features, including both bag-of-words features in conventional NLP classifier models and word-level embedding matrices in distributed representation models like the neural networks under study in this dissertation.

5.4.2 SNLI and natural logic

The types of inference—and the types of language understanding—that SNLI tests are strikingly different from the inferences studied in MacCartney-style natural logic: most rely on at least some amount of common sense or world knowledge, and few depend crucially on the kinds of lexical relationships and projectivity phenomena that natural logic is centered on. This observation need not reflect poorly on the value of either natural logic or on the value of the kinds of inferences captured by SNLI, but it is unproductive to discuss SNLI's place in the surrounding inference literature without keeping this stark difference in mind. This section looks at points of difference between natural logic and the style of inference captured by SNLI.

Lexical entailments Much of the machinery of natural logic is responsible for determining the relationship between lexical entailment relations and sentence-level entailment relations, and similar kinds of reasoning do appear in SNLI. While it is rare for a single lexical relation to determine a sentential relation in SNLI, it is fairly common for lexical relations between a premise word and a corresponding hypothesis word to play a key role in inference, as here.

(5.13) Premise: A woman leans over a small fence to take a picture of a **yellow** flower.
Hypothesis: The flower is **blue**
Label: contradiction

I was able to identify relevant lexical relations like this one in 28 out of 100 test examples, with these relations distributed fairly evenly across classes.

Monotonicity with quantifiers Monotonicity inferences under quantification are a central example of the kinds of inference that natural logic can handle effectively. Inferences of this kind play a subtle role in many SNLI examples, but sentences like those studied in Section 3.5 which directly foreground this kind of reasoning are rare. For example, of the 100 examples that I manually inspected, only four involved a substitution of one quantifier in the hypothesis for another in the premise, and all four had substantial additional complexity beyond the bare monotonicity inference, as here.

(5.14) Premise: **Three** people sit on a bench at a station, the man looks oddly at the two women, the redheaded women looks up and forward in an awkward position, and the yellow blond girl twiddles with her hair.
Hypothesis: **Some** people stand around.
Label: contradiction

SNLI and commonsense background knowledge

Natural logic—and logical approaches to NLI in general—tend to focus on recognizing as rich a set of inferences as possible following from a single premise or a small set of premises. Knowledge is generally introduced into the system through lexical entailment statements and projectivity rules (at least in the case of natural logic) rather than through additional premises. Since SNLI is built up of labeled sentence pairs, it superficially appears to have a similar kind of focus, highlighting the ability to do local inference without extensive background knowledge. However, the inferences in SNLI do appear to rely extensively on outside knowledge. This knowledge is unstated and takes the form of commonsense background knowledge, but it can often be effectively framed as sets of additional premises that need to be available to a model for the model to correctly interpret the inference, as in (5.15) below.

(5.15) a. PREMISE: Two children, both wearing tan coats, are embracing one another.

b. [COMMONSENSE PREMISE I:] If two people are embracing one another, they are facing one another.

c. [COMMONSENSE PREMISE II:] If someone is running, they are moving in the direction that they are facing.

d. [COMMONSENSE PREMISE III:] If two people are embracing one another, they cannot move in two different directions.

e. HYPOTHESIS: Two kids are running down a highway.

f. LABEL: CONTRADICTION

These commonsense premises are facts or tendencies about the world rather than facts about language, and they cannot be straightforwardly expressed as lexical entailments. These premises are also necessary. For example, if *embracing* were replaced by *holding hands*, then commonsense premise I would not apply, and the correct label would potentially change from CONTRADICTION to NEUTRAL. Out of the sample of 100 test examples, I was able to identify 49 that involved inferences from premises of this kind, distributed evenly across all three label classes.

It would be scientifically simplest to study the computational modeling of natural language understanding on its own, isolating it from the separate problem of modeling contingent facts about the world. However, if we are to be able to learn our models of language from naturally occurring text—a prerequisite for training large models like the neural networks studied here—we will be forced to grapple with the tight coupling between the way that language is used and the things that it is used to describe.

5.5 Conclusion

Natural languages are powerful vehicles for reasoning, and nearly all questions about meaningfulness in language can be reduced to questions of entailment and contradiction in context. This suggests that NLI is an ideal testing ground for theories of semantic representation, and that training for NLI tasks can provide rich domain-general semantic representations. To date, however, it has not been possible to fully realize this potential due to the limited nature of existing NLI resources. This chapter seeks to remedy this with a new, large-scale, naturalistic corpus of sentence pairs labeled for entailment, contradiction, and independence.

Chapter 6

Modeling natural language inference

This chapter presents work that was published as Bowman, Angeli, Potts & Manning (2015a). My coauthor Gabor Angeli is responsible for the evaluation of the existing baseline models presented in Section 6.2.1 and the design and evaluation of the feature-based classifier models presented in Section 6.2.2, and wrote the majority of those sections. The other two co-authors, my advisors, contributed primarily in an advisory role. Section 6.4.1 was composed for this dissertation, and is entirely my own.

6.1 Introduction

We aim for the Stanford Natural Language Inference (SNLI) corpus, introduced in Chapter 5, to serve as a vehicle for the training and evaluation of neural network models for NLI. In order for these evaluations to be informative, two things are necessary: a comparison with existing non-neural network models for NLI, and a set of strong neural network baselines. This chapter provides both.

In this chapter, we evaluate a variety of models for natural language inference, including rule-based systems, simple linear classifiers, and neural network-based models. We find that two models achieve comparable performance: a feature-rich classifier model and a neural network model centered around LSTM sentence encoders. We further evaluate the LSTM model by taking advantage of its ready support for transfer learning, and show that it can be adapted to an existing NLI challenge task, setting a new state of the art among neural network models. Finally, I survey work on SNLI that has been done subsequent to the release of SNLI and these baselines, and suggest that the corpus has already succeeded at spurring neural networks research on NLI.

6.2 Establishing baselines for SNLI

The most immediate application for the SNLI corpus is in developing models for the task of NLI. In particular, since it is dramatically larger than any existing corpus of comparable quality, we expect it to be suitable for training parameter-rich models like neural networks, which have not previously been competitive at this task. Our ability to evaluate standard classifier-based NLI models, however, is limited to those which are able to scale to SNLI's size without modification, so a more complete comparison of approaches will have to wait for future work. In this section, we explore the performance of three classes of models which scale readily:

- Models from a well-known NLI system, the Excitement Open Platform (Section 6.2.1).
- Variants of a strong but simple feature-based classifier model, which makes use of both unlexicalized and lexicalized features (Section 6.2.2).
- Distributed representation models, including a baseline model and neural network sequence models (Section 6.2.3).

6.2.1 Excitement Open Platform models

The first class of models is drawn from the Excitement Open Platform (EOP; Padó et al. 2015, Magnini et al. 2014). EOP is a tool for quickly developing NLI systems while sharing components such as common lexical resources and evaluation sets. We evaluate on two algorithms included in the distribution: a simple edit distance-based

System	**SNLI**	**SICK**	**RTE-3**
Edit Distance-Based	71.9	65.4	61.9
Classifier-Based	72.2	71.4	61.5
+ Lexical Resources	**75.0**	**78.8**	**63.6**

Table 6.1: Two-class test accuracy for two simple baseline systems included in the Excitement Open Platform, as well as a model making use of more sophisticated lexical resources.

algorithm and a classifier-based algorithm, the latter both in a bare form and augmented with EOP's full suite of lexical resources.[1]

Our primary goal with EOP is to better understand the difficulty of the task of classifying SNLI corpus inferences. We approach this by running the same system on several datasets: the SNLI test set, the SICK test set, and the standard RTE-3 test set (Giampiccolo et al. 2007). Each of the models is separately trained on the training set of each corpus. To best accommodate existing models, all evaluations in this section use two-class entailment. To convert three-class problems like SICK and SNLI to this setting, all instances of CONTRADICTION and UNKNOWN are converted to NON-ENTAILMENT. This yields a most-frequent-class baseline accuracy of 66% on SNLI, and 71% on SICK.

The edit distance algorithm tunes the weight of the three case-insensitive edit distance operations on the training set, after removing stop words. We also evaluate the base classifier-based system distributed with the platform, and a variant which uses information from WordNet (Miller 1995) and VerbOcean (Chklovski & Pantel 2004), as well as features based on tree patterns and dependency tree skeletons (Wang & Neumann 2007). The results for RTE-3 are taken from Magnini et al. (2014). The results of these experiments (Table 6.1) show that, at least for this limited family of two-class classification models, SNLI is comparably difficult to the smaller SICK corpus. No existing model finds any regularities in the data that would allow it to reach excellent performance.

[1]This subsection reflects work primarily done by my coauthor Gabor Angeli.

6.2.2 The lexicalized classifier

Unlike the RTE datasets, SNLI's size supports approaches which make use of rich lexicalized features. In this section, we evaluate a simple lexicalized classifier to explore the ability of non-specialized models to exploit these features in lieu of more involved language understanding. This classifier implements 6 feature types; 3 unlexicalized and 3 lexicalized:[2]

1. The BLEU score of the hypothesis with respect to the premise, using an n-gram length between 1 and 4.
2. The length difference between the hypothesis and the premise, as a real-valued feature.
3. The overlap between words in the premise and hypothesis, both as an absolute count and a percentage of possible overlap, and both over all words and over just nouns, verbs, adjectives, and adverbs.
4. An indicator for every unigram and bigram in the hypothesis.
5. Cross-unigrams: for every pair of words across the premise and hypothesis which share a POS tag, an indicator feature over the two words.
6. Cross-bigrams: for every pair of bigrams across the premise and hypothesis which share a POS tag on the second word, an indicator feature over the two bigrams.

We report results in Table 6.2, along with ablation results for models without the cross-bigram features and for models with no lexical features at all. The full model performs substantially better than any of the EOP models, reaching higher absolute accuracy on the harder three-class classification task. There is a substantial jump in accuracy from using lexicalized features, and another from using the very sparse cross-bigram features. The latter result suggests that there is value in letting the classifier automatically learn to recognize structures like explicit negations and adjective modification. A similar result was shown in Wang & Manning (2012) for bigram features in sentiment analysis.

[2]This subsection reflects work primarily done by my coauthor Gabor Angeli.

System	**SNLI**		**SICK**	
	Train	Test	Train	Test
Lexicalized	99.7	**78.2**	90.4	**77.8**
Unigrams Only	93.1	71.6	88.1	77.0
Unlexicalized	49.4	50.4	69.9	69.6

Table 6.2: Three-class accuracy, training on either SNLI or SICK, including models lacking cross-bigram features (Feature 6), and lacking all lexical features (Features 4–6). We report results both on the test set and the training set to judge overfitting.

It is surprising that the classifier performs as well as it does with only extremely limited access to information about word order and alignment. Although we expect that richer models would perform better, the results suggest that given enough data, cross bigrams with the noisy part-of-speech overlap constraint can produce a reasonably effective model.

6.2.3 Sentence-encoding models

SNLI is suitably large and diverse to allow for the training of neural network models that produce distributed representations of sentence meaning. In this section, we compare the performance of three such models on the corpus. As elsewhere in this dissertation, we use sentence encoding as an intermediate step in the NLI classification task: each model must produce a vector representation of each of the two sentences without using any context from the other sentence, and the two resulting vectors are then passed to a neural network classifier which predicts the label for the pair. This highlights the ability of these models to learn useful representations of meaning (which may be independently useful for subsequent tasks), at the cost of excluding from consideration possible strong neural models for NLI that directly compare the two inputs at the word or phrase level.

Our neural network classifier, depicted in Figure 6.1 (and largely identical to the one-layer model used in Section 3.4), is simply a stack of three 200D tanh layers, with the bottom layer taking the concatenated sentence representations as input and the

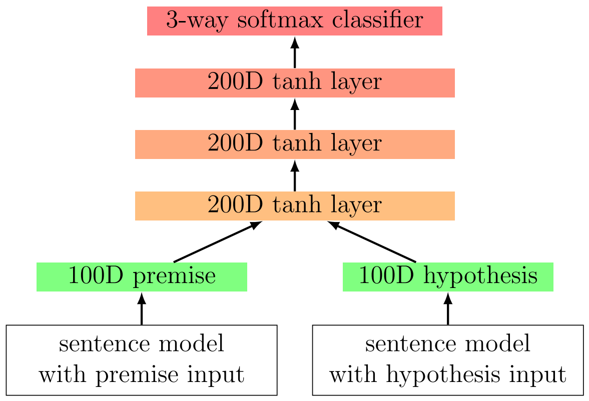


Figure 6.1: The neural network classification architecture: for each sentence-encoding model evaluated in Tables 6.3 and 6.4, two identical sentence encoders process the two input sentences and supply the 100D inputs shown here.

top layer feeding a softmax classifier, all trained jointly with the sentence-encoding model itself.

We test three sentence-encoding models, each set to use 100D phrase and sentence encodings. The baseline sum-of-words (a.k.a. continuous bag-of-words, or CBOW) sentence-encoding model simply sums the embeddings of the words in each sentence. In addition, we experiment with two simple sequence-based models: a plain RNN and an LSTM RNN. Efficiency considerations (discussed and addressed in Chapter 7) prevent the use of tree-structured models in this setting.

The word embeddings for all of the models are initialized with the 300D reference GloVe vectors (840B token version; Pennington et al. 2014) and fine-tuned as part of training. In addition, all of the models use an additional tanh neural network layer to map these 300D embeddings into the lower-dimensional phrase and sentence encoding space. All of the models are randomly initialized using standard techniques and trained using AdaDelta (Zeiler 2012) minibatch SGD until performance on the development set stops improving. We apply L2 regularization to all models, manually tuning the strength coefficient λ for each, and additionally apply dropout (Srivastava et al. 2014) to the inputs and outputs of the sentence-encoding models (though not to its internal connections) with a fixed dropout rate. All models are implemented in a common framework for this chapter.

Sentence model	**Train**	**Test**
100D Sum of words	79.3	75.3
100D RNN	73.1	72.2
100D LSTM RNN	84.8	**77.6**

Table 6.3: Accuracy in three-class classification on the SNLI training and test sets for each model.

The results are shown in Table 6.3. The sum-of-words model performs somewhat better than the fundamentally similar unigram lexicalized classifier, likely benefiting from its better ability to handle rare words through GloVe. However, it underperforms the full lexicalized classifier, likely since it lacks any access to word order. Of the two RNN models, the LSTM's more robust ability to learn long-term dependencies serves it well, giving it a substantial advantage over the plain RNN, and resulting in performance that is essentially equivalent to the lexicalized classifier on the test set (LSTM performance near the stopping iteration varies by up to 0.5% between evaluation steps). While the lexicalized model fits the training set almost perfectly, the gap between train and test set accuracy is relatively small for all three neural network models, suggesting that research into significantly higher capacity versions of these models would be productive.

6.2.4 Analysis and discussion

Figure 6.2 shows a learning curve for the LSTM and the lexicalized and unlexicalized feature-based models. It shows that the large size of the corpus is crucial to both the LSTM and the lexicalized model, and suggests that additional data would yield still better performance for both. In addition, though the LSTM and the lexicalized model show similar performance when trained on the current full corpus, the somewhat steeper slope for the LSTM towards the right of the figure hints that its ability to learn arbitrarily structured representations of sentence meaning may give it an advantage over the more constrained lexicalized model on still larger datasets.

We are also struck by the speed with which the lexicalized classifier outperforms

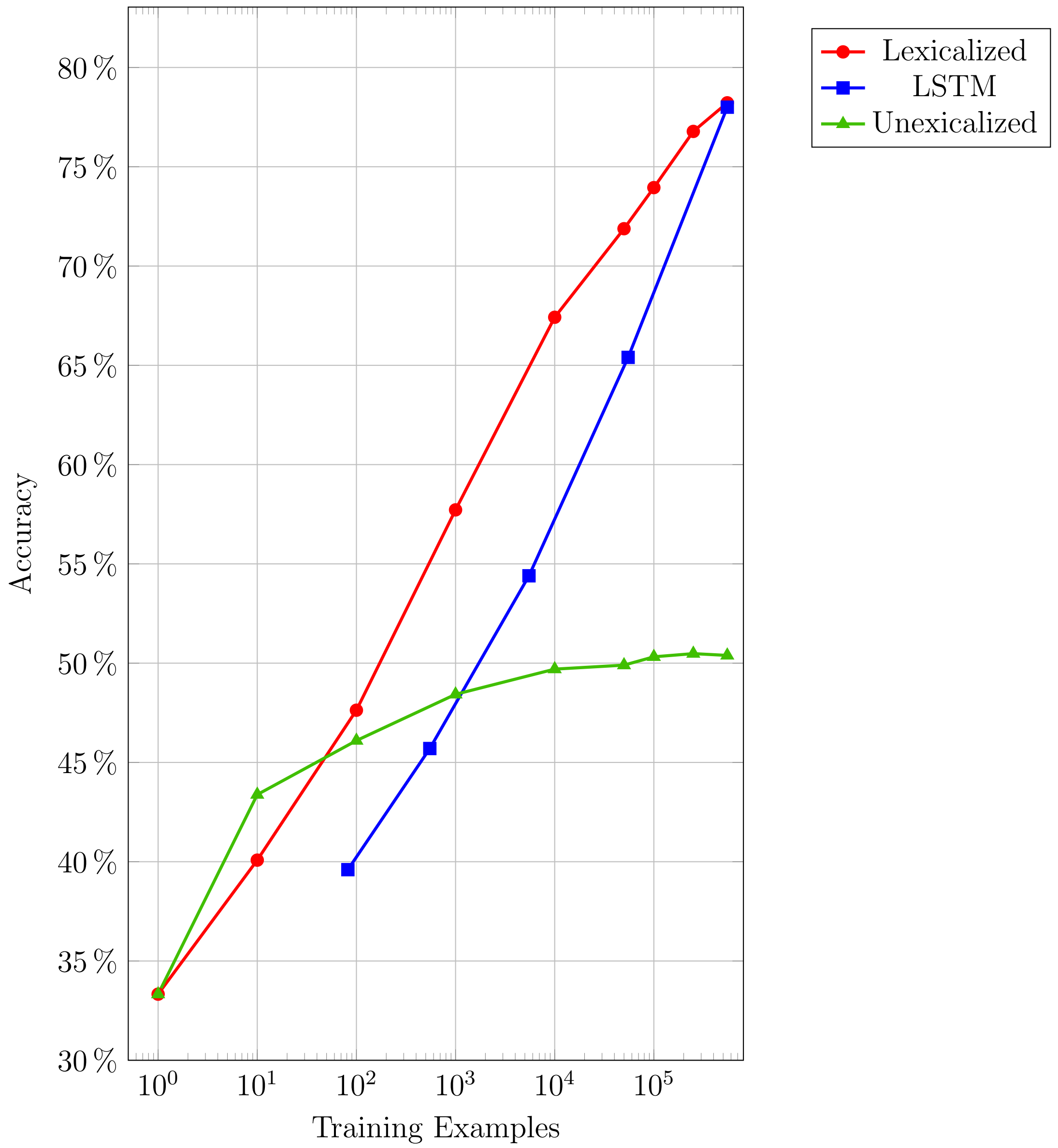


Figure 6.2: A learning curve showing how the baseline classifiers and the LSTM perform when trained to convergence on varied amounts of training data. The y-axis starts near a random-chance accuracy of 33%. The minibatch size of 64 that we used to tune the LSTM sets a lower bound on data for that model.

its unlexicalized counterpart. With only 100 training examples, the cross-bigram classifier already performs better. Empirically, we find that the top weighted features for the classifier trained on 100 examples tend to be high precision entailments; e.g. *playing* $\rightarrow$ *outside*, *a banana* $\rightarrow$ *person eating*. If relatively few spurious entailments get high weight—as it appears is the case—then it makes sense that, when prominent bigram features do fire, they boost accuracy in identifying entailments.

There are revealing patterns in the errors common to all the models considered here. Despite the large size of the training corpus and the distributional information captured by GloVe initialization, many lexical relationships are still misanalyzed, leading to incorrect predictions of NEUTRAL, even for pairs that are common in the training corpus like *beach/surf* and *sprinter/runner*. Semantic mistakes at the phrasal level, like the prediction of contradiction for

(6.1) PREMISE: A male is placing an order in a deli
HYPOTHESIS: A man buying a sandwich at a deli

indicate that additional attention to compositional semantics would pay off. However, many of the persistent problems run deeper, to inferences that depend on world knowledge and context-specific inferences, as in the entailment pair

(6.2) PREMISE: A race car driver leaps from a burning car
HYPOTHESIS: A race car driver escaping danger

for which both the lexicalized classifier and the LSTM predict NEUTRAL. In other cases, the models' attempts to shortcut this kind of inference through lexical cues can lead them astray. Some of these examples have qualities reminiscent of Winograd schemas (Winograd 1972, Levesque 2014). For example, all the models wrongly predict ENTAILMENT for

(6.3) PREMISE: A young girl throws sand toward the ocean
HYPOTHESIS: A girl can't stand the ocean

presumably because of distributional associations between *throws* and *can't stand*.

Analysis of the models' predictions also yields insights into the extent to which they grapple with event and entity coreference. For the most part, the original image prompts contained a focal element that the caption writer identified with a syntactic subject, following information structuring conventions associating subjects and topics in English (Ward & Birner 2004). The annotators generally followed suit, writing sentences that, while structurally diverse, share topic/focus (theme/rheme) structure with their premises. This promotes a coherent, situation-specific construal of each sentence pair. This is information that our models can easily take advantage of, but it can lead them astray. For instance, all of them stumble with the amusingly simple case

(6.4) PREMISE: A woman prepares ingredients for a bowl of soup
HYPOTHESIS: A soup bowl prepares a woman

in which prior expectations about parallelism are not met. Another headline example of this type is

(6.5) PREMISE: A man wearing padded arm protection is being bitten by a German shepherd dog
HYPOTHESIS: A man bit a dog

which all the models wrongly diagnose as ENTAILMENT, though the sentences report two very different stories. A model with access to explicit information about syntactic or semantic structure should perform better on cases like these.

The following chapter provides a more in-depth quantitative analysis of some of these patterns, both on models similar to these baselines and on newly proposed models.

6.3 Transfer learning with SICK

To the extent that successfully training a neural network model like the LSTM on SNLI forces that model to encode broadly accurate representations of English scene descriptions and to build an entailment classifier over those relations, we should expect it to be readily possible to adapt the trained model for use on other NLI tasks. In this section, we evaluate on the SICK entailment task using a simple transfer learning method (Pratt et al. 1991) and achieve competitive results.

Training sets	**Train**	**Test**
SNLI only	42.0	46.7
SICK only	100.0	71.3
SNLI and SICK (transfer)	99.9	**80.8**

Table 6.4: LSTM three-class accuracy on the SICK train and test sets under three training regimes.

To perform transfer, we take the parameters of the LSTM RNN model trained on SNLI and use them to initialize a new model, which is trained from that point only on the training portion of SICK. The only newly initialized parameters are softmax layer parameters and the embeddings for words that appear in SICK, but not in SNLI (which are populated with GloVe embeddings as above). We use the same model hyperparameters that were used to train the original model, with the exception of the L2 regularization strength, which is re-tuned. We additionally transfer the accumulators that are used by AdaDelta to set the learning rates. This lowers the starting learning rates, and is intended to ensure that the model does not learn too quickly in its first few epochs after transfer and destroy the knowledge accumulated in the pretransfer phase of training.

The results are shown in Table 6.4. Training on SICK alone yields poor performance, and the model trained on SNLI fails when tested on SICK data, labeling more NEUTRAL examples as CONTRADICTIONs than correctly, possibly as a result of subtle differences in label definitions between the two corpora. In contrast, transferring SNLI representations to SICK yields a new state of the art for an unaugmented neural network model, surpasses the available EOP models, and approaches both the non-neural network state of the art at 84.6% (Lai & Hockenmaier 2014) and the 84% level of interannotator agreement. This suggests that the introduction of a large high-quality corpus makes it possible to train representation-learning models for sentence meaning that are competitive with the best hand-engineered models on inference tasks.

We also attempted to apply this same transfer evaluation technique to the RTE-3 challenge, but found that the small training set (800 examples) did not allow the model to adapt to the unfamiliar genre of text used in that corpus, such that no training configuration yielded competitive performance. Subsequent research by Mou et al. (2016a) replicates the above result on SNLI–SICK transfer, but shows that no reasonably straightforward transfer mechanism is able to yield improvements on transfer from SNLI to the MSR Paraphrase corpus, which like RTE-3 is based on newswire text. This suggests that standard neural network models trained on SNLI are able to learn sentence representations that are effective for NLI over scene descriptions, but that these representations do not capture meaning in a general enough way to be effective on other tasks and text genres. Additional data collection or advances in semisupervised learning will likely be a necessary step in developing high-quality sentence-encoding models for new tasks and genres.

6.4 Discussion

In this chapter, we use SNLI to evaluate a range of models, and find that both simple lexicalized models and neural network models perform well, and that the representations learned by a neural network model on SNLI can be used to dramatically improve performance on a preexisting challenge dataset.

In the following chapter, I move from the evaluation of models based on existing sentence-encoding techniques to the development of new sentence-encoding models inspired by these results and those of earlier chapters.

6.4.1 Subsequent work

Since the release of SNLI and the publication of the experiments in this chapter, several outside research groups have used SNLI to evaluate novel neural network methods for language learning. This subsection discusses the highlights of these outside efforts.

Three of these papers are based on sentence-encoding techniques and follow the evaluation guidelines from Section 5.3.4:

- Vendrov et al. (2016) evaluate two different methods on SNLI. One is based on a pair of sequence models, as in this chapter, but uses GRU (Cho et al. 2014b) rather than LSTM encoders, and pretrains those encoders using the skip-thought objective (Kiros et al. 2015) rather than training them directly on SNLI. This unsupervised pretraining step allows them to effectively train large 1024D sequence models, and yields 81.4% test accuracy. Their second method builds on this by adding the use of their novel order-embeddings objective function, which encourages the model to learn sentence encodings that capture entailment information explicitly in the geometry of the embedding space. This objective is only compatible with two-class classification, but they outperform a two-class variant of their skip-thought baseline (87.7%) with this extended model, reaching 88.6% accuracy on this easier test configuration.

- Mou et al. (2016b) evaluate a novel tree-based convolutional neural network for sentence encoding, which encodes sentences by pooling information from a neural network filter layer that is applied to every subtree of each sentence. This allows them to reach 82.1% accuracy, though they do not compare with any baseline models implemented within their framework. This represents the first use of tree structure in neural network models for SNLI.

- In an as yet unpublished manuscript, Liu et al. (2016) evaluate a sentence-encoding model based on bidirectional LSTM RNNs augmented with within-sentence attention similar to that of the earlier Cheng et al. (2016) work discussed below. Unlike that earlier work, this model performs attention only within sentences, making it a pure sentence-encoding model. Their method is strikingly effective, yielding 84.2% test accuracy in a straightforward evaluation, and 85.0% with the addition of the collapsing-based preprocessing technique introduced in Section 5.2. The relationship between the information flow induced by tree-structured models and that induced by within-sentence attention models like this one remains a promising area for further research.

Four more papers have published results on SNLI using attention-based models which introduce dependencies between the model components that encode the two sentences:

- Rocktäschel et al. (2016) introduce the use of soft attention (Bahdanau et al. 2015) to sequence pair classification, encoding each sentence using an LSTM RNN and then using a novel attention component to build a new representation of the premise that captures information about its relationship with the hypothesis. This allows them to reach 83.5% accuracy.

- Wang & Jiang (2016) improve upon the model of Rocktäschel et al. (2016) by adding an additional separately parameterized LSTM that does not directly read either sentence, but rather accumulates the results of the attention process to incrementally construct a holistic representation of the sentence pair. This allows them to reach 86.1% accuracy.

- Cheng et al. (2016) introduce the novel LSTMN sentence encoder, which performs soft attention within the process of encoding single sentences. Adding this technique to soft attention between sentences, as in Wang & Jiang (2016), yields 86.3% accuracy.[3]

- Parikh et al. (2016) introduce a simpler attention-based model that does not rely on an RNN for preprocessing, and as such has limited access to word order information. That model alone reaches 86.3%, and adding a very weak word order signal through a form of intrasentence attention boosts performance to 86.9%.

Some additional recent work, concurrent with or subsequent to the creation of SNLI, has explored the use of learned representation models on the SICK entailment task. These results are consistent with the claim that SICK's small training set makes it relatively ineffective as an evaluation platform for low-bias machine learning models:

[3]The published version of that paper reports an additional result of 89.0% accuracy using what appear to be fairly standard methods on top of the base system, but the authors have privately informed me that the methods behind that particular result make it incomparable to other published results.

- Kruszewski et al. (2015) introduce a neural network with boolean activations that is meant to capture a similar intuition to the order-embeddings approach of Vendrov et al. (2016) described above. They evaluate this model on a number of tasks, including NLI with SICK, but they underperform even a simple most-frequent-class baseline model on this task.

- Pham et al. (2015) perform a similar multi-task evaluation that includes SICK as part of their evaluation of the unsupervised C-PHRASE compositional sentence-encoding model. They reach 75% accuracy on this task, exceeding simple baselines, but underperforming nearly all non-neural models, as well as the fully-supervised neural network model presented in Section 5.2.

- Yin et al. (2015) use SICK as part of a multi-task evaluation for their novel model which combines soft attention with convolutions. This model in conjunction with several novel techniques, including the addition of handcrafted features, a hybrid SVM-NN classifier, and an alignment-based preprocessing step similar to the one used in Section 5.2, reaches strong performance at 86.2%.

Chapter 7

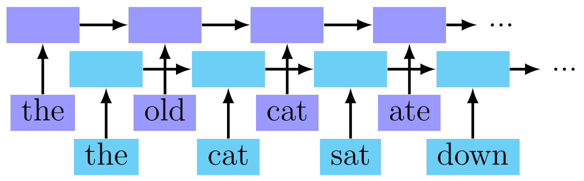
A fast unified parser-interpreter

This chapter presents work that was published as Bowman, Gauthier, Rastogi, Gupta, Manning & Potts (2016). This paper resulted from an active collaboration among the first four authors. Jon Gauthier and I were jointly responsible for the design of the novel models presented in this section. All four of us contributed substantially to the implementation and testing of these models, with Jon taking on nearly all of the optimization-related work discussed in Section 7.3.7 and in the development of Algorithm 1. Excepting those two areas, I composed the text of the paper. The last two co-authors, my advisors, contributed to this paper primarily in an advisory role. Sections 7.4.3 and 7.5 were composed exclusively for this dissertation, and are exclusively my work.

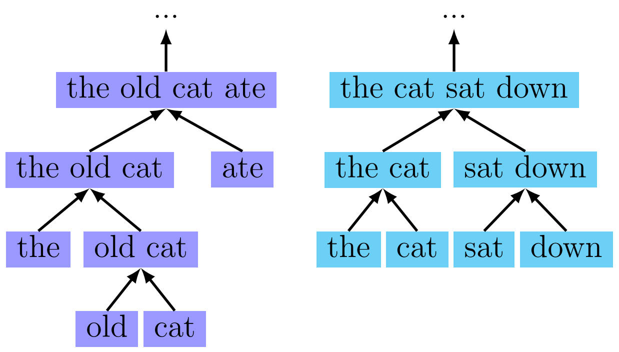
7.1 Introduction

Neural network sentence encoders often take the form of sequence-based recurrent neural network models (see Figure 7.1a and Section 2.3.1), which accumulate information over the sentence sequentially; convolutional neural networks (Kalchbrenner et al. 2014, Zhang et al. 2015), which accumulate information using filters over short local sequences of words or characters; and tree-structured recursive neural networks (see Figure 7.1b and 2.3.2), which propagate information up a binary parse tree.

Of these, the tree-structured approach appears to be the principled choice, since



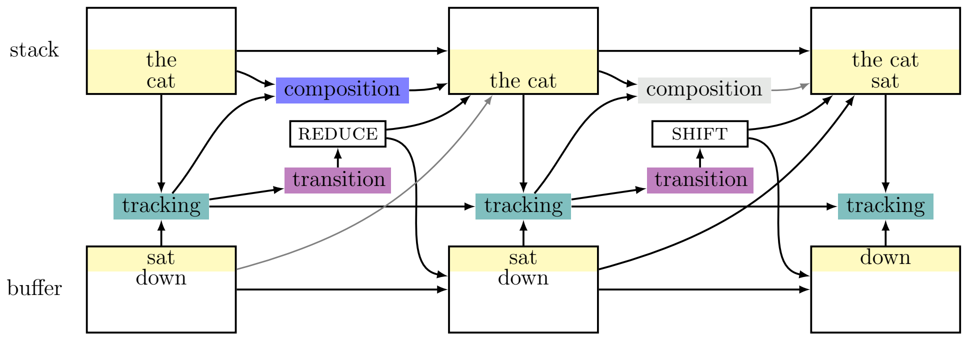
(a) A conventional sequence-based RNN for two sentences.



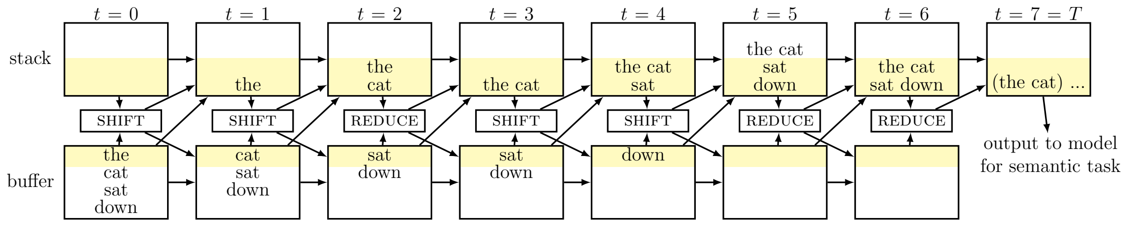
(b) A conventional TreeRNN for two sentences.

Figure 7.1: An illustration of two standard designs for sentence encoders. The Tree-RNN, unlike the sequence-based RNN, requires a substantially different connection structure for each sentence, making batched computation impractical.

meaning in natural language sentences is generally understood to be constructed recursively according to a tree structure. TreeRNNs have shown promise (Chapter 4; Tai et al. 2015, Li et al. 2015), but have largely been overlooked in favor of sequence-based RNNs because of their incompatibility with batched computation and their reliance on external parsers. Batched computation—performing synchronized computation across many examples at once—yields order-of-magnitude improvements in model run time, and is crucial in enabling neural networks to be trained efficiently on large datasets. Because TreeRNNs use a different model structure for each sentence, as in Figure 7.1, batching is impossible in standard implementations. In addition, standard TreeRNN models can only operate on sentences that have already been processed by a syntactic parser, which slows and complicates the use of these models at test time for most applications.



(a) The SPINN model unrolled for two transitions during the processing of the sentence *the cat sat down*. *Tracking*, *transition*, and *composition* are neural network layers. Gray arrows indicate connections which are blocked by a gating function.



(b) The fully unrolled SPINN for *the cat sat down*, with neural network layers omitted for clarity.

Figure 7.2: Two views of SPINN.

This chapter introduces a new model to address both these issues: the Stack-augmented Parser-Interpreter Neural Network, or SPINN, which is shown in Figure 7.2. SPINN executes the computations of a tree-structured model in a linearized sequence, and can incorporate a neural network parser that produces the required parse structure on the fly. This design improves upon the TreeRNN architecture in three ways:

- At test time, it can simultaneously parse and interpret unparsed sentences, removing the dependence on an external parser at nearly no additional computational cost.

- It supports batched computation for both parsed and unparsed sentences, yielding dramatic speedups over standard TreeRNNs.

- It supports a novel tree–sequence hybrid architecture for handling local linear context in sentence interpretation. This model is a basically plausible model of human sentence processing and yields substantial accuracy gains over pure sequence- or tree-based models.

We evaluate SPINN primarily on the Stanford Natural Language Inference entailment task (see Chapter 5), and find that it significantly outperforms preexisting sentence-encoding-based models, and that it yields speed increases of up to 25$\times$ over a standard TreeRNN implementation.

7.2 Related work

There is a fairly long history of work on building neural network-based parsers that use the core operations and data structures from transition-based parsing, which SPINN builds on (Henderson 2004, Emami & Jelinek 2005, Titov & Henderson 2010, Chen & Manning 2014, Buys & Blunsom 2015, Dyer et al. 2015, Kiperwasser & Goldberg 2016). In addition, there has been recent work proposing models designed primarily for generative language modeling tasks that use this architecture as well (Zhang et al. 2016, Dyer et al. 2016). To the best of our knowledge, SPINN is the first model to use this architecture for the purpose of sentence interpretation, rather than parsing or generation.

Socher et al. (2011a,b) present versions of the TreeRNN model which are capable of operating over unparsed inputs. However, these methods require an expensive search process at test time. SPINN represents a fast alternative approach.

7.3 The new model: SPINN

7.3.1 Background: Shift-reduce parsing

SPINN is inspired by the SHIFT-REDUCE PARSING formalism (Aho & Ullman 1972), a transition-based parsing strategy which builds a tree structure over a sequence (e.g. a natural language sentence) by a single left-to-right scan over its tokens. The formalism is widely used in natural language parsing (e.g. Shieber 1983, Nivre 2003).

A shift-reduce parser accepts a sequence of input tokens $\mathbf{x} = (x_0, \ldots, x_{N-1})$ and consumes transitions $\mathbf{a} = (a_0, \ldots, a_{T-1})$, where each $a^{(t)} \in \{\textsc{shift}, \textsc{reduce}\}$ specifies one step of the parsing process. In general a parser may also generate these transitions on the fly as it reads the tokens. It proceeds left to right through a transition sequence, combining the input tokens in $\mathbf{x}$ incrementally into a tree structure. For any binary-branching tree structure with N words, this requires $T = 2N - 1$ transitions through a total of $T + 1$ states.

The parser uses two auxiliary data structures: a stack S of partially completed subtrees and a buffer B of tokens yet to be parsed. The parser is initialized with the stack empty and the buffer containing the tokens $\mathbf{x}$ of the sentence in order. Let $\langle S, B \rangle = \langle \emptyset, \mathbf{x} \rangle$ denote this starting state. It next proceeds through the transition sequence, where each transition $a^{(t)}$ selects one of the two following operations. Below, the $|$ symbol denotes the CONS (concatenation) operator. We arbitrarily choose to always CONS on the left in the notation below.

SHIFT: $\langle S, x \mid B \rangle \rightarrow \langle x \mid S, B \rangle$. This operation pops an element from the buffer and pushes it onto the top of the stack.

REDUCE: $\langle x \mid y \mid S, B \rangle \rightarrow \langle (x, y) \mid S, B \rangle$. This operation pops the top two elements from the stack, merges them, and pushes the result back onto the stack.

7.3.2 Composition and representation

SPINN is based on a shift-reduce parser, but it is designed to produce a vector representation of a sentence as its output, rather than a tree as in standard shift-reduce parsing. It modifies the shift-reduce formalism by using fixed length vectors to represent each entry in the stack and the buffer. Correspondingly, its REDUCE operation combines two vector representations from the stack into another vector using a neural network function.

The composition function When a REDUCE operation is performed, the vector representations of two tree nodes are popped off of the stack and fed into a COMPOSITION FUNCTION, which is a neural network function that produces a representation for a new tree node that is the parent of the two popped nodes. This new node is pushed on to the stack.

We use a version of the TreeLSTM layer function for this purpose. In particular, we use the following formulation, modified from the version in (2.10–2.12) with the addition of an extra input $\vec{e}$:

$$\begin{bmatrix} \vec{i} \\ \vec{f_l} \\ \vec{f_r} \\ \vec{o} \\ \vec{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W_{\text{composition}} \begin{bmatrix} \vec{h}^{(1)}_{\text{stack}} \\ \vec{h}^{(2)}_{\text{stack}} \\ \vec{e} \end{bmatrix} + \vec{b}_{\text{composition}} \right) \tag{7.1}$$

$$\vec{c} = \vec{f_l} \odot \vec{c}^{(2)}_{\text{stack}} + \vec{f_r} \odot \vec{c}^{(1)}_{\text{stack}} + \vec{i} \odot \vec{g} \tag{7.2}$$

$$\vec{h} = \vec{o} \odot \vec{c} \tag{7.3}$$

where σ is the sigmoid activation function, $\odot$ is the elementwise product, the pairs $\langle \vec{h}^{(1)}_{\text{stack}}, \vec{c}^{(1)}_{\text{stack}} \rangle$ and $\langle \vec{h}^{(2)}_{\text{stack}}, \vec{c}^{(2)}_{\text{stack}} \rangle$ are the two input tree nodes popped off the stack, and $\vec{e}$ is an optional vector-valued input argument which is either empty or comes from from an external source like the tracking LSTM (see Section 7.3.3 below). The result of this function, the pair $\langle \vec{h}, \vec{c} \rangle$, is placed back on the stack. Each vector-valued variable listed is of dimension D except $\vec{e}$, of the independent dimension D_{tracking}.

The stack and buffer The stack and the buffer are arrays of N elements each (for sentences of up to N words), with two D-dimensional vectors $\vec{h}$ and $\vec{c}$ in each element.

Word representations We use word representations based on the 300D vector package provided with GloVe (840B token version; Pennington et al. 2014). We do not update these representations during training. Instead, we use a learned linear transformation to map each input word vector $\vec{x}$ into a vector pair $\langle \vec{h}, \vec{c} \rangle$ that is stored in the buffer:

$$\begin{bmatrix} \vec{h} \\ \vec{c} \end{bmatrix} = W_{\text{word}} \vec{x} + \vec{b}_{\text{word}} \tag{7.4}$$

7.3.3 The tracking LSTM

In addition to the stack, the buffer, and the composition function, the full SPINN model includes an additional component: the tracking LSTM. This is a simple low-dimensional sequence-based LSTM RNN that operates in tandem with the model, taking inputs from the buffer and stack at each step. It is meant to maintain a low-resolution summary of the portion of the sentence that has been processed so far, which is used for two purposes: it supplies feature representations to the transition classifier, which allows the model to stand alone as a parser, and it additionally supplies a secondary input $\vec{e}$ to the composition function—see 7.1—allowing context information to enter the construction of sentence meaning and forming what is effectively a tree–sequence hybrid model.

The tracking LSTM's inputs (yellow in Figure 7.2) are the top element of the buffer $\vec{h}_b^{(1)}$ (which would be moved in a SHIFT operation) and the top two elements of the stack $\vec{h}_{\text{stack}}^{(1)}$ and $\vec{h}_{\text{stack}}^{(2)}$ (which would be composed in a REDUCE operation).

Why a tree–sequence hybrid? Lexical ambiguity is ubiquitous in natural language. Most words have multiple senses or meanings, and it is generally necessary to use the context in which a word occurs to determine which of its senses or meanings is meant in a given sentence. Even though TreeRNNs are much more effective at composing meanings in principle, this ambiguity can give simpler sequence-based sentence-encoding models an advantage: when a sequence-based model first processes a word, it has direct access to a state vector that summarizes the left context of that word, which acts as a cue for disambiguation. In contrast, when a standard tree-structured model first processes a word, it only has access to the constituent that the word is merging with, which is often just a single additional word. Feeding a context representation from the tracking LSTM into the composition function is a simple and efficient way to mitigate this disadvantage of tree-structured models.

It would be straightforward to augment SPINN to support the use of some amount of right-side context as well, but this would add complexity to the model that we argue is largely unnecessary: humans are very effective at understanding the beginnings of sentences before having seen or heard the ends, suggesting that it is possible to get by without the unavailable right-side context.

7.3.4 Parsing: Predicting transitions

For SPINN to operate on unparsed inputs, it needs to be able to produce its own transition sequence **a** rather than relying on an external parser to supply it as part of the input. To do this, the model predicts the probabilities $\vec{p}^{(t)}$ of each possible action $a^{(t)}$ at each step using a simple two-class softmax classifier whose input is the state of the tracking LSTM:

$$\vec{p}^{(t)} = \text{softmax}(W_{\text{transition}}\vec{h}^{(\text{t})}_{\text{tracking}} + \vec{b}_{\text{transition}}) \tag{7.5}$$

The above approach is nearly the simplest viable transition decision function. In contrast, the decision functions in state-of-the-art transition-based parsers tend to use significantly richer feature sets as inputs, including features containing information about several upcoming words on the buffer. The value $\vec{h}_{\text{tracking}}$ is a function of only the very top of the buffer and the top two stack elements at each timestep.

At test time, the model uses whichever transition (i.e. SHIFT or REDUCE) is assigned a higher probability. The prediction function is trained to mimic the decisions of an external parser, and these decisions are used as inputs to the model during training. We use binary parse trees from the Stanford PCFG Parser. We did not find scheduled sampling (Bengio et al. 2015)—allowing the model to use its own transition decisions in some instances at training time—to help.

t	$S[t]$	$Q^{(t)}$	$a^{(t)}$
0		$\underline{\quad}$	SHIFT
1	*Spot*	$\underline{\ \ 1}$	SHIFT
2	*sat*	$\underline{1\ 2}$	SHIFT
3	*down*	$1\ \underline{2\ 3}$	REDUCE
4	(*sat down*)	$\underline{1\ 4}$	REDUCE
5	(*Spot* (*sat down*))	$\underline{\ \ 5}$	

Table 7.1: The thin-stack algorithm operating on the input sequence $\mathbf{x} = (\textit{Spot}, \textit{sat}, \textit{down})$ and the transition sequence shown in the last column. S is shown in the second column and represents the top of the stack at each step t. The last two elements of Q (underlined) specify which rows t would be involved in a REDUCE operation at the next step.

7.3.5 Implementation issues

Representing the stack efficiently A naïve implementation of SPINN needs to handle a size $O(N)$ stack at each timestep, any element of which may be involved in later computations. A naïve backpropagation implementation would then require storing each of the $O(N)$ stacks for a backward pass, leading to a per-example space requirement of $O(NTD)$ floats. This requirement is prohibitively large for significant batch sizes or sentence lengths N. Such a naïve implementation would also require copying a largely unchanged stack at each timestep, since each SHIFT or REDUCE operation writes only one new representation to the top of the stack.

We propose[1] a space-efficient stack representation inspired by the zipper technique (Huet 1997) that we call THIN STACK. For each input sentence, the stack is represented as a single $T \times D$ matrix S. Each row $S[t]$ (for $0 < t \leq T$) represents the top of the stack at timestep t. At each timestep the model can SHIFT a new element onto the stack, or REDUCE the top two elements of the stack into a single element. To shift an element from the buffer to the top of the stack at timestep t, it is simply written into the location $S[t]$. In order to perform the REDUCE operation, the top two elements of the actual stack are retrieved and used. The model maintains a queue Q of pointers into S which contains the row indices of S which are still present in the

[1]This method reflects work primarily done by my coauthor Jon Gauthier.

Algorithm 1 The thin stack algorithm

```
1: function STEP(bufferTop, op, $t$, $S$, $Q$)
2:     if op = SHIFT then
3:         $S[t]$ := bufferTop
4:     else if op = REDUCE then
5:         right := $S[Q.\text{pop}()]$
6:         left := $S[Q.\text{pop}()]$
7:         $S[t]$ := COMPOSE(left, right)
8:     $Q$.push($t$)
```

actual stack. The top two elements of the stack can be found by using the final two pointers in the queue Q. These retrieved elements are used to perform the REDUCE operation, which modifies Q to mark that some rows of S have now been replaced in the actual stack. Algorithm 1 describes the full mechanics of a stack feedforward in this compressed representation. It operates on the single $T \times D$ matrix S and a backpointer queue Q. Table 7.1 shows an example run.

This stack representation requires substantially less space. It stores each element involved in the feedforward computation exactly once, meaning that this representation can still support efficient backpropagation. Furthermore, all of the updates to S and Q can be performed batched and in-place on a GPU, yielding substantial speed gains over both a more naïve SPINN implementation and a standard TreeRNN implementation. We describe speed results in Section 7.3.7.

Preparing the data At training time, SPINN requires both a transition sequence $\mathbf{a}$ and a token sequence $\mathbf{x}$ as its inputs for each sentence. The token sequence is simply the words in the sentence in order. $\mathbf{a}$ can be obtained from any constituency parse for the sentence by first converting that parse into an unlabeled binary parse, then linearizing it (with the usual in-order traversal), then taking each word token as a SHIFT transition and each ‘)’ as a REDUCE transition, as here:

Unlabeled binary parse: ((the cat) (sat down))

x: *the*, *cat*, *sat*, *down*

a: SHIFT, SHIFT, REDUCE, SHIFT, SHIFT, REDUCE, REDUCE

Handling variable sentence lengths For any sentence model to be trained with batched computation, it is necessary to pad or crop sentences to a fixed length. We fix this length at $N = 25$ words, longer than about 98% of sentences in SNLI. Transition sequences $\mathbf{a}$ are cropped at the left or padded at the left with SHIFTs. Token sequences $\mathbf{x}$ are then cropped or padded with empty tokens at the left to match the number of SHIFTs added or removed from $\mathbf{a}$, and can then be padded with empty tokens at the right to meet the desired length N.

7.3.6 TreeRNN equivalence

Without the addition of the tracking LSTM, SPINN (in particular the SPINN-PI-NT variant, for *parsed input, no tracking*) is precisely equivalent to a conventional tree-structured neural network model in the function that it computes, and therefore also has the same learning dynamics. In both, the representation of each sentence consists of the representations of the words combined recursively using a TreeRNN composition function (here, the TreeLSTM function). SPINN, however, is dramatically faster, and supports both integrated parsing and a novel approach to context through the tracking LSTM.

7.3.7 Inference speed

In this section, we compare the test time speed of the SPINN-PI-NT with an equivalent TreeRNN implemented in the conventional fashion and with a standard RNN sequence model. While the full models evaluated below are implemented and trained using Theano (Theano Development Team 2016), which is reasonably efficient but not perfect for this model, we wish to compare well-optimized implementations of all three models. To do this, we reimplement the feedforward[2] of SPINN-PI-NT and an LSTM RNN baseline in C++/CUDA, and compare that implementation with a

[2] We chose to reimplement and evaluate only the feedforward/inference pass, as inference speed is the relevant performance metric for most practical applications.

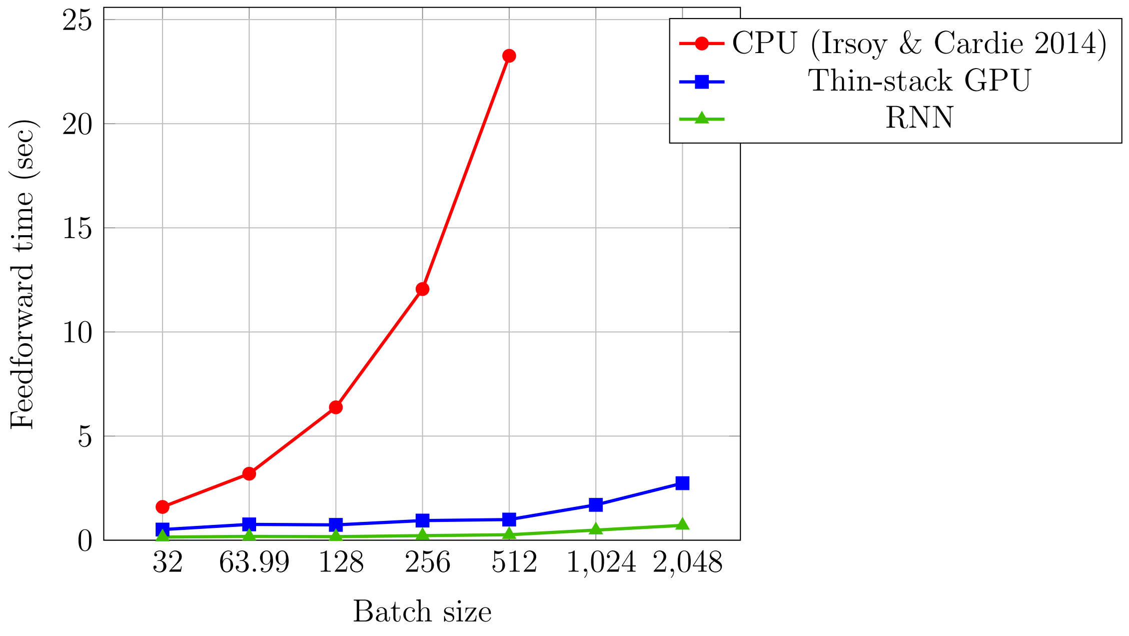


Figure 7.3: Feedforward speed comparison.

CPU-based C++/Eigen TreeRNN implementation from Irsoy & Cardie (2014), which we modified to perform exactly the same computations as SPINN-PI-NT.[3] TreeRNNs like this can only operate on a single example at a time and are thus poorly suited for GPU computation.

Each model is restricted to run on sentences of 30 tokens or fewer. We fix the model dimension D and the word embedding dimension at 300. We run the CPU performance test on a 2.20 GHz 16-core Intel Xeon E5-2660 processor with hyper-threading enabled. We test the thin-stack implementation and the RNN model on an NVIDIA Titan X GPU.

Figure 7.3 compares the sentence-encoding speed of the three models on random input data. We observe a substantial difference in runtime between the CPU and thin-stack implementations that increases with batch size. With a large but practical batch size of 512, the largest on which we tested the TreeRNN, SPINN is about 25× faster than the standard CPU implementation, and about 4× slower than the RNN

[3]The original code for Irsoy & Cardie's model is available at `https://github.com/oir/deep-recursive`. Optimized C++/CUDA models and the Theano source code for the full SPINN are available at `https://github.com/stanfordnlp/spinn`.

baseline.

Though this experiment only covers SPINN-PI-NT, the results should be similar for the full SPINN model: most of the computation involved in running SPINN is involved in populating the buffer, applying the composition function, and manipulating the buffer and the stack, with the low-dimensional tracking and parsing components adding only a small additional load.

7.4 NLI Experiments

We evaluate SPINN on the task of natural language inference using SNLI, and use FraCaS as a supplementary test set to better understand the results.

Creating a sentence-pair classifier To classify a sentence pair, we run two copies of SPINN with shared parameters: one on the premise sentence and another on the hypothesis sentence. We then use their outputs (the $\vec{h}$ states at the top of each stack at time $t = T$) to construct a feature vector $\vec{x}_{\text{classifier}}$ for the pair. This feature vector consists of the concatenation of these two sentence vectors, their difference, and their elementwise product (following Mou et al. 2016b):

$$\vec{x}_{\text{classifier}} = \begin{bmatrix} \vec{h}_{\text{premise}} \\ \vec{h}_{\text{hypothesis}} \\ \vec{h}_{\text{premise}} - \vec{h}_{\text{hypothesis}} \\ \vec{h}_{\text{premise}} \odot \vec{h}_{\text{hypothesis}} \end{bmatrix} \tag{7.6}$$

Following the general approach presented in Section 2.3.4, this feature vector is then passed to a series of 1024D ReLU neural network layers (i.e. an MLP; the number of layers is tuned as a hyperparameter), then passed into a linear transformation, and then finally passed to a softmax layer, which yields a distribution over the three labels.

The objective function The objective combines a cross-entropy objective $\mathcal{L}_{\text{s}}$ for the NLI classification task, cross-entropy objectives $\mathcal{L}_{\text{p}}^{t}$ and $\mathcal{L}_{\text{h}}^{t}$ for the parsing decision for each of the two sentences at each step t, and an L2 regularization term on the trained parameters. The terms are weighted using the tuned hyperparameters α and λ:

$$\mathcal{L}_{\mathrm{m}} = \mathcal{L}_{\mathrm{s}} + \alpha \sum_{t=0}^{T-1} (\mathcal{L}_{\mathrm{p}}^{t} + \mathcal{L}_{\mathrm{h}}^{t}) + \lambda \|\theta\|_2^2 \tag{7.7}$$

Initialization, optimization, and tuning We initialize the model parameters using the nonparametric strategy of He et al. (2015), with the exception of the softmax classifier parameters, which we initialize using random uniform samples from $[-0.005, 0.005]$.

We use minibatch SGD with the RMSProp optimizer (Tieleman & Hinton 2012) and a tuned starting learning rate that decays by a factor of 0.75 every 10K steps. We apply both dropout (Srivastava et al. 2014) and batch normalization (Ioffe & Szegedy 2015) to the output of the word embedding projection layer and to the feature vectors that serve as the inputs and outputs to the MLP that precedes the final entailment classifier.

We train each model for 250K steps in each run, using a batch size of 32. We track each model's performance on the development set during training and save parameters when this performance reaches a new peak. We use early stopping, evaluating on the test set using the parameters that perform best on the development set.

We use random search to tune the hyperparameters of the model, setting the ranges for search for each hyperparameter heuristically (and validating the reasonableness of the ranges on the development set), and then launching eight copies of each experiment each with newly sampled hyperparameters from those ranges. Table 7.2 shows the hyperparameters used in the best run of each model.

7.4.1 Models evaluated

We evaluate five models. The five all use the sentence-pair classifier architecture described in Section 7.4, and differ only in the function computing the sentence encodings. First, a sum-of-words model and a single-layer LSTM RNN (built on the one

Param.	Range	Samp.	Sum	RNN	PI-NT	PI	SP
Init. LR	2×10^{-4}–2×10^{-2}	LOG	8×10^{-3}	5×10^{-3}	3×10^{-4}	7×10^{-3}	2×10^{-3}
λ	1×10^{-7}–3×10^{-5}	LOG	1×10^{-7}	4×10^{-6}	3×10^{-6}	2×10^{-5}	3×10^{-5}
α	0.5–4.0	LIN	—	—	—	—	3.9
ET dr.	80–95%	LIN	—	—	83%	92%	86%
MLP dr.	80–95%	LIN	88%	94%	94%	93%	94%
D_{tracking}	24–128	LOG	—	—	—	61	79
MLP lrs.	1–3	LIN	2	2	2	2	1

Table 7.2: Hyperparameter ranges and values. *Range* shows the hyperparameter ranges explored during random search. *Samp.* indicates whether sampling from the range was uniform (LIN), or log-uniform (LOG). *Init. LR* is the starting learning rate for RMSProp. λ is the L2 regularization weight. α is the transition cost scaling weight. *ET dr.* is the dropout keep rate for the embedding transformation layer. *MLP dr.* is the dropout keep rate for the classifier MLP. D_{tracking} is the size of the tracking LSTM state. *MLP lrs.* is the number of layers in the classifier MLP.

presented in Chapter 6) serve as baseline encoders. Next, the minimal SPINN-PI-NT model (equivalent to a TreeLSTM) introduces the SPINN model design. SPINN-PI adds the tracking LSTM to that design. Finally, the full SPINN adds the integrated parser.

We compare these models against several baselines, including the strongest published non-neural network-based result from Chapter 6 and previous neural network models built around several types of sentence encoders.

7.4.2 Results

Table 7.3 shows our results on SNLI. For the full SPINN, we also report a measure of agreement between this model's parses and the parses included with SNLI, calculated as classification accuracy over transitions averaged across time steps.

We find that the bare SPINN-PI-NT model performs little better than the RNN baseline, but that SPINN-PI with the added tracking LSTM performs well, surpassing all prior work on sentence encoding (and, as of writing, all subsequent such work but the unpublished Liu et al. 2016). The success of SPINN-PI, which is a hybrid tree–sequence model, suggests that the tree- and sequence-based encoding methods are at

Model	Params.	Trans. (%)	Train (%)	Test (%)
Previous non-NN results				
Lexicalized classifier (Chapter 6)	—	—	99.7	78.2
Previous sentence encoder-based NN results				
100D LSTM encoders (Chapter 6)	220K	—	84.8	77.6
1024D pretrained GRU encoders[1]	15M	—	98.8	81.4
300D Tree-based CNN encoders[2]	3.5M	—	83.4	82.1
New results				
300D sum-of-words encoders	2.3M	—	77.8	77.2
300D LSTM RNN encoders	3.0M	—	83.9	80.6
300D SPINN-PI-NT encoders	3.4M	—	84.4	80.9
300D SPINN-PI encoders	3.7M	—	89.2	**83.2**
300D SPINN encoders	2.7M	92.4	87.2	82.6

Table 7.3: Results on SNLI 3-way inference classification. *Params.* is the approximate number of trained parameters (excluding word embeddings for all models). *Trans. acc.* is the model's accuracy in predicting parsing transitions at test time. *Train* and *test* are SNLI classification accuracy. External results are from [1]Vendrov et al. (2016) and [2]Mou et al. (2016b).

least partially complementary. The full SPINN model with its relatively weak internal parser performs slightly less well, but nonetheless robustly exceeds the performance of the RNN baseline. The sum-of-words model, while better than the smaller such model tested in Chapter 6, lags behind considerably.

Both SPINN-PI and the full SPINN significantly outperform all previous sentence-encoding models. Most notably, these models outperform the tree-based CNN of Mou et al. (2016b), which also uses tree-structured composition for local feature extraction, but uses simpler pooling techniques to build sentence features in the interest of efficiency. The results show that a model that uses tree-structured composition fully (SPINN) outperforms one which uses it only partially (tree-based CNN), which in turn outperforms one which does not use it at all (RNN).

The full SPINN performed moderately well at reproducing the Stanford Parser's parses of the SNLI data at a transition-by-transition level, with 92.4% accuracy at test time. However, its transition prediction errors were fairly evenly distributed across sentences, and most sentences were assigned partially invalid transition sequences that

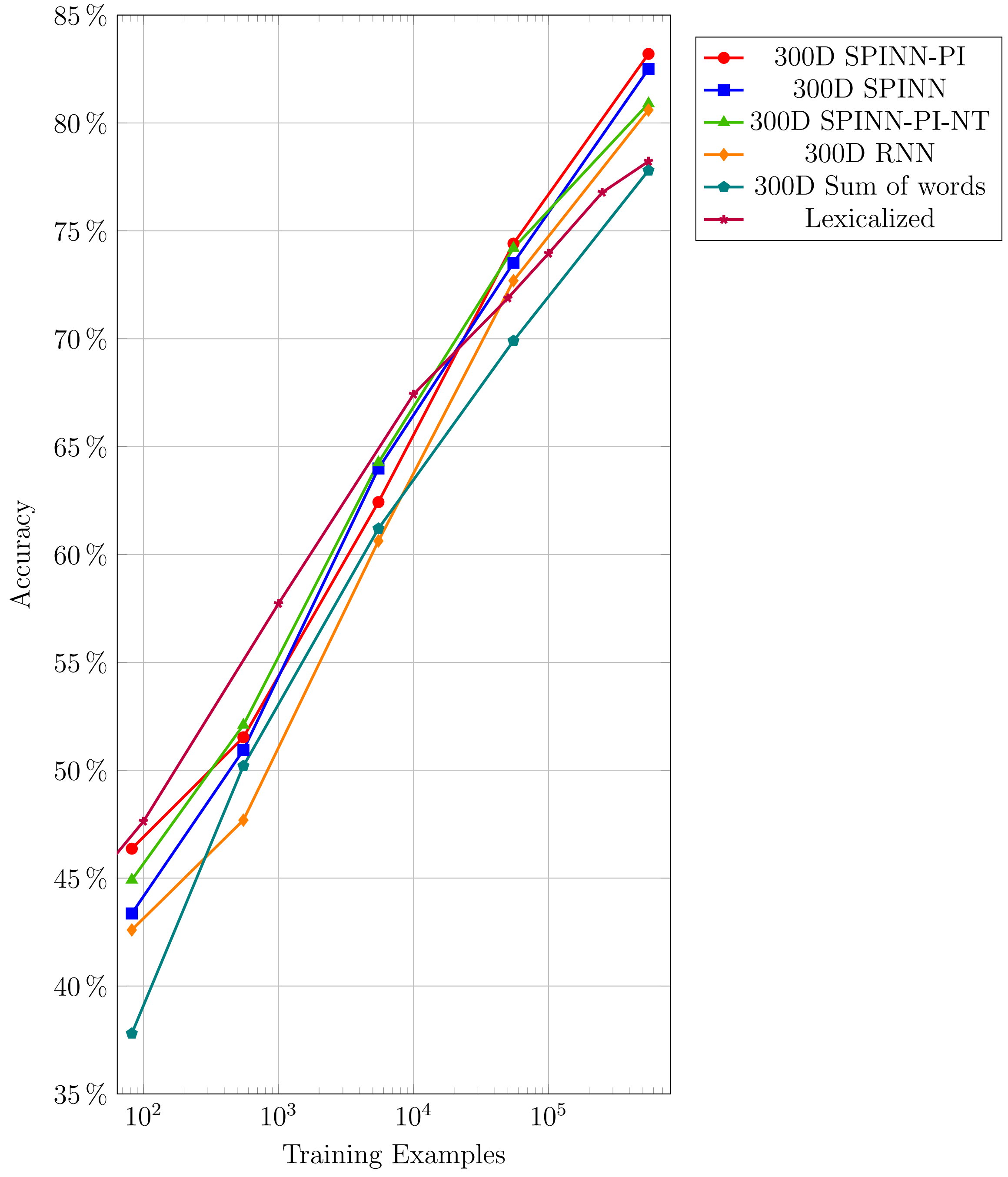


Figure 7.4: Test set performance for the models discussed in this chapter and the lexicalized classifier baseline from Chapter 6, shown at high resolution to highlight small differences.

either left a few words out of the final representation or incorporated a few padding tokens into the final representation.

Figure 7.4 shows how model performance changes as the models are trained on decreasing amounts of training data. For the experiments reflected in this learning curve, the tuned hyperparameter values are held constant, and all models are tested on the full SNLI test set. We observe that all models improve with increasing amounts of training data, and that none is close to saturation at the 550K-example training set size that the full SNLI corpus provides. We also observe that the RNN baseline performs somewhat worse than the richer models when less training data is available, with its performance only beginning to match that of the tree-structured SPINN-PI-NT at the full 550K examples.

7.4.3 Testing on FraCaS

Although the style of inferences that SNLI focuses on is quite different from the style of inferences studied in most previous work on natural logic, it is possible that the models under study in this chapter may have nonetheless learned some approximation of the latter, more directly logic-based, style of inference. To investigate that, I evaluate these models on the FraCas test suite (Cooper et al. 1996), allowing for a direct comparison with the NatLog implementation of natural logic (MacCartney 2009), which was developed and tested primarily on this corpus.

FraCas is intended only as a test set, and its small size leaves no room to set aside examples for training or tuning models. As such, I use models trained on SNLI with no further training or fine tuning on the target domain. This is a necessary compromise when working with FraCas with data-intensive models, but it does put these models at a disadvantage, both because of the mismatch in style between the corpora and because the definition of the output labels (ENTAILMENT, CONTRADICTION, and NEUTRAL) do not align exactly across the two.

The original release of FraCas is targeted at question-answering systems. For this evaluation, I use the modified version prepared by MacCartney (2009) which converts the question–answer pairs to three-label NLI problems. Like MacCartney, I discard

Section 1: Quantifiers

(7.8) PREMISE: No delegate finished the report.
HYPOTHESIS: Some delegate finished the report on time.
LABEL: CONTRADICTION

Section 2: Plurals

(7.9) PREMISE: Either Smith, Jones or Anderson signed the contract.
HYPOTHESIS: Jones signed the contract.
LABEL: NEUTRAL

Section 3: (Nominal) Anaphora

(7.10) PREMISE: John said Bill had hurt himself.
HYPOTHESIS: Someone said John had been hurt.
LABEL: NEUTRAL

Figure 7.5: Examples from FraCas. Note that (7.10) is labeled NEUTRAL, even though a natural extension of the SNLI label definitions to sentences like these would suggest a more likely label of CONTRADICTION.

multiple-premise examples and the handful of examples which are not assigned to one of the three standard labels.

FraCas is divided into nine sections. Each is meant to cover one narrow set of phenomena in NLI using a handful of expect-constructed inference problems over a formal, business-oriented register of English. Figure 7.5 shows examples from the first three sections of the corpus.

Table 7.4 shows the results of this evaluation, presented alongside results with NatLog from MacCartney (2009). No system performs well in absolute terms, though NatLog performs dramatically better overall, with the neural network models lagging behind by a gap of greater than 10% absolute. Looking at individual sections, NatLog performs best in most, including all of the sections that MacCartney marked as highlighting the strengths of his instantiation of natural logic, while neural models

Section	**#**	**Sum**	**LSTM**	**PI-NT**	**PI**	**SP**	**NatLog**
1. Quantifiers*	44	57%	64%	61%	64%	66%	**98%**
2. Plurals*	24	50%	54%	67%	46%	42%	**75%**
3. (Nominal) Anaphora	6	**83%**	67%	67%	67%	**83%**	50%
4. Ellipsis	25	**68%**	**64%**	**64%**	48%	44%	24%
5. Adjectives*	15	40%	47%	33%	40%	40%	**80%**
6. Comparatives*	16	63%	56%	31%	38%	56%	**81%**
7. Temporal Reference	36	44%	44%	**56%**	53%	39%	**58%**
8. Verbs	8	**63%**	**63%**	38%	**63%**	**63%**	**63%**
9. Attitudes*	9	67%	67%	44%	33%	44%	**89%**
All Sections	183	55%	57%	55%	51%	51%	**71%**

Table 7.4: Results on the FraCas test suite with the five models discussed above (*PI-NT*=SPINN-PI-NT, *PI*=SPINN-PI, *SP*=SPINN) and with MacCartney's (2009) NatLog. MacCartney noted that Sections 1, 2, 5, 6, and 9 (marked with *) highlight the strengths of his natural logic.

perform as well or better at *Anaphora*, *Ellipsis*, *Temporal Reference*, and *Verbs*. Performance in individual sections, though, provides only a very weak signal to model quality and is easily misinterpreted. Sections generally contain variants of only a few sentence types, leading to the potential for extremely high variance in model performance. It is striking and somewhat mysterious that performance on FraCas test data by the five neural models appears to be weakly *inversely* correlated with performance on the in-domain SNLI test set by the same models.

These negative results should not be surprising in light of what we know about both SNLI and neural network models in general. While an ideal NLI model would be able to recognize inferences of a broad range of styles across a broad range of genres of text, SNLI is not designed to be the sole training corpus for such a model. It is restricted to the genre of captions and to the kinds of inferences encouraged by the task framing presented to the annotators (Figure 5.1). In addition, an empirical evaluation has shown that standard cross-domain transfer techniques for neural networks, which would be needed for success here, are not especially effective for sentence-encoding models like these (Mou et al. 2016a). In an effort to better understand the ability of the SPINN models (and the LSTM RNN) to effectively model its target domain, the following section presents an in-depth analysis of the performance of these models on SNLI.

7.4.4 Analysis and discussion

The use of tree structure improves the performance of sentence-encoding models for SNLI. I suspect that this improvement is largely due to the more efficient learning of accurate generalizations overall, and not to any particular few phenomena. However, some patterns are identifiable in the results.

While all five models under study have trouble with negation, the tree-structured SPINN models do quite substantially better on these pairs. This is likely due to the fact that parse trees make the scope of any instance of negation (the portion of the sentence's content that is negated) relatively easy to identify and separate from the rest of the sentence. For test set sentence pairs like the one below where negation (*not* or *n't*) does not appear in the premise but does appear in the hypothesis, the RNN shows 67% accuracy, while all three tree-structured models exceed 73%. Only the RNN and sum-of-words models got the below example wrong.

(7.11) PREMISE: A man dressed in a light blue shirt dumping items from a bin into another bin, while standing in a room full of food donations.
HYPOTHESIS: Foods are not stored in room by a man.
LABEL: CONTRADICTION

Note that the presence of negation in the hypothesis is correlated with a label of CONTRADICTION in SNLI, but not as strongly as one might intuit: only 45% of these examples in the test set are labeled as contradictions.

In addition, it seems that tree-structured models, and especially the tree–sequence hybrid models, are more effective than RNNs at extracting informative representations of long sentences. The RNN model falls off in test accuracy more quickly with increasing sentence length than SPINN-PI-NT, which in turn falls off substantially faster than the two hybrid models, repeating a pattern seen more dramatically on artificial data in Chapter 4. On pairs with premises of 20 or more words, the RNN reaches 76.7% accuracy, while SPINN-PI reaches 80.2%. All three SPINN models labeled the following example correctly, while the RNN and sum-of-words models did not.

(7.12) PREMISE: A man wearing glasses and a ragged costume is playing a Jaguar electric guitar and singing with the accompaniment of a drummer.
HYPOTHESIS: Two men are playing on a street corner.
LABEL: NEUTRAL

I suspect that the hybrid nature of the full SPINN model is also responsible for its surprising ability to perform better than an RNN baseline even when its internal parser is relatively ineffective at producing correct full-sentence parses. It may act somewhat like the tree-based CNN, only with access to larger trees: using tree structure to build up local phrase meanings, and then using the tracking LSTM, at least in part, to combine those meanings.

Finally, as is likely inevitable for models evaluated on SNLI, all five models under study did several percent worse on test examples whose ground truth label is NEUTRAL than on examples of the other two classes. ENTAILMENT–NEUTRAL and NEUTRAL–CONTRADICTION confusions appear to be much harder to avoid than ENTAILMENT–CONTRADICTION confusions, where relatively superficial cues might be more readily useful.

The following few sections provide some more detailed quantitative analysis to elaborate upon the above impressions.

Commonsense premises Table 7.5 shows the performance of the five models under study on the 49 examples (from the sample of 100 test examples introduced in Section 5.4.2) that I manually tagged as requiring some form of commonsense background knowledge. Learning a reasonable model of common sense over the domain of situations described in Flickr30K captions is part of the challenge of learning to model SNLI, and these examples highlight the ability of each model to do this. However, this challenge is largely orthogonal to the primary goal of learning a high-quality model of sentence meaning. All five models under study do dramatically worse on these examples than on the test set overall, reflecting the fact that none of them has access

Test set	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
Common sense	49	59%	65%	63%	67%	69%

Table 7.5: Results on manually tagged SNLI test examples that require commonsense background knowledge.

Test set	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
Lexical	28	57%	64%	68%	71%	64%

Table 7.6: Results on manually tagged SNLI test examples that require lexical relation knowledge.

to a high-quality source of commonsense background knowledge, nor any especially effective mechanism for extracting such knowledge from either GloVe or the SNLI training set. Integrating knowledge of this kind into models remains a substantial open problem in NLI and in NLP more broadly.

Lexical relations Table 7.6 shows the performance of the five models under study on the 28 examples that I manually tagged as crucially requiring knowledge of a specific lexical relation. All five models do substantially worse on these examples than they do on the test set overall. While Section 3.3 shows that similar neural network models can learn to encode a rich set of lexical relations effectively, it appears that the models under study in this chapter do not have access to high-quality information about lexical entailment and exclusion. Unlike in the artificial language experiments, Zipf's law ensures that any natural language test set will contain many word pairs that are unattested or only rarely attested in the training set. Because of that, the models here are forced to rely on the information that is present in GloVe rather than simply relying on its own learned lexical representations. While it is possible to extract a good deal of information about lexical entailment from distributional word vectors like GloVe (Kotlerman et al. 2010, Shwartz et al. 2016), this information appears to be incomplete and difficult to use, and existing systems have not been able to approach the near-perfect precision that would be necessary to make SNLI examples like these easy.

Premise len.	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
0–10	2610	80.2%	83.1%	82.8%	**84.8%**	83.3%
11–14	2798	78.6%	81.8%	81.5%	**83.5%**	82.8%
15–19	2096	76.9%	80.4%	81.0%	**84.3%**	82.7%
20+	2320	72.5%	76.7%	77.9%	**80.2%**	78.3%

Table 7.7: Results on SNLI test examples by premise length.

Root tags	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
NP–NP	378	81.0%	83.6%	82.8%	**85.7%**	84.4%
S–NP & NP–S	2762	79.1%	81.4%	82.1%	**84.8%**	82.9%
S–S	6684	76.2%	80.1%	80.3%	**82.5%**	81.3%

Table 7.8: Results on SNLI test examples by root tags.

Length As introduced above, the SPINN models appear to be somewhat better than the baseline models on long sentences. Table 7.7 demonstrates this, dividing the SNLI test set into four bins by premise length (rather than hypothesis length, which has a much lower variance).

S vs. NP Table 7.8 shows the performance of the five models under study on three classes of test set example: sentence–sentence pairs, sentence–noun phrase pairs, and noun phrase–noun phrase pairs. Both the RNN and the syntax-sensitive SPINN models have an easier time with the relatively simpler (but still non-trivial) structures of bare noun phrases, even despite their relatively low frequency in the corpus. The advantage of the hybrid SPINN models over the RNN persists even on these examples.

Negation Table 7.8 shows the performance of the five models under study on examples where *not* or *n't* is present in the hypothesis but not the premise. These relatively difficult examples, discussed in Section 5.4, highlight the models' ability to reason effectively with negation. As reported above, all five models struggle with these examples, but the RNN has a substantially harder time than the three tree-based SPINN models.

Test set	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
Negation	141	64%	67%	**74%**	**74%**	**75%**

Table 7.9: Results on SNLI test examples with negation (*not*, *n't*) in the hypothesis but not the premise.

Test set	**#**	**Sum**	**LSTM**	**SPINN-PI-NT**	**SPINN-PI**	**SPINN**
Deletion only	246	92%	95%	95%	95%	93%
Insertion only	57	88%	88%	89%	89%	91%

Table 7.10: Results on SNLI test examples where the word types in the hypothesis are a strict subset (deletion only) or a strict superset (insertion only) of the words types in the premise.

Pure insertion and pure deletion examples As I observe in Section 5.4, sentences where the hypothesis differs from the premise either only through deletions or only through insertions are among the easiest examples in the corpus, with the former consisting almost entirely of entailments, and the latter largely of neutral examples. Table 7.10 shows that all five models are able to take advantage of this property, and do substantially better on these examples than on the test set as a whole.

7.5 Conclusion and future work

In this chapter, we introduce a model architecture (SPINN-PI-NT) that is equivalent to a TreeLSTM, but an order of magnitude faster at test time. We expand that architecture into a tree–sequence hybrid model (SPINN-PI), and show that this yields significant gains on the SNLI entailment task. Finally, we show that it is possible to exploit the strengths of this model without the need for an external parser by integrating a fast parser into the model (as in the full SPINN), and that the lack of external parse information yields little loss in accuracy.

Because this dissertation is focused on general-purpose models for sentence encoding, I do not pursue the use of soft attention (Bahdanau et al. 2015, Rocktäschel et al. 2016), despite its demonstrated effectiveness on the SNLI task.[4] However, I expect that it should be possible to productively combine SPINN with soft attention to reach state-of-the-art performance.

The tracking LSTM uses only simple, quick-to-compute features drawn from the head of the buffer and the head of the stack. It is plausible that giving the tracking LSTM access to more information from the buffer and stack at each step would allow it to better represent the context at each tree node, yielding both better parsing and better sentence encoding. One promising way to pursue this goal would be to encode the full contents of the stack and buffer at each time step following the method used by Dyer et al. (2015).

Removing the parsing objective For a more ambitious goal, I expect that it should be possible to implement a variant of SPINN that learns to parse using guidance from the semantic representation objective, essentially allowing it to learn to produce parses that are, in aggregate, better suited to supporting semantic interpretation than those supplied in the training data. This has the promise both to yield a sentence-encoding model that is more effective than any presently available, and in addition to yield potentially powerful new way of testing hypotheses about the role of syntax in human language understanding.

Learning to parse from a purely semantic objective is not possible in these models as currently implemented, at least using standard gradient-based learning as we do here. This is because the decision to SHIFT or REDUCE at any timestep is a hard decision, leaving no way for gradient information to pass back through this decision to inform the transition classifier that it would have been better off had it made a different decision. There are two ways around this obstacle: reinforcement learning and the use of a differentiable (or SOFT) stack.

Reinforcement learning would have the model explicitly explore multiple different parses at training time, and to learn to choose parses that tend to result in correct semantic classifications. This approach is likely to be slow and unstable, but progress

[4]Attention-based models like Rocktäschel et al. (2016), Wang & Jiang (2016), and the unpublished Cheng et al. (2016) have shown accuracies as high as 86.3% on SNLI, but are more narrowly engineered to suit the task and do not yield sentence encodings.

in reinforcement learning with neural network models has been rapid in recent years (e.g. Mnih et al. 2015), and it should be possible to take advantage of much of this progress.

A differentiable stack would directly remove the source of the problem, the hard SHIFT/REDUCE decision. With a differentiable stack, the model would be able to perform an operation that is a blend of 91% SHIFT and 9% reduce at training time, allowing the gradient to reflect the relative quality of choosing either option. It is not obvious a priori that simply additively combining the results of a SHIFT operation and a REDUCE operation would lead to usable representations and reasonable classification performance, but Joulin & Mikolov (2015) have shown that a strategy like this can be used effectively on small toy problems. In addition, Grefenstette et al. (2015) have proposed a novel soft stack design that is explicitly designed to produce usable representations in gradient based learning.

Chapter 8

Conclusion

This dissertation presents three major contributions to the applications of neural network models to sentence-level language understanding. Chapters 3 and 4 show that existing neural network architectures are capable of learning to perform complex symbolic reasoning in controlled settings, despite their continuous representations and weak prior knowledge. Chapters 5 and 6 present SNLI, a corpus targeted at training and evaluating sentence-understanding models that is by far the largest of its kind and has become a major benchmark for research in this area. Chapter 7 introduces SPINN, an extension of the recursive neural network architecture that both improves its strength as a learner and makes it practical for use on large-scale language learning tasks.

8.1 The contributions of this dissertation

Chapter 1 introduces the relational view of semantics, both in the context of semantic theory and in the context of natural language inference and applied computational semantics. It then goes on to discuss the relationship between this view and issues of semantic representation, and contrasts that view with the prevailing truth-conditional approach to natural language meaning. With that context in place, it offers a summary of the goals of this dissertation.

Chapter 2 provides the technical foundations for the dissertation, laying out the fundamentals of natural logic, neural networks, and sentence-encoding models, and discussing prior work at the intersection of logic, formal semantics, and distributed representation modeling.

Chapter 3 uses artificial language learning experiments to investigate a fundamental tension that arises when using neural networks to model natural language semantics: while neural network models can only learn effectively when using fully continuous distributed representations, we know that language has a rich discrete symbolic structure that is difficult to capture precisely in a continuous setting. The experiments in this chapter show that this is not an obstacle in practice, at least in the domain of NLI: existing neural network models are capable of learning to perform several of the kinds of symbolic reasoning that are necessary for NLI, including learning a structured lexicon, reasoning with recursive grammars, and reasoning with quantification and monotonicity.

Chapter 4 extends these artificial language methods to address a narrower but still pressing question within the application of neural networks to natural language semantics: what is the value, if any, of integrating knowledge of the syntactic structure of natural languages into the architectures of models? I find that while models that do not incorporate tree structure show some basic ability to learn languages with a known syntactic structure, tree-structured models are dramatically more effective, suggesting that tree structure should be valuable in building sentence understanding models for natural language.

Chapter 5 presents SNLI, a human-annotated corpus for NLI. At 570K examples, the corpus is nearly two orders of magnitude larger than the largest preexisting human-annotated NLI corpus, making it uniquely suited for the training and evaluation of low-bias machine learning models like neural networks.

Chapter 6 uses both novel and existing baseline models to situate SNLI as an evaluation dataset among other NLI corpora. Additionally, it shows for the first time that a simple sequence-based neural network model can be trained to reach strong performance on NLI. In the year since the release of the corpus and these results, SNLI has quickly become a major testing ground for neural network models of sentence understanding.

Finally, Chapter 7 introduces SPINN, a novel model architecture for sentence encoding which builds on the lessons of Chapters Chapter 3 and Chapter 4. The SPINN model draws on ideas from shift-reduce parsing to implement the core computations of a tree-structured neural network within a radically different model structure. This improves on that existing architecture in three ways. First, it makes it possible to train a tree-structured model using batched computation, yielding order-of-magnitude speed increases and making it possible for the first time to train these models on large-scale tasks like SNLI. In addition, it allows for the ready integration of local context information into the process of semantic composition, allowing the model to produce sentence representations that surpass the prior state of the art on SNLI. Finally, the model adds the capability to parse sentences as it interprets them, breaking the dependence on an external parser.

8.2 Future work

8.2.1 Symbolic reasoning and sentence encoding

Chapters 3 and 4 reveal that existing models are effective at learning symbolic reasoning in many domains. There is straightforwardly more work to do to better understand the limits of these abilities: Are there any behaviors from natural logic that cannot be learned from reasonable amounts of training data? Is it possible to accurately predict how much data a model will need in order to be able to learn a given reasoning pattern? Further experiments of the kind presented in these chapters should yield answers to questions of this kind.

In addition, these chapters demonstrate that artificial neural network models can be effective at simulating inference in a simple but nontrivial formal logic. It would be valuable to better understand the limits of this ability for logical systems more generally, and how closely these limits track known lower bounds on complexity for inference.

Finally, difficult questions remain about the ways in which neural network models like these represent symbolic data: Do notions like entailment or contradiction have any geometric interpretations within the semantic spaces induced by these models? How do complex function words like quantifiers act on their arguments within the highly limited composition functions that are available to these models? Neither the theory of natural logic nor any existing tools for neural network analysis have yet offered any clear insights on these questions, or even any clear directions for research. Producing substantial human-interpretable results about the structures of these representations will require creative new methods for analysis, but to do so could yield substantial improvements to the design and training of these models.

8.2.2 Data for natural language inference

Chapter 5 introduces the first large-scale human-annotated corpus for NLI, but this corpus only contains sentences that come from a single constrained genre of text: image captions. My new data collection strategy required the use of captions, but this genre limits the applications of the corpus. There are many phenomena like tense, aspect, reported beliefs, and conditionals that appear frequently in open domain text, but rarely if ever in captions. This makes it impractical to use SNLI as the primary training corpus for models that are meant to be able to operate on open-domain text, and while this was not a major goal in the creation of SNLI, there would be clear value in a multiple-genre training corpus. More crucially, this choice of a single genre limits SNLI's value for its intended use as a means of evaluating sentence-encoding models. It is possible that a model could do well on SNLI without being able to learn to reason effectively with tense or aspect, and the only way to circumvent this weakness of SNLI is to develop new techniques to enable the collection of high quality inference corpora for new genres of text.

8.2.3 Using tree structure in sentence encoding

Chapter 7's introduction of SPINN leaves open two promising directions for future work. SPINN's tree–sequence hybrid model is demonstrably effective as a way of incrementally constructing sentence encodings, and soft attention—a technique I have not yet explored—is demonstrably effective as a way of reasoning over sentence pairs using intermediate representations from sentence-encoding models. Combining the two should be straightforward and effective. In a more ambitious direction, it should be possible to adapt the design of the model to make it possible to learn to parse using only the supervision signal from a semantic task like SNLI without any explicit parsing supervision. In the best case, this would make it possible to learn novel semantically optimal parsing formalisms, yielding both better model performance and new sources of insight into the role of syntax in semantic interpretation.

8.2.4 The future of learned models for sentence meaning

To take a broader perspective, the techniques explored in this dissertation require accurate training data for a semantic task in order to learn to do that task. It is possible to find or collect usable data for many NLP tasks, at least in well-studied high-resource languages like English, but this need for data presents a major obstacle to building effective systems for more minor tasks or for any of the thousands of currently-spoken lower-resource languages. Building high-quality general-purpose neural networks for sentence understanding in these settings will ultimately require substantial progress in the areas of unsupervised learning—or learning from data without task-specific labels—and transfer learning—adapting learned knowledge from one task or language to another.

The positive results in this dissertation on both artificial and natural language data suggest that neural network models are broadly capable of learning effective representations for natural language meaning and that models that make use of syntactic structure can be both fast and effective. With further research into the directions named here and others, the future is bright for these models both as solutions to major open problems in NLP and as new models for better understanding the nature of human language and human language learning.

Bibliography

Aho, Alfred V. & Jeffrey D. Ullman. 1972. *The theory of parsing, translation, and compiling*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Angeli, Gabor & Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 534–545. Doha, Qatar: Association for Computational Linguistics.

Bahdanau, Dzmitry, Kyunghyun Cho & Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the international conference on learning representations (ICLR)*.

Bankova, Desislava, Bob Coecke, Martha Lewis & Daniel Marsden. 2016. Graded entailment for compositional distributional semantics. In *Proceedings of the 13th international conference on quantum physics and logic*.

Baroni, Marco, Raffaela Bernardi & Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology (LiLT)* 9. 241–346.

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly & Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in neural information processing systems (NIPS)*, 1171–1179.

Bengio, Yoshua, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin & Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in machine learning*, 137–186. New York, NY: Springer.

van Benthem, Johan. 2008. A brief history of natural logic. In M. Chakraborty, B. Löwe, M. Nath Mitra & S. Sarukki (eds.), *Logic, Navya-Nyaya and applications:*

Homage to Bimal Matilal, London: College Publications.

Blackburn, Patrick & Johan Bos. 2005. *Representation and inference for natural language*. Stanford, CA: CSLI Publications.

Bordes, Antoine, Nicolas Usunier, Sumit Chopra & Jason Weston. 2016. Large-scale simple question answering with memory networks. `http://arxiv.org/abs/1506.02075`.

Bos, Johan & Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing (HLT-EMNLP)*, 628–635. Vancouver, Canada: Association for Computational Linguistics.

Bowman, Samuel R., Gabor Angeli, Christopher Potts & Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, 632–642. Lisbon, Portugal: Association for Computational Linguistics.

Bowman, Samuel R., Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning & Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany: Association for Computational Linguistics.

Bowman, Samuel R., Christopher D. Manning & Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015 NIPS workshop on cognitive computation: Integrating neural and symbolic approaches*, Montreal, Canada.

Bowman, Samuel R., Christopher Potts & Christopher D. Manning. 2015c. Learning distributed word representations for natural logic reasoning. In *Knowledge representation and reasoning: Integrating symbolic and neural approaches: Papers from the 2015 AAAI spring symposium*, 10–13. Stanford, CA: AAAI.

Bowman, Samuel R., Christopher Potts & Christopher D. Manning. 2015d. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 12–21. Beijing, China:

Association for Computational Linguistics.

Buys, Jan & Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 2: Short papers*, 863–869. Beijing, China: Association for Computational Linguistics.

Chen, Danqi & Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 740–750. Doha, Qatar: Association for Computational Linguistics.

Chen, Danqi, Richard Socher, Christopher D. Manning & Andrew Y. Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. In *Proceedings of the international conference on learning representations (ICLR)*.

Cheng, Jianpeng, Li Dong & Mirella Lapata. 2016. Long short-term memory-networks for machine reading. `http://arxiv.org/abs/1601.06733`.

Chklovski, Timothy & Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In Dekang Lin & Dekai Wu (eds.), *Proceedings of EMNLP 2004*, 33–40. Barcelona, Spain: Association for Computational Linguistics.

Cho, Kyunghyun, Bart van Merrïenboer, Dzmitry Bahdanau & Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation*, 103–111. Doha, Qatar: Association for Computational Linguistics.

Cho, Kyunghyun, Bart van Merrïenboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk & Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Clark, Stephen, Bob Coecke & Mehrnoosh Sadrzadeh. 2013. The Frobenius anatomy of relative pronouns. In András Kornai & Marco Kuhlmann (eds.), *Proceedings of the 13th meeting on the mathematics of language (MoL)*, Sofia, Bulgaria: Association for Computational Linguistics.

Clarke, Daoud. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics* 38(1). 41–71.

Coecke, Bob, Mehrnoosh Sadrzadeh & Stephen Clark. 2011. Mathematical foundations for a compositional distributed model of meaning. *Linguistic Analysis* 36(1–4). 345–384.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu & Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)* 12(Aug). 2493–2537.

Condoravdi, Cleo, Dick Crouch, Valeria de Paiva, Reinhard Stolle & Daniel G. Bobrow. 2003. Entailment, intensionality and text understanding. In Graeme Hirst & Sergei Nirenburg (eds.), *Proceedings of the HLT-NAACL 2003 workshop on text meaning*, 38–45.

Conneau, Alexis, Holger Schwenk, Loïc Barrault & Yann Lecun. 2016. Very deep convolutional networks for natural language processing. `http://arxiv.org/abs/1606.01781`.

Cooper, Robin, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio et al. 1996. Using the framework. Tech. rep. LRE 62-051 D-16, The FraCaS Consortium.

Dagan, Ido, Oren Glickman & Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, 177–190. New York, NY: Springer.

Duchi, John, Elad Hazan & Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)* 12(Jul). 2121–2159.

Dummett, Michael A. E. 1991. *The logical basis of metaphysics.* Cambridge, MA: Harvard University Press.

Dyer, Chris, Miguel Ballesteros, Wang Ling, Austin Matthews & Noah A. Smith.

2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 1: Long papers*, 334–343. Beijing, China: Association for Computational Linguistics.

Dyer, Chris, Adhiguna Kuncoro, Miguel Ballesteros & Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (NAACL-HLT)*, 199–209. San Diego, California: Association for Computational Linguistics.

Elman, Jeffrey L. 1989. Representation and structure in connectionist models. Tech. Rep. 8903 Center for Research in Language.

Elman, Jeffrey L. 1990. Finding structure in time. *Cognitive science* 14(2). 179–211.

Elman, Jeffrey L. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning* 7(2–3). 195–225.

Emami, Ahmad & Frederick Jelinek. 2005. A neural syntactic language model. *Machine learning* 60(1–3). 195–227.

Fellbaum, Christiane. 2010. WordNet. In *Theory and applications of ontology: Computer applications*, 231–243. New York, NY: Springer.

Francez, Nissim, Roy Dyckhoff & Gilad Ben-Avi. 2010. Proof-theoretic semantics for subsentential phrases. *Studia Logica* 94(3). 381–401.

Francis, W. Nelson & Henry Kucera. 1979. Brown corpus manual. Brown University.

Fried, Daniel, Tamara Polajnar & Stephen Clark. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 2: Short papers*, 731–736. Beijing, China: Association for Computational Linguistics.

Fyodorov, Yaroslav, Yoad Winter & Nissim Francez. 2000. A natural logic inference system. In *Proceedings of the 2nd workshop on inference in computational semantics (ICoS)*.

Garcez, A. S. d'Avila, Krysia Broda & Dov M. Gabbay. 2001. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence* 125(1). 155–207.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan & Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 1–9. Prague: Association for Computational Linguistics.

Giles, C. Lee, Clifford B. Miller, Dong Chen, Guo-Zheng Sun, Hsing-Hen Chen & Yee-Chun Lee. 1992. Extracting and learning an unknown grammar with recurrent neural networks. In *Advances in neural information processing systems (NIPS)*, 317–324.

Glorot, Xavier & Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the international conference on artificial intelligence and statistics (AISTATS)*, vol. 9, 249–256.

Goller, Christoph & Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the IEEE international conference on neural networks*, vol. 1, 347–352. IEEE.

Goodfellow, Ian, Yoshua Bengio & Aaron Courville. 2016. Deep learning. Book in preparation for MIT Press. `http://www.deeplearningbook.org`.

Grefenstette, Edward. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of the second joint conference on lexical and computational semantics (*SEM), volume 1: Proceedings of the main conference and the shared task: Semantic textual similarity*, 1–10. Atlanta, Georgia: Association for Computational Linguistics.

Grefenstette, Edward, Karl Moritz Hermann, Mustafa Suleyman & Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Advances in neural information processing systems (NIPS)*, 1828–1836.

Grefenstette, Edward & Mehrnoosh Sadrzadeh. 2013. Multi-step regression learning for compositional distributional semantics.

Grefenstette, Edward, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke & Stephen Pulman. 2014. Concrete sentence spaces for compositional distributional models of meaning. In *Computing meaning*, 71–86. New York, NY: Springer.

Hallnäs, Lars & Peter Schroeder-Heister. 1990. A proof-theoretic approach to logic programming. I. Clauses as rules. *Journal of Logic and Computation* 1(2). 261–283.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren & Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 1026–1034.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford, UK: Blackwell.

Henderson, James. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd meeting of the Association for Computational Linguistics (ACL), main volume*, 95–102. Barcelona, Spain.

Hochreiter, Sepp & Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8). 1735–1780.

Hornik, Kurt, Maxwell Stinchcombe & Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5). 359–366.

Huet, Gérard. 1997. The zipper. *Journal of functional programming* 7(5). 549–554.

Icard III, Thomas F. 2012. Inclusion and exclusion in natural language. *Studia Logica* 100(4). 705–725.

Icard III, Thomas F. & Lawrence S. Moss. 2013a. A complete calculus of monotone and antitone higher-order functions. *Proceedings of Topology, Algebra, and Categories in Logic (TACL)* 96–100.

Icard III, Thomas F. & Lawrence S. Moss. 2013b. Recent progress on monotonicity. *Linguistic Issues in Language Technology (LiLT)* 9(7). 167–194.

Ioffe, Sergey & Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on machine learning (ICML)*, 448–456.

Irsoy, Ozan & Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems (NIPS)*, 2096–2104.

Iyyer, Mohit, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher & Hal

Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 633–644. Doha, Qatar: Association for Computational Linguistics.

Ji, Yangfeng & Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)* 3. 329–344.

Joulin, Armand & Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems (NIPS)*, 190–198.

Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer & Yonghui Wu. 2016. Exploring the limits of language modeling. `http://arxiv.org/abs/1602.02410`.

Kádár, Ákos, Grzegorz Chrupała & Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. `http://arxiv.org/abs/1602.08952`.

Kalchbrenner, Nal, Edward Grefenstette & Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (ACL), volume 1: Long papers*, 655–665. Baltimore, Maryland: Association for Computational Linguistics.

Karpathy, Andrej, Justin Johnson & Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. `http://arxiv.org/abs/1506.02078`.

Katz, Jerrold J. 1972. *Semantic theory*. New York: Harper & Row.

Kiperwasser, Eliyahu & Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree LSTMs. `http://arxiv.org/abs/1603.00375`.

Kiros, Ryan, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba & Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems (NIPS)*, 3276–3284.

Klein, Dan & Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the Association for Computational Linguistics (ACL)*, 423–430. Sapporo, Japan: Association for Computational Linguistics.

Kotlerman, Lili, Ido Dagan, Idan Szpektor & Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(4). 359–389.

Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael Bernstein & Li Fei-Fei. 2016. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. `http://arxiv.org/abs/1602.07332`.

Kruszewski, German, Denis Paperno & Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics (TACL)* 3. 375–388.

Lai, Alice & Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval)*, 329–334. Dublin, Ireland: Association for Computational Linguistics and Dublin City University.

Lambek, Joachim. 1958. The mathematics of sentence structure. *The American Mathematical Monthly* 65(3). 154–170.

LeCun, Yann, Léon Bottou, Yoshua Bengio & Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11). 2278–2324.

Levesque, Hector J. 2014. On our best behaviour. *Artificial Intelligence* 212. 27–35.

Levy, Omer, Ido Dagan & Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. In *Proceedings of the eighteenth conference on computational natural language learning (CoNLL)*. Ann Arbor, Michigan: Association for Computational Linguistics.

Lewis, David. 1970. General semantics. *Synthese* 22(1/2). 18–67.

Li, Jiwei, Xinlei Chen, Eduard Hovy & Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (NAACL-HLT)*, 681–691. San Diego, California: Association for Computational Linguistics.

Li, Jiwei, Thang Luong, Dan Jurafsky & Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, 2304–2314. Lisbon, Portugal: Association for Computational Linguistics.

Liang, Percy, Michael I. Jordan & Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39(2). 389–446.

Liu, Yang, Chengjie Sun, Lei Lin & Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. `http://arxiv.org/abs/1605.09090`.

Luong, Minh-Thang, Hieu Pham & Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Luong, Thang, Ilya Sutskever, Quoc Le, Oriol Vinyals & Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 1: Long papers*, 11–19. Beijing, China: Association for Computational Linguistics.

Maas, Andrew L., Awni Y. Hannun & Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the ICML workshop on deep learning for audio, speech and language processing.*

Van der Maaten, Laurens & Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)* 9(Nov). 2579–2605.

MacCartney, Bill. 2009. *Natural language inference.* Stanford, CA: Stanford University dissertation.

MacCartney, Bill & Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the eight international conference on computational semantics (IWCS)*, 140–156. Tilburg, The Netherlands: Association for Computational Linguistics.

Magnini, Bernardo, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Guenter Neumann, Tae-Gil Noh, Sebastian Padó, Asher Stern & Omer Levy. 2014. The Excitement open platform for textual inferences. In *Proceedings of 52nd annual meeting of the Association for Computational Linguistics (ACL): System demonstrations*, 43–48. Baltimore, Maryland: Association for Computational Linguistics.

Marelli, Marco, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini & Roberto Zamparelli. 2014a. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval)*, 1–8. Dublin, Ireland: Association for Computational Linguistics and Dublin City University.

Marelli, Marco, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi & Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the ninth international conference on language resources and evaluation (LREC)*, 216–223. Reykjavik, Iceland: European Language Resources Association (ELRA).

de Marneffe, Marie-Catherine, Christopher D. Manning & Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics* 38(2). 301–333.

de Marneffe, Marie-Catherine, Anna N. Rafferty & Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, 1039–1047. Columbus, Ohio: Association for Computational Linguistics.

Mikolov, Tomas, Kai Chen, Greg S. Corrado & Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. `http://arxiv.org/abs/1301.3781`.

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černockỳ & Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the eleventh annual conference of the International Speech Communication Association (INTERSPEECH)*, 1045–1048.

Mikolov, Tomas, Wen-tau Yih & Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics:*

Human language technologies (NAACL-HLT), 746–751. Atlanta, Georgia: Association for Computational Linguistics.

Miller, George A. 1995. WordNet: a lexical database for english. *Communications of the ACM* 38(11).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540). 529–533.

Montague, Richard. 1973. The proper treatment of quantification in ordinary english. In *Approaches to natural language*, 221–242. New York, NY: Springer.

Mou, Lili, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang & Zhi Jin. 2016a. How transferable are neural networks in NLP applications? `http://arxiv.org/abs/1603.06111`.

Mou, Lili, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan & Zhi Jin. 2016b. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of 54th annual meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

Narasimhan, Karthik, Tejas Kulkarni & Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, 1–11. Lisbon, Portugal: Association for Computational Linguistics.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th international workshop on parsing technologies (IWPT)*.

Padó, Sebastian, Tae-Gil Noh, Asher Stern, Rui Wang & Roberto Zanoli. 2015. Design and realization of a modular architecture for textual entailment. *Natural Language Engineering* 21(2). 167–200.

Parikh, Ankur P., Oscar Täckström, Dipanjan Das & Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. `http://arxiv.org/abs/1606.01933`.

Partee, Barbara H. 1984. Compositionality. In *Varieties of formal semantics: Proceedings of the fourth Amsterdam colloquium*, 281–311. Dordrecht: Foris Publications.

Paulus, Romain, Richard Socher & Christopher D. Manning. 2014. Global belief recursive neural networks. In *Advances in neural information processing systems (NIPS)*, 2888–2896.

Pavlick, Ellie, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme & Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 2: Short papers*, 425–430. Beijing, China: Association for Computational Linguistics.

Pennington, Jeffrey, Richard Socher & Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543. Doha, Qatar: Association for Computational Linguistics.

Pham, Nghia The, Germán Kruszewski, Angeliki Lazaridou & Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 1: Long papers*, 971–981. Beijing, China: Association for Computational Linguistics.

Pratt, Lorien Y., Jack Mostow, Candace A. Kamm & Ace A. Kamm. 1991. Direct transfer of learned information among neural networks. In *Proceedings of the ninth national conference on artificial intelligence (AAAI), volume 2*, 584–589. AAAI.

Prawitz, Dag. 1965. *Natural deduction: A proof-theoretical study.* Mineola, NY: Dover Publications.

Rocktäschel, Tim, Edward Grefenstette, Moritz Hermann, Karl, Tomáš Kočiský & Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the international conference on learning representations (ICLR).*

Rocktäschel, Tim & Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In *Proceedings of the 5th workshop on automated knowledge base construction (AKBC)*, 45–50. San Diego, CA: Association for Computational Linguistics.

Rumelhart, David E., Geoffrey E. Hinton & Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323. 533–536.

Sánchez-Valencia, Víctor. 1991. Categorial grammar and natural reasoning. ILTI Publication Series for Logic, Semantics, and Philosophy of Language LP-91-08 University of Amsterdam Amsterdam, The Netherlands.

Shieber, Stuart M. 1983. Sentence disambiguation by a shift-reduce parsing technique. In *Proceedings of the 21st annual meeting of the Association for Computational Linguistics (ACL)*, 113–118. Cambridge, Massachusetts: Association for Computational Linguistics.

Shwartz, Vered, Yoav Goldberg & Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of 54th annual meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

Socher, Richard, Brody Huval, Christopher D. Manning & Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 1201–1211. Jeju Island, Korea: Association for Computational Linguistics.

Socher, Richard, Andrej Karpathy, Quoc V. Le, Christopher D. Manning & Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics (TACL)* 2. 207–218.

Socher, Richard, Cliff C. Lin, Chris Manning & Andrew Y. Ng. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML)*, 129–136.

Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng & Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 conference on empirical methods* *in natural language processing (EMNLP)*, 151–161. Edinburgh, UK: Association for Computational Linguistics.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng & Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*, 1631–1642. Seattle, Washington: Association for Computational Linguistics.

Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever & Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)* 15(1). 1929–1958.

Sutskever, Ilya, Oriol Vinyals & Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*, 3104–3112.

Szabolcsi, Anna. 2009. *Quantification*. Cambridge, UK: Cambridge University Press.

Tai, Kai Sheng, Richard Socher & Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (ACL-IJCNLP), volume 1: Long papers*, 1556–1566. Beijing, China: Association for Computational Linguistics.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. `http://arxiv.org/abs/1605.02688`.

Tieleman, Tijmen & Geoffrey Hinton. 2012. Lecture 6.5: RMSProp: Divide the gradient by a running average of its recent magnitude. Coursera course: Neural Networks for Machine Learning.

Titov, Ivan & James Henderson. 2010. A latent variable model for generative dependency parsing. In *Trends in parsing technology*, New York, NY: Springer.

Van Eijck, Jan & Christina Unger. 2010. *Computational semantics with functional programming*. Cambridge, UK: Cambridge University Press.

Vendrov, Ivan, Ryan Kiros, Sanja Fidler & Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the international conference on learning* *representations (ICLR)*.

Vinyals, Oriol, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever & Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in neural information processing systems (NIPS)*, 2773–2781.

Wang, Rui & Günter Neumann. 2007. Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 36–41. Prague: Association for Computational Linguistics.

Wang, Shuohang & Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (NAACL-HLT)*, 1442–1451. San Diego, California: Association for Computational Linguistics.

Wang, Sida & Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the Association for Computational Linguistics (ACL), volume 2: Short papers*, 90–94. Jeju Island, Korea: Association for Computational Linguistics.

Ward, Gregory & Betty Birner. 2004. Information structure and non-canonical syntax. In Laurence R. Horn & Gregory Ward (eds.), *Handbook of pragmatics*, 153–174. Oxford, UK: Blackwell.

Watanabe, Yotaro, Junta Mizuno, Eric Nichols, Naoaki Okazaki & Kentaro Inui. 2012. A latent discriminative model for compositional entailment relation recognition using natural logic. In *Proceedings of COLING*, 2805–2820. Mumbai, India: The COLING 2012 Organizing Committee.

Wen, Tsung-Hsien, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke & Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, 1711–1721. Lisbon, Portugal: Association for Computational Linguistics.

Weston, Jason, Antoine Bordes, Sumit Chopra & Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings* *of the international conference on learning representations (ICLR)*, .

Weston, Jason, Sumit Chopra & Antoine Bordes. 2015. Memory networks. In *Proceedings of the international conference on learning representations (ICLR)*.

Widdows, Dominic. 2004. *Geometry and meaning*. Stanford, CA: CSLI Publications.

Widdows, Dominic & Trevor Cohen. 2014. Reasoning with vectors: a continuous model for fast robust inference. *Logic Journal of IGPL* 23(2). 141–173.

Winograd, Terry. 1972. Understanding natural language. *Cognitive Psychology* 3(1). 1–191.

Woods, William A., Ronald M. Kaplan, Bonnie Nash-Webber & Center Manned Spacecraft Center. 1972. The lunar sciences natural language information system: Final report. Tech. rep. BBN Technologies Cambridge, MA.

Yin, Wenpeng, Hinrich Schütze, Bing Xiang & Bowen Zhou. 2015. ABCNN: attention-based convolutional neural network for modeling sentence pairs. `http://arxiv.org/abs/1512.05193`.

Young, Peter, Alice Lai, Micah Hodosh & Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2. 67–78.

Zeiler, Matthew D. 2012. ADADELTA: An adaptive learning rate method. `http://arxiv.org/abs/1212.5701`.

Zettlemoyer, Luke S. & Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st conference on uncertainty in artificial intelligence (UAI)*.

Zhang, Xiang, Junbo Zhao & Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems (NIPS)*, 649–657.

Zhang, Xingxing, Liang Lu & Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (NAACL-HLT)*, 310–320. San Diego, California: Association for Computational Linguistics.