Recognizing Logical Entailment: Reasoning with Recursive and Recurrent Neural Networks

MSc Thesis *(Afstudeerscriptie)*

written by

Mathijs S. Mul
(born October 14th, 1993 in Nottingham, United Kingdom)

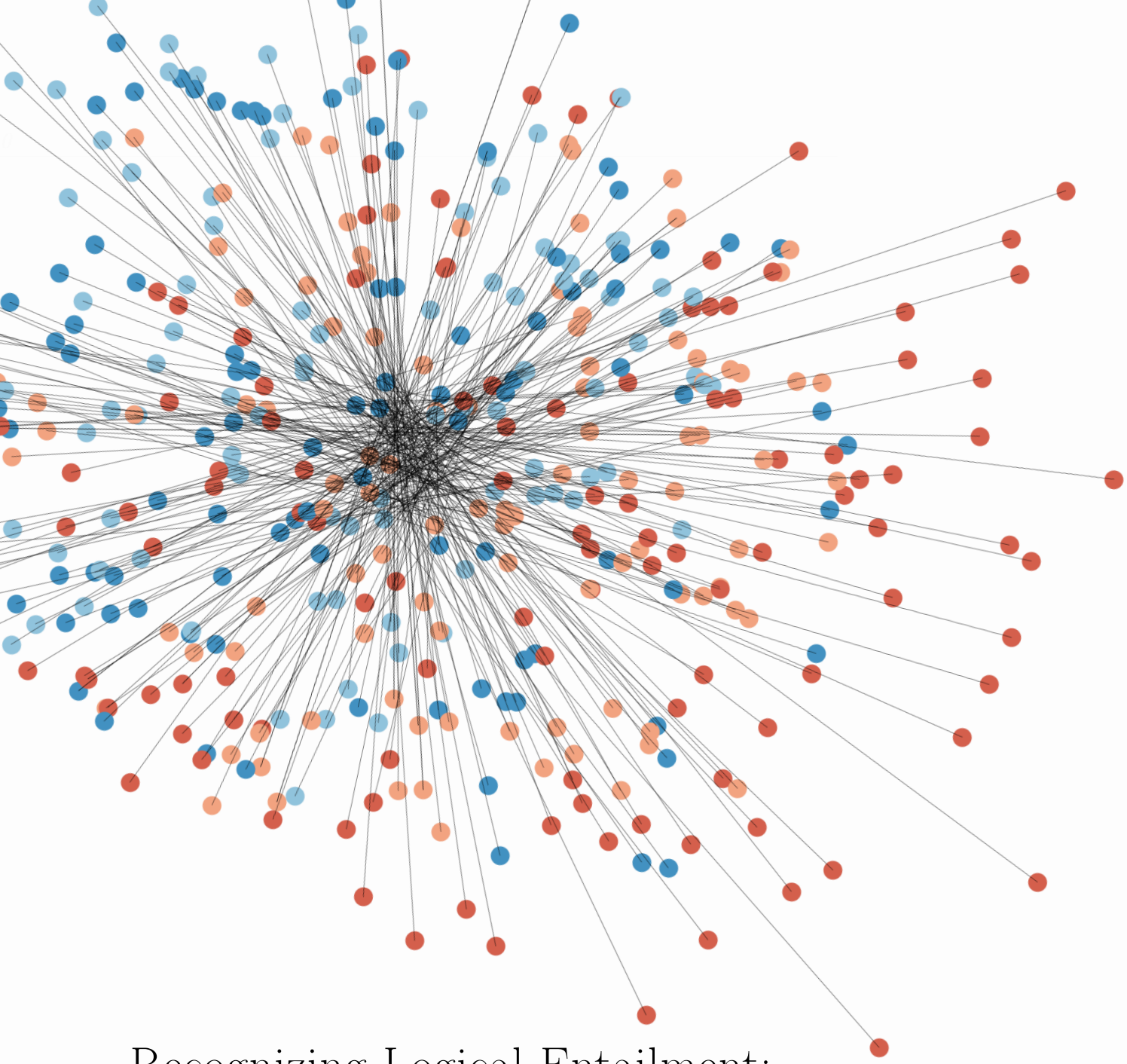
under the supervision of **Dr. Willem Zuidema**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
July 9th, 2018	Dr. Iacer Calixto
	Dr. Raquel Fernández
	Dr. Jakub Szymanik
	Prof. Dr. Yde Venema (*chair*)
	Dr. Willem Zuidema

Institute for Logic, Language and Computation



Recognizing Logical Entailment: Reasoning with Recursive and Recurrent Neural Networks

Mathijs S. Mul

Abstract

This thesis studies the ability of several compositional distributional models to recognize logical entailment relations. We first investigate the performance of recursive neural matrix and tensor networks on an artificially generated data set labelled according to a natural logic calculus. Several issues with this set-up are identified, which we aim to solve by introducing a new task: First-Order Entailment Recognition. In combination with an automated theorem prover and model builder, an artificial language with higher grammatical complexity is used to generate a new data set, whose labels are determined by the semantics of first-order logic. The tree-shaped networks perform well on the new task, as opposed to a bag-of-words baseline. Qualitative analysis is performed to reveal meaningful clusters on the level of word embeddings and sentence vectors. A novel recurrent architecture is proposed and evaluated on the same task. The high testing scores obtained by GRU cells in particular prove that recurrent models can learn to apply first-order semantics without any cues about syntactic structure or lexicalization. Projection of sentence vectors demonstrates that negation creates a mirroring effect at sentence level, while strong clustering with respect to verbs and object quantifiers is observed. Diagnostic classifiers are used for quantitative interpretation, which suggests that the best-performing GRU encodes information about several linguistic hypotheses. Additional experiments are conducted to assess whether the recurrent models owe their success to compositional learning. They do not profit from the availability of syntactic cues, but prove capable of generalization to unseen lengths. After training with fixed GloVe vectors, the GRU can handle sentence pairs with unseen words whose embeddings are provided. These results suggest that recurrent models possess at least basic compositional skills.

Acknowledgements

I am most grateful to my supervisor Jelle Zuidema for all provided guidance, support and advice while I was working on this thesis. If I overwhelmed you with the amount of material to proofread, I can only blame this on the many interesting discussions we had. I also want to thank everyone associated with the Cognition, Language & Computation Lab at the ILLC for their valuable comments - in particular Michael for generously sharing his hard-earned insights with me, Samira for opening the black box of the server and Dieuwke for spotting that one missing logarithm. Sara Veldhoen, Jakub Szymanik and Luciano Serafini I want to thank for their help and comments during the initial stage of the project. And of course there are my family and friends. I doubt whether my thesis committee members would appreciate it if they had to read even more pages, so I will not mention you by name. Thank you all for helping me to focus on this thesis - or for doing the exact opposite.

Contents

Chapter 1

Introduction

"We don't see any contradiction"
— Sarah Huckabee Sanders,
Press Secretary of Donald Trump

The ability to recognize entailment, contradiction or any other kind of logical relation is an essential aspect of human reasoning and language use. It is what enables us to follow arguments, to reject conclusions or detect inconsistencies. How can this faculty best be modelled? Both from an NLP and a cognitive science perspective this is a crucial question, which will be the focus of the current research.

The task of identifying semantic relations in natural language is known as 'natural language inference', or 'recognition of textual entailment' in the case of written objects. There is a tradition of symbolic approaches to the challenge, which have attempted to determine relations between natural language expressions by representing the problem in logical terms (Rinaldi et al. 2003; Bos and Markert 2005; Tatu and Moldovan 2005). Statements are analyzed in terms of their compositional structure and formalized as per the syntactic conventions of a logic. The logical encodings are processed by a rule-based deductive method to establish their relation. Usually, the proof system is not only presented with the pair of statements to be assessed, but also with a large amount of background information. This knowledge base captures the ontology of a particular context, which comprises valid formulas that serve as axioms in a derivation.

In their purest form, symbolic methods face some major shortcomings. Due to the ambiguity of natural language, formalization is not always straightforward, let alone that translation into logical representations can easily be automated on a large scale. The approach relies on the principle of compositionality, which states that meaning primarily depends on syntax and lexical semantics, to which factors such as context are subordinated. Ontologies have to be composed separately, and quickly risk becoming excessive in size or complexity. Even if a manageable knowledge base is available, the

content matter tends to concern demarcated themes, which risks rendering the system as a whole overly specific in its applicability.

An alternative course of action, which has gained popularity in recent years, is based on data rather than logic. Data-driven models do not infer entailment relations from hardcoded logical heuristics, but apply machine learning techniques to large numbers of training instances. They adapt their internal representations to match the statistical properties of the data witnessed during the training phase, a process that need not be governed by mechanisms with a clear linguistic or logical interpretation. It is common for such probabilistic models to replace the compositional hypothesis with its distributional counterpart, which assigns most importance to context. Based on this assumption, vectors capturing distributional properties can be used to represent words or higher-level entities.

Data-driven models have proven successful in entailment recognition tasks. Here follows a short overview of relevant research, mainly aimed readers with prior knowledge on the subject. Baroni, Bernardi, et al. 2012 showed that entailment between short nominal and quantifier phrases can be established using purely distributional encodings of the phrases. Socher, Huval, et al. 2012 combined distributional and compositional semantics in tree-shaped neural networks, which recursively construct complex representations from word embeddings. These networks can determine semantic relations such as cause-effect or member-collection. Rocktäschel, Bošnjak, et al. 2014 computed low-dimensional embeddings of objects and relations, which can be used to perform first-order inferences between rudimentary formal expressions, thereby providing a proof-of-concept for the distributional modelling of first-order semantics.

In 2014, the Sentences Involving Compositional Knowlege (SICK) data set was released. It contains approximately 10,000 English sentence pairs, which human annotators labelled with the entailment relation 'neutral', 'contradiction' or 'implication' on Amazon Mechanical Turk (Marelli et al. 2014). It has since been used as a benchmark for compositional distributional model testing. Bowman, Potts, and Manning 2015 created an artificial data set with pairs of short premises and hypotheses, automatically labelled by a natural logic calculus, and showed that tree-shaped neural networks can accurately predict the entailment relations between unseen sentence pairs. Performance improved if the composition function contained a tensor product term in addition to traditional matrix multiplication. Moreover, when combined with a parser the recursive networks obtained competitive results on the SICK test.

In 2015, Bowman and others released the Stanford Natural Language Inference (SNLI) corpus, which is similar to SICK in its format, labels and generation, but much larger: it contains more than 500,000 English sentence pairs. Bowman, Angeli, et al. 2015 showed that an LSTM architecture achieves a reasonable testing accuracy on the SNLI data. Rocktäschel, Grefenstette, et al. 2015 improved their results by extending the LSTM set-up with an attention mechanism.

Other types of compositional distributional modelling were proposed by Bankova et al. 2016, Serafini and Garcez 2016 and Sadrzadeh, Kartsaklis, and Balkır 2018. These approaches have in common that models are endowed with algebraic characterizations of

particular linguistic features, either in the form of pregroup grammars or neural-symbolic structures. Allamanis et al. 2016 described a neural method to classify algebraic and logical expressions according to their truth conditions, thereby modelling the particular entailment relation of equivalence. Evans, Saxton, et al. 2018 recently introduced 'PossibleWorldNets' to predict the semantic relation between pairs of formulas in propositional logic, computed as a convolution over possible worlds.

The above-mentioned entailment recognition models intend to capture semantic behavior. Therefore, their targets depend on the semantics that are assumed, either by explicit knowledge representation or by extraction from data. In this regard, all described methods can be divided into one of the following two categories:

1. those that have a clear theory of meaning
2. those that do not

Category 2 prevails and includes all compositional distributional models that are designed to address the challenges of corpora such as SICK and SNLI. Apart from reliability issues of Amazon Mechanical Turk, the notion of semantic entailment that these data sets intend to cover is at best ill-defined. E.g., SNLI assigns 'entailment' to a pair with premise 'A soccer game with multiple males playing' and hypothesis 'Some men are playing a sport'. What notion of consequence licenses the entailment from a nominal phrase to a sentence? No semantic framework is available to assess or control such inferences, which ultimately depend on the colloquial understanding of Mechanical Turk workers. If the objective is purely empirical this need not be problematic, but from a theoretical perspective it seems preferable to be aware of the actual notion of entailment that is being modelled. Furthermore, the informal nature of the data generation process causes coarse-grained distinctions between three general classes, and leaves no space for more nuanced entailment relations.

Methods belonging to category 1, which depend or concentrate on explicit semantics, can do so in two ways. They either generate data sets that adhere to the principles of a particular theory (e.g. Evans, Saxton, et al. 2018), or they supply their models with a symbolic representation of such principles (e.g. Sadrzadeh, Kartsaklis, and Balkır 2018). In practice, this has mostly led to research focusing on totally formal data, or on models that are so dependent on representations of specific entities that their generalizing potential is limited. In other words, if methods lacking a clear semantic profile are not logical enough, then methods that do adopt such a profile seem not natural enough.

The most notable exception is the research of Bowman, Potts, and Manning 2015. Bowman e.a. adopted a natural logic as semantic foundation, but did not supply their tree-shaped models with explicit details about this theory. Moreover, using natural logic they generated a data set that is artificial but not formal, containing readily interpretable expressions such as '((all (not warthogs)) move)'. The instances were labelled with seven different entailment relations, as opposed to the three general SICK and SNLI classes. The target behavior on this data set is dictated by the logic that was

used to generate it. In other words, good testing performance indicates that classifiers learnt logical semantics while performing natural language inference. This is very different from other experiments, which dealt with logical semantics or natural language, but not with both.

This thesis continues Bowman's line of research, by approaching entailment as a semantic phenomenon that can be recognized in natural language but that is produced by logic: can compositional distributional models learn logical semantics from natural language? By taking this perspective, the aim is to address a niche in the field of computational semantics that has been largely unoccupied since Bowman's experiments.

Essential notions from semantics, logic and neural modelling are discussed in Chapter 2, which is especially aimed at readers with a different background. Chapter 3 is dedicated to the Quantified Natural Logic Inference task, as introduced by Bowman 2013. The data generation process is described, as well as the recursive models, their performance in the original study and a new replication. The chapter concludes by listing a variety of problems associated with these experiments and the role of natural logic.

Chapter 4 addresses the identified issues by proposing a new classification task: First-Order Entailment Recognition. A new data set is generated according to the semantics of first-order logic. The included sentences are non-formal and have greater length and linguistic complexity than those in Bowman's data. The same tree-shaped models are trained on the new data, and subjected to interpretation.

Chapter 5 examines whether the task can also be handled by recurrent models, which process sentences sequentially instead of recursively. A new architecture is suggested, and its performance is studied with three different recurrent units: Simple Recurrent Networks, Gated Recurrent Units and Long-Short Term Memory. The results are interpreted both qualitatively and quantitatively, using the new technique of diagnostic classification. Particular attention is paid to the question whether the implemented recurrent networks are capable of compositional learning, which would enable them to handle arbitrary structures with known constituents. Several experiments are performed to test different compositional capacities of the models. A concluding section reflects on the obtained results, and the new questions that they raise.

Chapter 2

Theoretical background

This preliminary chapter briefly introduces the most important theoretical themes and notions underlying the project. Three different perspectives on semantics are described, as well as the two types of logic and neural network basics that will be used in later chapters. The information in this chapter is mostly assumed familiar, but included for a broader readership.

2.1 Semantics

This research focuses on the automated recognition of entailment relations between sentences, which is essentially a semantic challenge. It requires not only an adequate representation of the meaning of individual statements, but also of paired expressions whose relation is captured by a specific kind of logical entailment. In modern semantics, two major paradigms can be distinguished: the compositional and the distributional approach. Both are discussed below, as well as the hybrid perspective combining both. Note that, in practice, the opposition between the different strands need not be as harsh as this introduction might suggest. The focus on their discrepancies is intended to single out the theoretical principles lying at the basis of the approaches in their least compromised form.

2.1.1 Compositional

Compositional semantics relies on the principle of compositionality. A clear formulation of the principle states that 'the meaning of a complex expression is determined by its structure and the meanings of its constituents' (Szabó 2004). It is often attributed to Frege, who did not explicitly assert it as a doctrine but made many claims in the same spirit, e.g. that 'the possibility of our understanding sentences which we have never heard before rests evidently on this, that we can construct the sense of a sentence out of parts that correspond to words' (Frege [1914] 1980, p. 79).

If compositional semanticists are right, the meaning of any expression is derivable from the meanings of its elements, together with the way in which these elements are

structured. Or, as Partee phrases it: 'the meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined' (Partee 1984, p. 281). This requires that both syntax and lexical semantics (meanings of primitive vocabulary items) of the language in question are known.

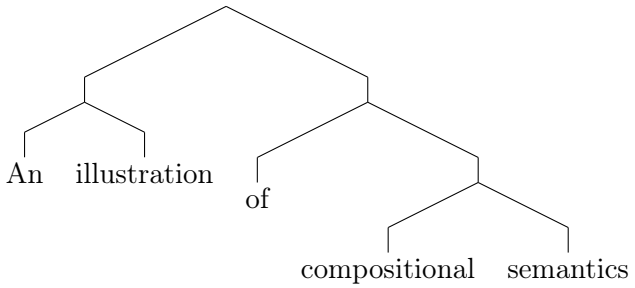


Figure 2.1: An illustration of compositional semantics.

Suppose that the syntax of English is shaped by binary trees. Then Figure 2.1 shows the interpretation of the nominal phrase 'An illustration of compositional semantics'. The total statement is analyzed syntactically, which produces the parse tree with the individual words at its leaf nodes. The words are atomic items, whose meaning must be specified by the lexical semantics and cannot be further decomposed under the current syntactic assumptions. An alternative example is propositional logic, whose semantics is fully compositional. If a valuation specifies the truth values of the atomic constants occurring in a formula, then the status of the entire proposition follows immediately from the truth-functional definition of the involved connectives.

Compositionality is often justified by the 'argument from productivity'. According to this argument, it explains the human capacity to interpret and generate infinitely many different expressions on the basis of a finite number of linguistic experiences and internalized semantic units (Dowty 2007; Groenendijk and Stokhof 2004). Additional support is offered by the 'argument from systematicity' (Fodor 1998). Systematicity is the property allowing languages to combine syntactically analogous expressions in identical ways. E.g., if 'an illustration' and 'a visualization' are of the same syntactic type, and 'an illustration of compositional semantics' is grammatical, then 'a visualization of compositional semantics' is so too. For these and similar reasons, many consider compositionality to be a major, if not the defining, foundational element of human language and thought (Chomsky 1957; Montague 1970).

As intuitive as it seems, the compositional approach to semantics is not uncontested. Consider idiomatic expressions such as 'beating around the bush' or 'barking up the wrong tree'. These are complex expressions whose interpretation is not constructed from the literal meaning of its constituent words, but rather from their combination as phrasal configurations, thereby challenging the principle of compositionality (Goldberg 2015). Another problem lies in sentences containing propositional attitudes such as 'believing' (Pelletier 1994). 'Phacochoeri' is the Latin name for 'warthogs', and is therefore supposed to have the same meaning. However, 'Sam believes warthogs are mammals' is

not necessarily equivalent to 'Sam believes Phacochoeri are mammals', for Sam may not know the term 'Phacochoerus'. This suggests that lexical semantics and syntactic structure alone do not account for all meaning. Even if this were the case, lexical semantics is problematic enough in itself, because it assumes that words have unique meanings, which exist in isolation. This does no justice to the relevance of context to correctly interpreting sentence constituents (Travis 1994; Lahav 1989).

2.1.2 Distributional

Orthogonal to compositional semantics is distributional semantics. This approach does not depart from the principle of compositionality, but relies on the distributional hypothesis, which states that 'words that occur in the same contexts tend to have similar meaning ' (Pantel 2005, p. 126). Firth is often credited with an early formulation of the hypothesis, by claiming that '[y]ou shall know a word by the company it keeps' (Firth 1957, p. 11). Harris had already stated three years earlier that 'distributional statements can cover all of the material of a language without requiring support from other types of information' (Harris 1954, p. 34).

Distributional semantics is also known as statistical semantics, because it uses statistical methods to find numerical representations of meaning. Delavenay defined it as the 'statistical study of meanings of words and their frequency and order of recurrence' (Delavenay 1960, p. 133). To carry out this study, natural language corpora are required for the computation of word counts relative to documents, topics or other words. Such quantitative analysis gives shape to a context-based notion of meaning. The more data are available, the more accurately the resulting distribution is supposed to match semantic intuitions. To put it otherwise, 'words with similar meanings will occur with similar neighbors if enough text material is available' (Schütze and Pedersen 1995, p. 166).

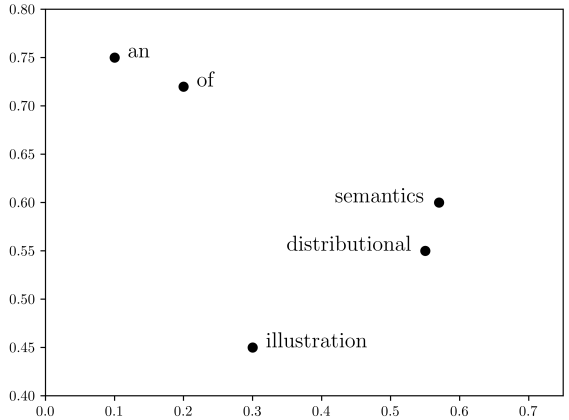


Figure 2.2: An illustration of distributional semantics.

More concretely, distributional semanticists process linguistic data in order to con-

struct representations of semantic units (usually words) as finite-dimensional vectors, also known as 'embeddings'. Figure 2.2 visualizes this for the sample phrase 'An illustration of distributional semantics'. Assuming that 2-dimensional word embeddings were generated on the basis of a large and representative corpus of English texts, the plotted points are located at the learned positions of the five words in the phrase. Items 'an' and 'of' are located in the same region, as are 'semantics' and 'distributional'. This reflects the distributional hypothesis: if the meaning of a word is determined by its context, then words with similar statistical properties should be represented by nearby vectors. 'An' and 'of' are both function words, which cluster together. 'Distributional' and 'semantics' often occur in the same contexts, so they also share a subspace. 'Illustration' has no strong semantic connection to any of the other words in the phrase, so it is located in a different region. We see that the toy vector distribution of Figure 2.2 embodies geometric analogues of the semantic connections between words, which demonstrates how statistics can provide a model of relational meaning.

In modern applications, word embeddings tend to contain many more than two units. Dimensionality reduction methods such as Principal Component Analysis and t-distributed Stochastic Neighbor Embedding can be used to project such vectors to lower-dimensional representations (Pearson 1901; Maaten and Hinton 2008). A variety of distance measures is used to express (dis)similarity between embeddings, such as subtractive norm, cosine and Euclidean distance (Lee 1999). Cluster analysis can be performed quantitatively by means of algorithms such as k-means (MacQueen et al. 1967).

There are many different methods of distributional semantic modelling. Traditional algorithms are mostly unsupervised and count-based, focusing on co-occurrence matrices as a means to statistically relate semantic entities. An example is Latent Semantic Analysis, introduced by Letsche and Berry 1997. For an overview of other such methods, see Bullinaria and Levy 2007. Count-based models are hardly used anymore nowadays. Instead, neural or predictive models are favored, which use the machine learning framework of neural networks to learn word embeddings from a training corpus. Well-known examples are the continuous bag-of-words and skip-gram models (Mikolov, Chen, et al. 2013). Roughly stated, the former aims to predict words given their context, whereas the latter predicts a likely context from words. Both are commonly referred to as 'word2vec'. Many follow-up algorithms have been proposed, such as GloVe, which is trained on global co-occurrence counts rather than separate local context windows (Pennington, Socher, and Manning 2014). These and other systems have been used to generate large databases of pretrained word embeddings, many of which are available online. Alternatively, it is possible to treat the embeddings as parameters in existing architectures, and update them as regular layer weights.

Baroni, Dinu, and Kruszewski 2014 showed that neural models outperform their count-based predecessors on several benchmark NLP tasks. The predictive models are not only capable of producing meaningful geometries for words, but also for phrases (Mikolov, Sutskever, et al. 2013). However, encoding more complex objects such as sentences or documents has proven problematic, because a distributional account alone

appears insufficiently scalable to accommodate higher-level compounds (Mohammad et al. 2013; Sadrzadeh and Kartsaklis 2016). Even if this were theoretically possible, data sparsity would be a bottleneck for longer expressions.

2.1.3 Hybrid

Compositional and distributional semantics complement each other in this respect, that the former describes how to handle composite statements without specifying individual word meanings, while the latter successfully captures lexical semantics but fails to model utterances with high complexity. This observation has motivated semanticists to combine both perspectives in a hybrid approach, which is known as 'compositional distributional semantics'. The following citation gives a clear introduction:

> Compositional distributional models extend [word] vector representations from words to phrases and sentences. They work alongside a principle of compositionality, which states that the meaning of a phrase or sentence is a function of the meanings of the words therein. (Sadrzadeh and Kartsaklis 2016, p. 2)

In other words, compositional distributional semantics aims to express the meaning of complex expressions as a function on the embeddings of their primitive constituents. This function can have a recursive structure, such as the parse tree of Figure 2.1, but accept vectorial word representations as input arguments, such as the embeddings in Figure 2.2. There are three main categories of compositional distributional models:

- *Vector mixture models.* Simple bag-of-words models that perform element-wise operations on the input vectors. The embeddings can be added or multiplied, without taking word order into consideration. E.g. Mitchell and Lapata 2008.
- *Tensor-based models.* Models that treat relational words as functions, which take one or more vectors as input arguments. The functions are implemented as tensors, whose order depends on the grammatical type of the represented word. E.g. Coecke, Sadrzadeh, and Clark 2010.
- *Neural models.* Models that use a neural network to compose input embeddings. There are several ways for the model to take sentence structure into account, e.g. by topologically mimicking the syntactic parse (recursive networks) or by implementing feedback cycles (recurrent networks). E.g. Socher, Huval, et al. 2012.

Compositional distributional models have performed very well on a variety of NLP tasks, including sentence similarity recognition (Marelli et al. 2014), sentiment analysis (Le and Zuidema 2015) and entailment classification (Sadrzadeh, Kartsaklis, and Balkır 2018).

2.2 Logics

The relation that has to be identified in the entailment recognition task depends on the logic underlying the utterances in question. In this project, several data sets will be used that contain large numbers of sentence pairs that are not labelled by hand, but by automated deduction methods. These methods rely on the implementation of two systems: natural logic (NL) and first-order logic (FOL). The below subsections describe how both logics are understood in this project.

2.2.1 Natural logic

In modern times, the dominant conception of logic has focused on formal systems. From this perspective, the only legitimate inferences are those between formulas adhering to a rigid, predefined syntax. NL opposes this paradigm, by seeking to offer a deductive method that computes the relations between natural language expressions without requiring intermediate formalization. In other words, its aim is to provide a logical calculus that operates immediately on natural language expressions.

Van Benthem describes NL as 'a system of modules for ubiquitous forms of reasoning that can operate directly on natural language surface form' (Van Benthem 2007, p. 5), and argues that "natural logic' itself cannot be one single mechanism', due to the many different ways in which natural language is used to draw inferences (Van Benthem 1987, p. 460). Thus, rather than one unified framework, NL must be a collection of separate faculties, each of which accounts for a different aspect of the reasoning encountered in natural language usage.

One kind of reasoning that is commonly considered essential to human cognition and language, and which has therefore been the focus of many theories on NL, revolves around monotonicity (Sommers 1982; van Eijck 2005; van Benthem 2007). Monotonicity, following Barwise and Cooper 1981, is a property of quantifiers that specifies how their arguments can be substituted with weaker or stronger notions without affecting validity of the statements in which they occur. There are two kinds of monotonicity: upward (increasing) and downward (decreasing). Their general definitions are as follows:

Definition 2.1. *Upward monotonicity.* A quantifier Q_M of type $(n_1, \cdots, n_k)$ is upward monotone in the ith argument iff the following holds:

If $Q_M[R_1, \cdots, R_k]$ and $R_i \subseteq R'_i$, then $Q_M[R_1, \cdots, R_{i-1}, R'_i, Ri + 1, \cdots, R_k]$, where $1 \leq i \leq k$.

Definition 2.2. *Downward monotonicity.* A quantifier Q_M of type $(n_1, \cdots, n_k)$ is downward monotone in the ith argument iff the following holds:

If $Q_M[R_1, \cdots, R_k]$ and $R'_i \subseteq R_i$, then $Q_M[R_1, \cdots, R_{i-1}, R'_i, Ri + 1, \cdots, R_k]$, where $1 \leq i \leq k$.

Roughly said, an upward monotone quantifier legitimizes the inference from subsets to supersets, while a downward monotone quantifier sanctions the inference from supersets to subsets. E.g., 'every mammal moves' implies that 'every warthog moves', because

'every' is downward monotone in its first argument and 'warthog' is a hyponym of 'mammal'. Likewise, 'some warthog moves' entails 'some mammal moves', because 'some' is upward monotone in its first argument and 'mammal' is a hypernym of 'warthog'.

One NL that is based on a monotonicity calculus is the system advanced by MacCartney 2009 and MacCartney and Manning 2009. This version was adopted by Bowman, Potts, and Manning 2015, and is most relevant to the current project. MacCartney distinguishes seven different entailment relations, which are defined in Table 2.1. Figure 2.3 offers a diagrammatic illustration of their differences. The relations are defined with respect to pairs of sets, but they are also applied to pairs of sentences, which can be interpreted as the sets of possible worlds where they hold true.

name	symbol	set-theoretic definition	example
cover	$\vee$	$x \cap y \neq \emptyset \wedge x \cup y = \mathcal{D}$	`animal, non-turtle`
negation	$\wedge$	$x \cap y = \emptyset \wedge x \cup y = \mathcal{D}$	`able, unable`
forward entailment	$<$	$x \subset y$	`turtle, reptile`
equivalence	$=$	$x = y$	`couch, sofa`
backward entailment	$>$	$x \supset y$	`reptile, turtle`
alternation	$\mid$	$x \cap y = \emptyset \wedge x \cup y \neq \mathcal{D}$	`turtle, warthog`
independence	$\#$	(else)	`turtle, pet`

Table 2.1: The seven entailment relations of MacCartney and Manning 2009. $\mathcal{D}$ denotes the universe of discourse (i.e. the total domain). (Note the difference between the symbols for negation ($\wedge$), conjunction ($\wedge$), cover ($\vee$) and disjunction ($\vee$). This notation is used in order to maintain coherence between the current project, MacCartney and Manning 2009 and the data.)

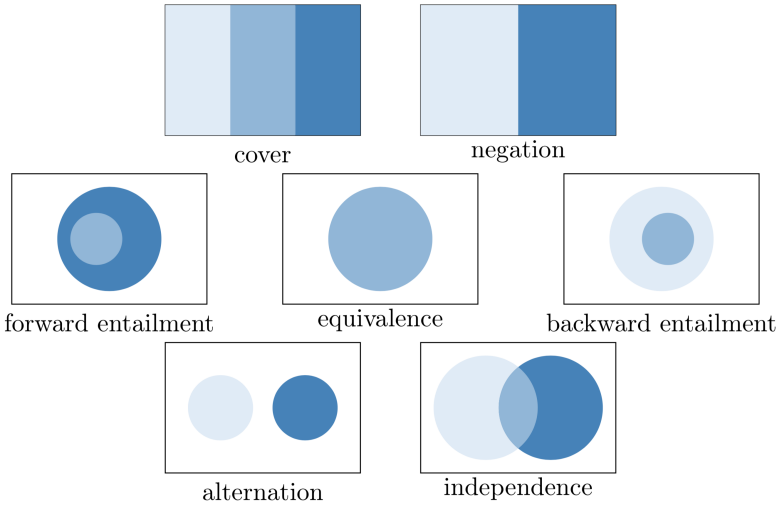


Figure 2.3: Visualization of the seven entailment relations specified in Table 2.1.

MacCartney proposes a procedure to determine the entailment relation between a premise p and a hypothesis h, which is shown here as Algorithm 2.1. The atomic edits of step 1 in Algorithm 2.1 can be deletions, insertions or substitutions of subexpressions in a sentence. E.g., SUB($warthog, mammal$), the substitution of 'warthog' with 'mammal', is the edit that changes 'every warthog moves' into 'every mammal moves'. $(e_j \circ e_i)(s)$ denotes the sucessive applications of edits e_i and e_j to sentence s. Hence, provided that $h = (e_n \circ \cdots \circ e_1)(p)$, the total sequence of n edits transforming premise p into hypothesis h is $\langle e_1, \cdots, e_n \rangle$. By definition, $x_0 = p$. For any $i \in [1, n]$ sentence x_i is the result of applying edit e_i to predecessor x_{i-1} (step 4). Each edit e_i generates a lexical entailment relation $\beta(e_i)$, which is determined at step 5 of Algorithm 2.1. Hence, if e_i = SUB($warthog, mammal$), then $\beta(e_i) =$'$<$', denoting forward entailment because 'warthog' is a hyponym of 'mammal'. Step 6 then assesses the atomic entailment relation $\beta(x_{i-1}, e_i)$ between sentence x_{i-1} and edit e_i, by means of several fixed matrices containing the monotonicity properties of different grammatical categories.[1] Finally, step 7 applies the 'join' operation $\bowtie$ to all previously established atomic entailments, which amounts to a series of look-up operations in the join table for the seven basic entailment relations, given on page 85 of MacCartney 2009 (not repeated here for the sake of brevity).

Input: premise p, hypothesis h
Output: entailment relation $\beta(p, h)$
1 Find atomic edits $\langle e_1, \cdots, e_n \rangle$ such that $h = (e_n \circ \cdots \circ e_1)(p)$
2 $x_0 \leftarrow p$
3 **for** $i \in [1, n]$ **do**
4 $\quad x_i \leftarrow e_i(x_{i-1})$
5 $\quad$ Determine lexical entailment relation $\beta(e_i)$ generated by e_i
6 $\quad$ Project $\beta(e_i)$ upward through semantic composition tree of x_{i-1} to find atomic entailment relation $\beta(x_{i-1}, e_i) = \beta(x_{i-1}, x_i)$
7 $\beta(p, h) \leftarrow \beta(x_0, x_n) = \beta(x_0, e_1) \bowtie \cdots \beta(x_{i-1}, e_i) \bowtie \cdots \bowtie \beta(x_{n-1}, e_n)$

Algorithm 2.1: Inference NL entailment relation, from MacCartney 2009, pp. 105-106.

For a detailed description of MacCartney's NL calculus, with more examples and the exact projectivity matrices that are used, I refer to MacCartney's PhD thesis on this topic (MacCartney 2009) and the concise overview in MacCartney and Manning 2009.

[1] More precisely, MacCartney uses an extended version of the calculus proposed by Sánchez Valencia 1991, generalizing the concept of monotonicity to the notion of 'projectivity'. A number of projectivity matrices specify the functions that return an atomic entailment relation, given some linguistic construction and lexical entailment. MacCartney does not only spell out the projectivity signatures of logical connectives and quantifiers, but also of particular verb classes such as implicatives and factives.

2.2.2 First-order logic

This thesis will also make use of the more conventional system of FOL (also known as first-order predicate calculus). Most of the details of this logic are assumed familiar. In the context of this thesis, the main difference between NL and FOL is that the latter requires natural language statements to be translated into an encoding that meets the syntactic requirements of the logic before inference can take place. That is, expressions must be formalized into sequences of connectives, quantifiers, parentheses, variables, equality symbols, predicates and/or constants that adhere to the recursive definition of a well-formed formula.

The semantics of a FOL system must specify a non-empty domain of discourse $\mathcal{D}$ and an interpretation function $\mathcal{I}$ mapping constants and predicates in the language to elements and relations in the domain $\mathcal{D}$. Together, the domain and the interpretation function form the model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$. Truth of a FOL formula is evaluated with respect to a model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ and a variable assignment g, which sends each variable to some object in $\mathcal{D}$.

There are many deductive systems for FOL, which establish logical consequence by syntactic means, such as natural deduction, sequent calculus and Hilbert-style (axiomatic) systems. With his completeness theorem, Gödel proved that deductive systems for FOL can be sound and complete (Gödel 1929; Gödel 1930). Soundness of the logic with respect to a deductive system D means that only valid formulas are provable by D. Completeness, which is much harder to demonstrate, states that only formulas that are provable by D are valid. These are important metalogical properties, which make FOL a convenient system for many purposes that do not require total expressivity. This will be an important consideration in Chapter 4, where FOL is used to generate a new data set.

Another metalogical feature is decidability: is there an effective method to determine the (in)validity of an arbitrary FOL formula? This question is known as the *Entscheidungsproblem*. Church and Turing independently established that it must be answered negatively (Church 1936; Turing 1936). FOL is undecidable, because there exists no decision procedure for determining the (in)validity of arbitrary formulas in a FOL language containing at least one n-ary predicate such that $n \geq 2$. For the fragment of FOL that uses only unary or nullary predicates an effective method does exist, so if the logic is thus restricted it is decidable.[2]

In Chapter 4 I will construct a data set containing pairs of sentences labelled according to their FOL entailment relation. In order to do so, I will make use of an automated theorem prover and model builder: Prover9 and Mace4 (McCune 2010). Given a premise, a set of axioms and a hypothesis, Prover9 pursues a proof by contradiction. Statements that are derivable from the assumptions are compared to the negated hypothesis. If this leads to a contradiction, Prover9 concludes that the hypothesis must hold. Derivation is governed by the following rules:

[2]For a more in-depth explanation of FOL and its metalogical properties, see e.g. Mendelson 1987, Boolos, Burgess, and Jeffrey 2007 or Sider 2010.

- *Clausification.* Conversion from quantified formulas into unquantified clauses. For existentially quantified formulas this amounts to instantiation with a term. Universally quantified variables are substituted with free ones.
- *Resolution.* Application of the resolution rule to lines of the form $P(x) \vee Q_1(x) \vee \cdots \vee Q_n(x)$ (with free variable x) and $\neg P(t)$ (with term t):

$$\frac{P(x) \vee Q_1(x) \vee \cdots \vee Q_n(x), \neg P(t)}{Q_1(t) \vee \cdots \vee Q_n(t)} \text{ resolution}[t/x]$$

- *Denial.* Negation of previously encountered expressions.

Model builder Mace4 addresses entailment semantically, and tries to find a countermodel. That is, it attempts to construct a model $\mathcal{M}$ (as specified above) that satisfies the premise and all axioms, while falsifying the hypothesis. It does so by checking models of all domain sizes up to some n. If a countermodel is identified, the argument is concluded to be invalid.

A sample trace produced by Prover9 and Mace4 is provided in Appendix A. It illustrates that searching proofs using this method is computationally expensive and prone to combinatorial explosion as the number of axioms increases. The algorithms continue to apply admissible rules until a contradiction or a countermodel is found, or until the maximum recursive depth or time is exceeded. For these reasons, in addition to the fact that all input sentences have to be parsed, automated theorem proving of this kind faces serious scalability issues.

2.3 Neural networks

Most of the models used in this thesis are artificial neural networks, a class of machine learning algorithms that combine linear transformations with nonlinear activation functions in order to approximate the conditional distribution of a set of classes given vectorized input. There are many different kinds of neural networks, so this section serves to introduce only some of their most basic properties. As a simple example, let us consider a so-called *feed-forward neural network* and its defining characteristics.

Suppose that we have a data set $\mathcal{D}$, which consists of pairs of input vectors $\mathbf{x}_i$ and corresponding targets t_i. $\mathcal{D}$ consists of a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$. We want a neural network to predict the labels t of the data instances. To achieve this, the network is trained on the items in $\mathcal{D}_{train}$, and tested on the unseen items in $\mathcal{D}_{test}$.

Figure 2.4 shows a small feed-forward neural network, which we call $\mathcal{N}$, as an example for four-dimensional input vectors $\mathbf{x}_i$, a five-dimensional hidden layer and three output classes. If data instance $\mathbf{x}_i$ is presented to $\mathcal{N}$, the corresponding hidden layer vector $\mathbf{h}_i$ is computed as follows:

$$\mathbf{h}_i = f(\mathbf{W}_h \times \mathbf{x}_i + \mathbf{b}_h), \tag{2.1}$$

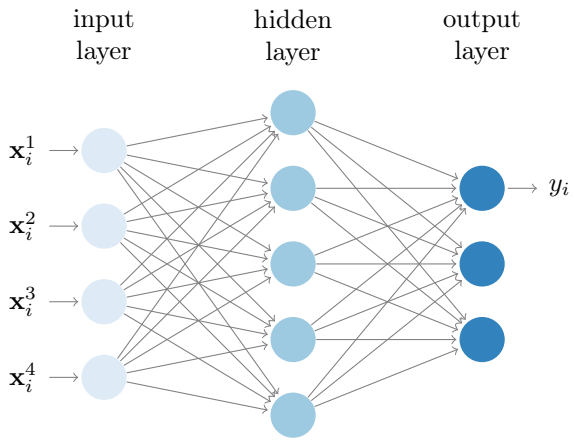


Figure 2.4: Schematic visualization of the small feed-forward neural network $\mathcal{N}$.

where $\mathbf{W}_h$ is the input-to-hidden weight matrix (c.q. of dimensions 5×4) and $\mathbf{b}_h$ is a bias term. f is a nonlinear activation function, e.g. tanh. Next, to move from the hidden layer to the output layer, a similar computation is performed on $\mathbf{h}_i$:

$$\mathbf{y}_i = \mathbf{W}_o \times \mathbf{h}_i + \mathbf{b}_o, \tag{2.2}$$

where $\mathbf{y}_i$ denotes the output vector, which is three-dimensional for this three-class classification problem. $\mathbf{W}_o$ is the hidden-to-output weight matrix (in this example of dimensions 3×5) and $\mathbf{b}_o$ is a bias term. It is common to apply a softmax function to $\mathbf{y}_i$ in order to represent the output as a probability distribution. Generally, the softmax of $\mathbf{y}_i^j$, the jth entry of $\mathbf{y}_i$, is given by:

$$\text{softmax}(\mathbf{y}_i^j) = \frac{e^{\mathbf{y}_i^j}}{\sum_{j'} e^{\mathbf{y}_i^{j'}}} \tag{2.3}$$

The index of the entry of $\mathbf{y}_i$ with the highest softmax value can then be returned as the predicted label y_i for input $\mathbf{x}_i$:

$$y_i = \underset{j}{\text{argmax}}(\text{softmax}(\mathbf{y}_i^j)) \tag{2.4}$$

In the example of Figure 2.4, the upper node of the output layer yields the highest softmax value and is therefore returned as the predicted label.

The aim is to predict the target value of the input, i.e. $y_i = t_i$. In order to achieve this, the parameters in the (randomly initialized) weight matrices $\mathbf{W}_h$, $\mathbf{W}_o$ and bias vectors $\mathbf{b}_h$, $\mathbf{b}_o$ must be optimized. This can be done during the training phase with a method called *backpropagation*, which updates the network parameters by differentiating an objective function. This function, also known as the loss function, expresses the deviance between the output of $\mathcal{N}$ and the training target.

A popular objective function is the negative log likelihood (NLL). Let $\mathbf{z}_i$ denote the vector containing the softmax output for each class (i.e. $\mathbf{z}_i^j = \text{softmax}(\mathbf{y}_i^j)$ for each index j). For target t_i, let $\mathbf{t}_i$ denote the corresponding target vector, which is a one-hot vector with the 1-entry at the index of the correct class. Then the NLL loss is given by:

$$E(\mathbf{z}_i, \mathbf{t}_i) = -\sum_j \log(\mathbf{z}_i^j)\mathbf{t}_i^j \tag{2.5}$$

Additionally, L2-regularization can be applied to prevent overfitting. This is the phenomenon occurring when a model fits the training data too closely to be capable of successful generalization to unseen data. A model is at particular risk of overfitting when its learned parameter values become excessively high. L2-regularization seeks to prevent this effect by penalizing large coefficients. It does so by extending the loss function with a weight decay term:

$$E_r(\mathbf{z}_i, \mathbf{t}_i) = E(\mathbf{z}_i, \mathbf{t}_i) + \lambda \sum_j w_j^2 \tag{2.6}$$

Here, $E(\mathbf{z}_i, \mathbf{t}_i)$ is a loss function, c.q. the NLL of Equation (2.5). The quantities w_j represent the model weights and λ is the L2 penalty term. The higher λ, the more cost is associated with large parameter values.

During the training phase, the aim is to minimize the error as computed by Equation (2.6) for any output-target pair $\mathbf{z}_i, \mathbf{t}_i$. This is done by using the outcome of the objective function as an error signal for parameter updating with a differential optimization algorithm. Stochastic Gradient Descent (SGD) is the most basic such method, which updates the network weights w_i as per the following equation:

$$w_i^{n+1} = w_i^n - \eta \nabla_{w_i} E_r, \tag{2.7}$$

where w_i^n represents the value of parameter w_i at time n, η is the learning rate and $\nabla_{w_i} E_r$ is the gradient of the objective function E_r with respect to w_i. During training, the network iterates over the training set several times. One such iteration is called an 'epoch', and starts with reshuffling the data instances. At testing time the learned parameters are frozen.

Chapter 3

Quantified Natural Logic Inference

This chapter serves to introduce the Quantified Natural Logic Inference task as addressed by Bowman 2013 and Bowman, Potts, and Manning 2015. Earlier results are reported, as well as those obtained by means of a replication. Consecutively, some major shortcomings of this type of research are addressed.

3.1 Bowman's research

In recent years, one of the most influential studies on textual entailment recognition was led by Bowman. His 2013 paper (Bowman 2013), tentatively titled 'Can recursive neural tensor networks learn logical reasoning?', was soon followed by a more decisive publication: 'Recursive Neural Networks Can Learn Logical Semantics' (Bowman, Potts, and Manning 2015). The claim made in the second title is supported by the high scores that recursive neural networks obtained in classification tasks based on the semantics of propositional and natural logic. Here, I focus on the more advanced natural logic task.

3.1.1 Data generation

Natural logic, as introduced in Section 2.2.1, is a calculus comprising inference rules between words and phrases in natural language. Because it operates directly on unformalized utterances, no translations to a particular logical syntax are required in order to assess the relation between expressions. Bowman uses the version of Natural Logic specified by MacCartney and Manning (MacCartney and Manning 2009) to automatically generate data containing pairs of sentences, labelled with their logical relation.

Prior to the data generation process, a small toy language is constructed. Let this language be denoted by $\mathcal{L}_B$ (with B for Bowman). The vocabulary of $\mathcal{L}_B$ consists of four classes: quantifiers, nouns, intransitive verbs and adverbs. Let these classes be represented by $\mathcal{Q}^{\mathcal{L}_B}$, $\mathcal{N}^{\mathcal{L}_B}$, $\mathcal{V}^{\mathcal{L}_B}$, $\mathcal{A}^{\mathcal{L}_B}$, respectively. They contain the following words:

$$\mathcal{L}_B \begin{cases} \mathcal{Q}^{\mathcal{L}_B} = \{\texttt{all}, \texttt{some}, \texttt{most}, \texttt{two}, \texttt{three}, \texttt{not\_all}, \texttt{no}, \texttt{not\_most}, \texttt{lt\_two}, \texttt{lt\_three}\} \\ \mathcal{N}^{\mathcal{L}_B} = \{\texttt{warthogs}, \texttt{turtles}, \texttt{mammals}, \texttt{reptiles}, \texttt{pets}\} \\ \mathcal{V}^{\mathcal{L}_B} = \{\texttt{walk}, \texttt{swim}, \texttt{move}, \texttt{growl}\} \\ \mathcal{A}^{\mathcal{L}_B} = \{\texttt{not}, \epsilon^1\} \end{cases}$$

Individual sentences can be generated combinatorially from the above classes according to the phrase structure grammar in Table 3.1 (Chomsky 1957). The statements X $\rightarrow \mathcal{X}$ in the right column summarize all production rules connecting a non-terminal X to the terminals in class $\mathcal{X}$. That is, X $\rightarrow \mathcal{X}$ abbreviates the set of rules $\{\text{X} \rightarrow \texttt{x} \mid \texttt{x} \in \mathcal{X}\}$.

S $\rightarrow$ NP VP	Det $\rightarrow \mathcal{Q}^{\mathcal{L}_B}$
NP $\rightarrow$ Det NP	N $\rightarrow \mathcal{N}^{\mathcal{L}_B}$
NP $\rightarrow$ Adv N	V $\rightarrow \mathcal{V}^{\mathcal{L}_B}$
VP $\rightarrow$ Adv V	Adv $\rightarrow \mathcal{A}^{\mathcal{L}_B}$

Table 3.1: Phrase structure grammar for artificial language $\mathcal{L}_B$.

In the data, sentences are formed by constituent words together with the underlying syntactic structure, which takes the shape of a binary tree and is represented by the use of brackets. E.g.: '((all warthogs) walk)'. The set $\mathcal{S}_{\mathcal{L}_B}$ of all sentences in $\mathcal{L}_B$ can also be expressed as the Cartesian product $\mathcal{Q}^{\mathcal{L}_B} \times \mathcal{A}^{\mathcal{L}_B} \times \mathcal{N}^{\mathcal{L}_B} \times \mathcal{A}^{\mathcal{L}_B} \times \mathcal{V}^{\mathcal{L}_B}$. Evaluating the cardinality of $\mathcal{S}_{\mathcal{L}_B}$ shows that there are $10 \times 2 \times 5 \times 2 \times 4 = 800$ different valid sentences.

Note that class $\mathcal{A}^{\mathcal{L}_B}$ contains the expression `not`, for logical complement, and the dummy term ϵ, representing an empty string. Including ϵ as an adverb is useful because it facilitates a more concise description of the language. This is due to the fact that negation is allowed, but not required, at two positions: in front of the noun and in front of the verb.

The set $\mathcal{Q}^{\mathcal{L}_B}$ can be divided into the 'positive' quantifiers `all`, `some`, `most`, `two`, `three` and their negated counterparts `not_all`, `no`, `not_most`, `lt_two`, `lt_three`. (The prefix `lt` for the numeric quantifiers `two` and `three` should be read as 'less than'.) Because the negated versions are not decomposed, a model can effectively learn to apply negations specifically tailored for particular quantifiers.

The classes $\mathcal{N}^{\mathcal{L}_B}$ and $\mathcal{V}^{\mathcal{L}_B}$ contain non-logical symbols whose lexical meaning is captured relationally by a taxonomy of terms. This is a hierarchy denoting the set-theoretic relations between the different concepts in a domain of discourse, c.q. some fragment of the animal world. The hierarchy for $\mathcal{N}^{\mathcal{L}_B}$ adopted by Bowman is shown in the Venn diagram of Figure 3.1. The one for $\mathcal{V}^{\mathcal{L}_B}$ is shown in Figure 3.2. If a region A is fully contained by another region B, the corresponding terms relate as subset and superset,

[1] ϵ denotes the empty string.

respectively. Partial overlap denotes independence. Absence of any overlap between regions indicates that the corresponding terms cannot apply together, i.e. that they are mutually exclusive.

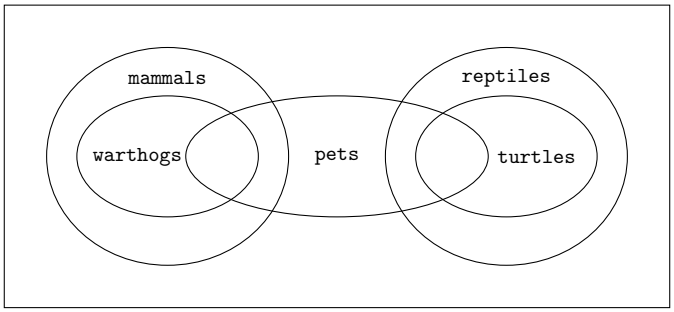


Figure 3.1: Venn diagram visualizing the taxonomy of nouns $\mathcal{N}^{\mathcal{L}_B}$ in $\mathcal{L}_B$.

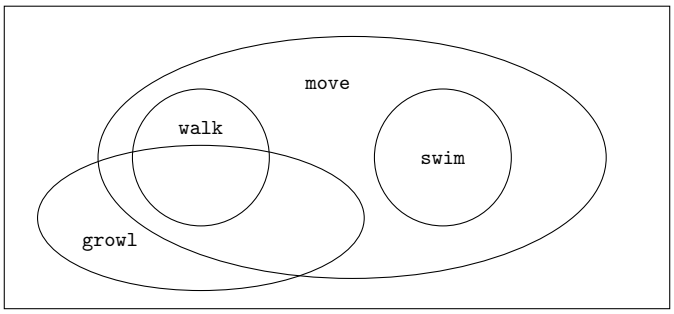


Figure 3.2: Venn diagram visualizing the taxonomy of verbs $\mathcal{V}^{\mathcal{L}_B}$ in $\mathcal{L}_B$.

These ontologies can be regarded as a knowledge base. Together with the calculus outlined in Section 2.2.1, they determine which natural logic inferences are allowed. Bowman uses this system to establish the entailment relation between pairs of sentences, according to the relations defined in Table 2.1 and visualized in Table 2.1. With the lexical relations between the different vocabulary items in place, Bowman's implementation of the natural logic calculus is used to generate data. A sample looks as follows:

```
<  ( ( all warthogs ) walk )  ( ( all warthogs ) move )
#  ( ( lt_two pets ) ( not growl ) )  ( ( two turtles ) ( not growl ) )
|  ( ( three turtles ) ( not walk ) )  ( ( lt_three ( not warthogs ) ) ( not walk ) )
<  ( ( all reptiles ) ( not walk ) )  ( ( most ( not warthogs ) ) ( not walk ) )
>  ( ( lt_three turtles ) ( not swim ) )  ( ( lt_three turtles ) growl )
```

Five different data sets are generated. Each of them contain a train and a test set, which are mutually exclusive not only with respect to pairs of sentences, but also with respect to single sentences. This is guaranteed by first randomly partitioning the total set of possible, individual expressions into training and testing instances. None of the training sentences are included in any test set pairs and vice versa. Hence, no

test sentences can be seen during training. Sentence pairs with relation 'unknown' are omitted from the data.[2]

Bowman's training sets contain some 27,000 pairs on average, and his test sets some 7,000 pairs. As his toy language $\mathcal{L}_B$ allows for 800 different sentences, there are $800 \times 800 = 640,000$ unique sentence combinations. This means that each training set contains approximately 4.2% of all possible pairs.

3.1.2 Models

Recursive models Bowman uses compositional distributional models to address the task described in the previous section. See Section 2.1 for a general description of such models and the underlying assumptions. Meaning is progressively constructed, starting with single words, and gradually moving upwards to a representation of complete sentences by the repeated application of some composition function. This process is guided by the syntactic form of the input sentences. Because sentences in the data are represented as binary trees, it is this recursive structure that dictates the topology of the models. Therefore, they are called 'recursive' or 'tree-shaped' networks. Notable examples of studies that examined comparable tree-shaped models are Socher, Karpathy, et al. 2014, Irsoy and Cardie 2014, Le and Zuidema 2015 and Tai, Socher, and Manning 2015.

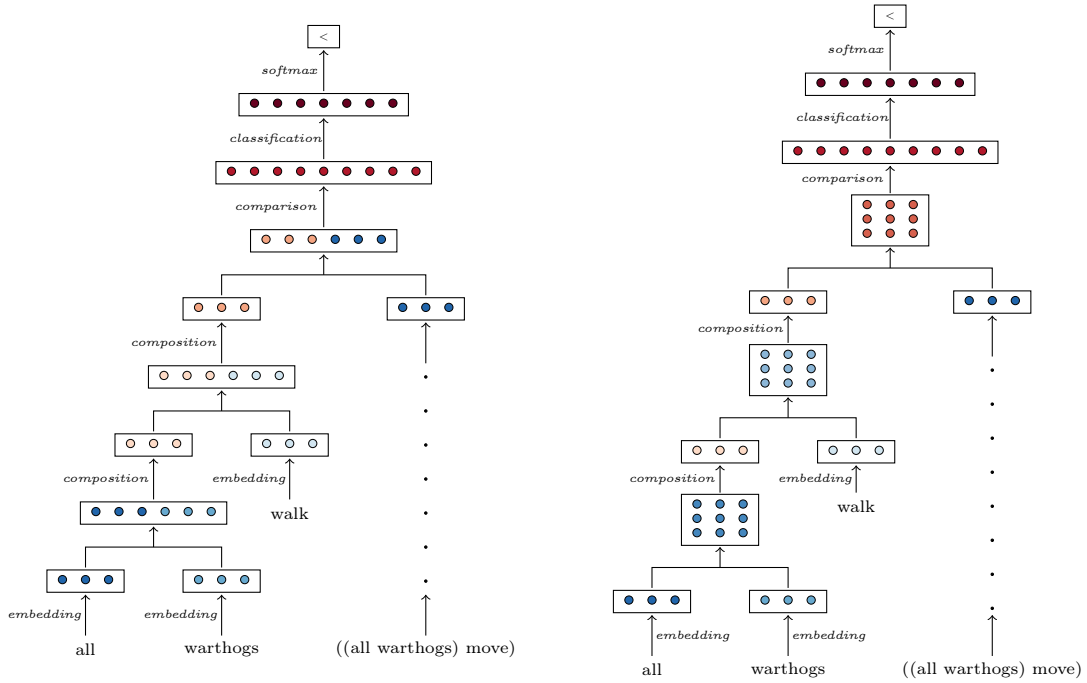
Figure 3.3 provides a schematic visualization of the architecture characterizing Bowman-style recursive networks. Two types of neural compositional distributional models are introduced: the tree-shaped Recursive Neural Network (tRNN, shown in Figure 3.3a) and the tree-shaped Recursive Neural Tensor Network (tRNTN, shown in Figure 3.3b). Their topology is largely identical, but the parameter space is different, as will be explained below in more detail.

The input consists of a pair of sentences from the data. First, all member words are mapped to trainable word embeddings. Next, the two sentences are separated and processed individually by identical recursive networks. 'Identical' is meant to say that the networks contain the same set of parameters. This type of architecture is also known as a 'Siamese network' (Mueller and Thyagarajan 2016a). The specific recursive topology, however, differs from sentence to sentence, depending on the syntactic structure of the individual input phrases. Once both sentences have been processed, their final representations are combined and passed to a comparison layer. Finally, a softmax classifier is used to determine the most likely logical relation for the input pair.

In the illustration, the forward-pass is visualized for an input pair whose left sentence is '((all warthogs) walk)', and whose right sentence '((all warthogs) move)'. The process is only shown for the left sentence, but takes place in exactly the same fashion for the right one.

First, the individual words at the leaf nodes are mapped to n-dimensional word embeddings. These embeddings are the result of a linear transformation. Before the

[2] Note the difference between the relations 'unknown' and 'independent'. If there is conclusive evidence that none of the other relations hold, 'independent' is assigned. If nothing at all can be concluded, which is frequently the case in this natural logic system, the label becomes 'unknown'.



(a) tRNN

(b) tRNTN

Figure 3.3: Schematic visualization of the two types of recursive networks: the tree-shaped Recursive Neural Network (tRNN), also referred to as the matrix model, and the tree-shaped Recursive Neural Tensor Network (tRNTN), sometimes abbreviated as the tensor network.

actual training begins, the model reads the available data in order to create a vocabulary, which contains all the words occurring in the training corpus. These words are indexed, so that they can be mapped to corresponding one-hot vectors. Generally, for a sentence S containing words w_1 to w_n, we can represent the one-hot encoding of a member w_i as $\mathbf{h}_i$. The n-dimensional embedding $\mathbf{e}_i$ for this word is then determined as follows:

$$\mathbf{e}_i = \mathbf{M}_{emb} \times \mathbf{h}_i, \tag{3.1}$$

where $\mathbf{M}_{emb}$ denotes the embedding matrix. This matrix has dimensions $n \times V$, where n is the desired dimensionality of the embeddings, and V is the vocabulary size, i.e. the number of unique words occurring in the training data. As there are V unique words, the one-hot encodings $\mathbf{h}_i$ are sparse column vectors with V entries.

Next, the composition stage takes place. This process is structured according to the parse trees of the individual input phrases. In Figure 3.3, the parse tree for '((all warthogs) walk)' requires that 'all' and 'warthogs' be composed together, the result of which is composed with 'walk' to obtain the complete sentence vector. Generally, there are no restrictions on sentence length or recursive depth, because the network topology is automatically adapted in accordance with any binary parse tree.

A learned composition function takes pairs of vectors as input, and returns an output with the same dimensionality as the argument vectors. The input vectors can be word embeddings, results of earlier compositions or a combination of both. Two different composition functions are implemented. The one is a standard neural layer function using only matrix multiplication, the other uses the more advanced notion of tensor multiplication. Which of these two options is adopted determines the difference between the regular tRNN of Figure 3.3a and the tRNTN of Figure 3.3b. Let $\mathbf{e}_i$, $\mathbf{e}_j$ be the n-dimensional column vector representations for the left and right child of some composition node (e.g. those for 'all' and 'warthogs' at the first composition node in Figure 3.3). Let function $f(x)$ denote the nonlinearity $\tanh(x)$. Then the composition function produces the result $\mathbf{c}_{tRNN}$ for the matrix model or $\mathbf{c}_{tRNTN}$ for the tensor model:

$$\mathbf{c}_{tRNN} = f(\mathbf{M}_{cps} \times \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_j \end{bmatrix} + \mathbf{b}_{cps}) \tag{3.2}$$

$$\mathbf{c}_{tRNTN} = \mathbf{c}_{tRNN} + f(\mathbf{e}_i^\intercal \times \mathbf{T}_{cps} \times \mathbf{e}_j) \tag{3.3}$$

Here, $\mathbf{M}_{cps}$ is the composition matrix of dimensions $n \times 2n$, which is multiplied with the $2n$-dimensional concatenation of $\mathbf{e}_i$ an $\mathbf{e}_j$. $\mathbf{b}_{cps}$ is a bias vector. As shown in Equation (3.3), the tRNTN model applies the tRNN composition function of Equation (3.2) and adds the tensor product of the input vectors with the $n \times n \times n$-dimensional tensor $\mathbf{T}_{cps}$ (following Chen et al. 2013).[3] The result of this computation is another n-dimensional vector. Matrix and tensor multiplication model the interactions of child vectors in different ways. Matrix multiplication, on the one hand, treats them additively, by taking a concatenation of vectors as the only input. Tensor multiplication, on the other hand, provides a way of modelling the multiplicative interactions between the argument vectors, too. Furthermore, the tensor $\mathbf{T}_{cps}$ contains $n \times n \times n$ weights, whereas the matrix $\mathbf{M}_{cps}$ only contains $n \times 2n$ weights. Hence, for any $n > 2$, the tensor encodes more information than the matrix, and is therefore a more powerful tool (at the expense of longer training times).

The difference between both routines is illustrated in Figure 3.3. The tRNN, in Figure 3.3a, concatenates child vectors before applying the composition function. The tRNTN, in Figure 3.3b, first computes the $n \times n$-dimensional Kronecker product of the child vectors, which is flattened to form an n^2-dimensional vector.[4] The $n \times n \times n$-dimensional tensor $\mathbf{T}_{cps}$ can be transformed to a regular $n \times n^2$-dimensional matrix. Multiplying this matrix with the flattened Kronecker product of the child vectors at the composition stage returns an n-dimensional output vector and is equivalent to directly taking the

[3] See Hackbusch 2012 or Kolda and Bader 2009 for an introduction to tensor algebra.

[4] For n-dimensional child vectors $\mathbf{e}_i$, $\mathbf{e}_j$, the Kronecker product is interpreted as follows:

$$\mathbf{e}_i \otimes \mathbf{e}_j^\intercal = \begin{bmatrix} \mathbf{e}_i^1 \mathbf{e}_j^\intercal \\ \vdots \\ \mathbf{e}_i^n \mathbf{e}_j^\intercal \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i^1 \mathbf{e}_j^1 & \dots & \mathbf{e}_i^1 \mathbf{e}_j^n \\ \vdots & \ddots & \vdots \\ \mathbf{e}_i^n \mathbf{e}_j^1 & \dots & \mathbf{e}_i^n \mathbf{e}_j^n \end{bmatrix}$$

This matrix can be flattened to produce an n^2-dimensional row vector.

tensor product of Equation (3.3). Note that Figure 3.3b does not visualize the complete tRNTN composition, because this requires addition with the result of the regular tRNN composition function.

The composition function is repeatedly applied, until all branches have been collapsed and an entire sentence is represented as an n-dimensional vector. This process takes place for both the left and the right sentence, so that eventually, the two sentence vectors can be concatenated into a single $2n$-dimensional vector. This representation is presented to an m-dimensional comparison layer. Let (S, T) denote a pair of input sentences, with final vector representations $\mathbf{s}_S, \mathbf{s}_T$. Then the comparison layer performs one of the following transformations to compute the output vector $\mathbf{p}_{tRNN}$ or $\mathbf{p}_{tRNTN}$, depending on whether the network is a tRNN or a tRNTN:

$$\mathbf{p}_{tRNN} = g(\mathbf{M}_{cpr} \times \begin{bmatrix} \mathbf{s}_S \\ \mathbf{s}_T \end{bmatrix} + \mathbf{b}_{cpr}) \tag{3.4}$$

$$\mathbf{p}_{tRNTN} = \mathbf{p}_{tRNN} + g(\mathbf{s}_S^\intercal \times \mathbf{T}_{cpr} \times \mathbf{s}_T) \tag{3.5}$$

Here, $\mathbf{M}_{cpr}$ is the comparison matrix of dimensions $m \times 2n$, which is multiplied with the $2n$-dimensional concatenation of $\mathbf{s}_S$ and $\mathbf{s}_T$. $\mathbf{b}_{cpr}$ is a bias vector. $\mathbf{T}_{cpr}$ is the $n \times m \times n$ comparison tensor. Equation (3.4) is applied by the matrix model, and (3.5) by the tensor model. The functions are essentially the same as the ones used at the composition stage, but of course the parameters are learned independently. Obviously, the dimensionality of both layers can also differ, in which case $m \neq n$. Usually, the comparison layer has a higher dimensionality than the embeddings, implying $m > n$. A different nonlinearity function is adopted. Instead of $f(x) = \tanh(x)$, the leaky rectified linear function $g(x)$ is used (Maas, Hannun, and Ng 2013). Bowman reports that this function gives better results at the comparison stage than a tanh nonlinearity.

$$g(x) = \max(x, 0) + 0.01 \min(x, 0) \tag{3.6}$$

Following the comparison layer, classification takes place. For this purpose, the vector produced by either Equation (3.2) or (3.3) must be transformed so as to form a new vector whose dimensionality matches the number of different classes in the task. There are currently seven different classes, namely the possible entailment relations of Table 2.1 and Figure 2.3. Hence, the m-dimensional output of the comparison layer must be processed by a classification layer with 7-dimensional vectors as output. Let $\mathbf{p}$ represent the output of the comparison layer for an arbitrary recursive network. Then the classification layer function outputs the vector $\mathbf{y}$:

$$\mathbf{y} = \mathbf{M}_{class} \times \mathbf{p} + \mathbf{b}_{class} \tag{3.7}$$

$\mathbf{M}_{class}$ denotes the classification layer matrix of dimensions $7 \times m$, and $\mathbf{b}_{class}$ the bias vector for this final layer. At this stage, no distinction is made between matrix and tensor models. Finally, the softmax function is applied to the last layer output $\mathbf{y}$ in order to represent the vector as a probability distribution and determine the most likely output class (see Section 2.3). In Figure 3.3, this output is '$<$' (forward entailment) for

the input sentences '((all warthogs) walk)' and '((all warthogs) move)'. This is correct, because indeed the left sentence implies the right one, but not vice versa.

Summing baseline In addition to the recursive models described in this section so far, a simple baseline model is implemented. This is a summing neural network based on a unweighted vector mixture model, abbreviated 'sumNN'. Its architecture is identical to the one described above, but differs in one crucial aspect: instead of using a learned composition function that is recursively applied to the constituents of complex expressions, the embeddings of member words are summed. This means that the sumNN architecture is visualized by Figure 3.3a, if only the steps labelled 'composition' are omitted and replaced by a simple summation of the word embeddings. Technically, the final representation $\mathbf{s}_S$ of a single sentence S containing words w_1 to w_n then becomes:

$$\mathbf{s}_S = \sum_{i=1}^{n} \mathbf{e}_i, \tag{3.8}$$

where $\mathbf{e}_i$ is the embedding of word w_i with one-hot encoding $\mathbf{h}_i$, following Equation (3.1). Due to the commutativity of summation, the sumNN is not sensitive to word order and hierarchy. In 'natural' contexts this should severely disadvantage the baseline, but due to the rigid syntax of the toy language $\mathcal{L}_B$, the only factor that could currently confuse the model is the location of a single negation (allowed both in front of the noun and the verb). The disregard of order in the word input qualifies this baseline as a bag-of-words model.

For all models, the training regime is mostly as described in the example of Section 2.3. NLL loss is used as an objective function and L2-regularization is applied to prevent overfitting. The optimization method is SGD, together with AdaDelta for the adaptive computation of the learning rate (Zeiler 2012).

3.1.3 Results and replication

Bowman applies five fold cross-validation to generate sets of train and test data that are called `f1` to `f5`. A partitioning is made between *single* sentences, so that no sentence seen during training time is encountered during testing and vice versa. All three models (sumNN, tRNN and tRNTN) are trained on the different training data. Per fold, five runs are performed, so that 75 different models are trained in total. Training and testing accuracy are averaged per model over all runs and folds.

Parameters are initialized by drawing from a uniform distribution. For layer parameters (composition, comparison and classification matrices), the range is $(-0.05, 0.05)$. For the word embeddings it is $(-0.01, 0.01)$.[5] Although this is not explicitly mentioned, it is to be expected that biases are initialized as zero vectors, as is common practice.

[5]Slightly better results are obtained with Xavier initialization, but to keep all training conditions in the replication as similar to Bowman's as possible, only uniform initialization with the given parameters is applied.

The comparison layer dimensionality is set to 75, and for the word embeddings it is kept constant at 25.

Assuming that the same regularization coefficients are used for all experiments described in Bowman, Potts, and Manning 2015, the L2-penalty for the tRNN is $\lambda = 0.001$, and for for the tRNTN $\lambda = 0.0003$. No weight decay term for the baseline is reported, so here I assume that for the sumNN, also $\lambda = 0.0003$. Manual tuning shows that this is a reasonable choice.

It is not clear for how many epochs the models are trained, or what stopping condition is maintained. It is possible that Bowman trained the models for a fixed number of epochs, and reported accuracy scores at the very end. However, it is also possible that the model with the best testing accuracy at some earlier epoch was selected. Due to the fluctuating performance after stabilization of a trained model, it is often the case that better scores are obtained at epochs (shortly) preceding the final one.

For the current project, the sumNN, tRNN and tRNTN models are reimplemented using the PyTorch machine learning library (Paszke et al. 2017).[6] The different architectures are exactly as described in the above. The hyperparameter values and parameter initializations are also identical to Bowman's, insofar as this information could be inferred from the literature. A possible difference is the adopted training regime, because the number of epochs is fixed at 50 for all replicated experiments. All reported results are obtained after the 50th training epoch, at which point the learning is terminated. On CPUs, training time per run in the new implementation is below 10 minutes for the sumNN, approximately 1.5 hours for the tRNN and slightly longer than 2 hours for the tRNTN. The long training times for the tree-shaped models are largely due to the fact that the recursively applied composition function makes efficient batching impossible. The network topology differs from sentence to sentence, and the composition arguments often depend on the results of preliminary compositions. Because of these variations and interdependencies, the models cannot fully profit from the significant speed-ups associated with large-scale matrix manipulations.

Table 3.2 shows the performance of the different models on the quantified natural logic inference task. The results reported by Bowman are included, as well as those obtained in the PyTorch replication. Performance is expressed in terms of accuracy, which is the percentage of correctly predicted classes for some collection of instances. Accuracy scores are provided for train and test sets. As noted, they are averaged with respect to five runs on five folds for each model.

It is clear from Table 3.2 that the tRNN and the tRNTN both perform very well on the task, and that the tRNTN reaches almost perfect scores. In fact, even the sumNN obtains seemingly decent results. The scores reported by Bowman and those obtained in the current replication are definitely in the same ballpark. The relative differences between training and testing accuracy and those between different models are also very similar. Yet, it must be noted that the replication scores are consistently lower than Bowman's results. On average, the difference is smaller than one percentage point, and

[6] The full replication code, together with all other scripts written for this thesis, is available at www.github.com/MathijsMul/mol-thesis.

	Bowman		replication	
	train	test	train	test
25/75 sumNN	96.9	93.9	95.0	93.0
25/75 tRNN	99.6	99.2	98.8	98.1
25/75 tRNTN	100	99.7	99.7	99.1

Table 3.2: Training and testing accuracy scores on the quantified natural logic inference task, as reported in Bowman, Potts, and Manning 2015 and obtained by own replication. An n/m model has n-dimensional word embeddings and an m-dimensional comparison layer. Results are averaged over five runs on five folds for each model.

there is a variance across runs of two percentage points at most, but the generality of this observation and the large number of experimental runs exclude the possibility that this effect is due to chance. Most probably it is caused by differences between hyperparameter settings, number of training epochs or stopping conditions. As mentioned above, superior models are often available if one takes previous states of the network into consideration. Stagnation of the loss may occur at a relatively early stage, after which there remains only some minor oscillation. In the replication, evaluation always happens after epoch 50, even if the final model is outperformed by some predecessor. This could partially explain the slightly lower scores in Table 3.2.

Figure 3.4 shows the development of the accuracy score on the test set of fold `f1` for a single run of all three models on the training data of `f1`. Although this plot is based on only one training session per model, its rough characteristics are still representative because the variance between different runs is very limited. The graph clearly shows how stabilization of the testing accuracy sets in around the 30th epoch for the tRNN, and already around the 15th epoch for the tRNTN. The baseline produces a less abrupt asymptote, but does not obtain significant improvement after the 30th epoch anymore either. It is interesting to see that the tRNTN soon outperforms the baseline, and very quickly reaches its peak performance, whereas the tRNN only improves on the sumNN at a much later epoch. Apparently the tensors require little training to obtain good results. During the initial epochs, the baseline is on a par with the recursive models, which is explained by the fact that summing word embeddings is a more sensible composition method than multiplication with random, untrained matrices.

We are dealing with a logical classification task that can be solved without errors by the same algorithmic implementation of the natural logic calculus that was used to generate the data, so even if there are few errors, they must be inspected with extra care. Any potential pattern in the error profile could clarify something about the particular weaknesses of the models, or reveal which fragments of the data are particularly challenging. Therefore, Figure 3.5 shows the confusion matrices with respect to the `f1` test data for the fully trained models whose accuracy development was visualized in Figure 3.4. In these matrices, the rows represent the target labels, and the columns represent the classes assigned by the model in question. The entries in the rows are normalized, so that they can be interpreted as a probability distributions over predicted classes per

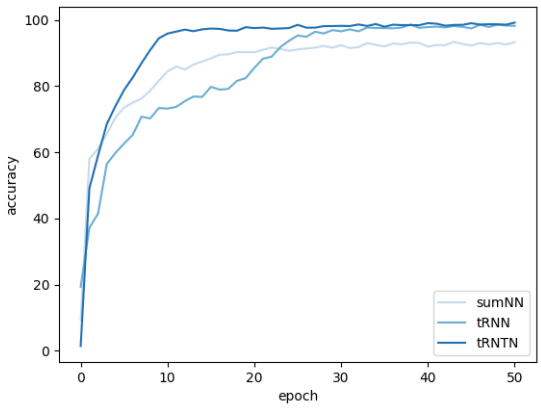
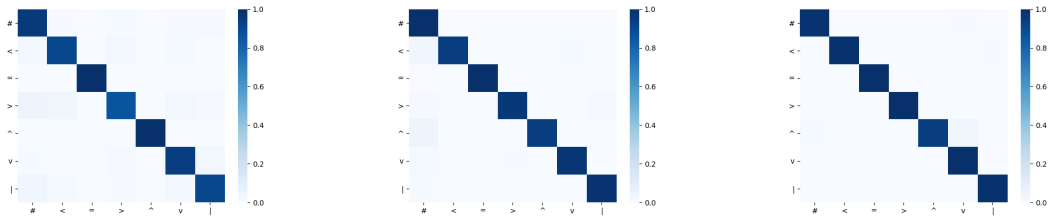


Figure 3.4: Accuracy development on the test set of fold `f1` for a single run on the training data of `f1`, for sumNN, tRNN and tRNTN.

actual class. The rows and columns are indexed according to the seven different classes, whose symbols are the same as those in Table 2.1.



(a) sumNN (b) tRNN (c) tRNTN

Figure 3.5: Confusion matrices with respect to the `f1` test data for the fully trained sumNN, tRNN and tRNTN models whose accuracy development is visualized in Figure 3.4. Rows represent targets, columns predictions.

Evidently, all models perform well enough to produce strongly diagonal confusion matrices. This indicates that most of the predictions for testing instances with a certain label are identical to that label. All non-zero entries other than those on the diagonal indicate misclassifications. The sumNN seems slightly biased towards forward entailment ($<$) and independence (#) in the case of backward entailment ($>$), while forward entailment is most often misclassified as backward entailment or independence. This can be explained by the strong analogy between the two kinds of entailment, and the high frequency of data pairs that are labelled with independence. In the `f1` training data, 30% of all sentence pairs has this target. For the tRNN, the only relevant observation seems to be the fact that negation ($\wedge$) is most often mistaken for independence. This is because negation is the rarest label, assigned to only 1% of the training instances of `f1`,

which can be easily overlooked and dominated by the prevalent class of independence. The confusion matrix for the tRNTN is almost perfectly diagonal, so no further remarks seem justifiable. In fact, all of these three models perform too well for their confusion matrices to reveal any surprising information, but at least they can serve to introduce this useful method of visualization.

3.2 Related research

At this point it is worthwhile to mention some follow-up research on Bowman's experiments:

- Veldhoen 2015 investigated whether recursive neural networks of the kind used by Bowman can actually learn a principled solution to the task of natural language inference, rather than applying an ad hoc approach with little generalizability. As a more easily interpretable analogue to the problem, she focused on arithmetic expressions. She proposed an interpretative method called *project-sum-squash*, which recognizes that during the composition stage, embeddings are only multiplied with one part of the composition matrix. In the case of arithmetic, this transformation reveals a meaningful geometry of the learned embeddings, suggesting that a principled solution has been found.[7]

- Veldhoen, Hupkes, and Zuidema 2016 continued this line of research. Not only tree-shaped networks were considered, but also sequential architectures with recurrent units. Gated Recurrent Units (GRUs) were used to obtain good results on the arithmetic toy data, and proved capable of generalization to longer statements than seen during training. The method of *diagnostic classification* was proposed to analyse the high-dimensional hidden units of the recurrent networks. See Section 5.4 for an explanation of this technique.

- Repplinger 2017 replicated Bowman's experiments, and complicated the task by decomposing negated quantifiers such as `not_some` into two separate words as: `not some`. He critically assessed Bowman's data generation process, focusing in particular on limitations of the implemented natural logic calculus. Using PCA and three-dimensional visualization, some meaningful hyperplane separations between learned expressions were observed.

3.3 Problems

In the preceding sections, the quantified natural logic task was introduced, as well as the models used by Bowman to address it, his results and the ones obtained by means of a

[7] As shown by Veldhoen 2015, tree-shaped networks can learn to perform addition and subtraction of the numbers $[-10, \cdots, 10]$, but the word embeddings are not neatly ordered. A spatial organization according to ordinality only manifests itself after multiplication with the relevant fragments of the composition matrix.

new replication. Given the task at hand, it is fair to conclude that the presented tree-shaped networks perform extremely well. This was reported by Bowman, and confirmed by the replication and the follow-up research of the preceding section.

However, if the aim is to investigate the extent to which such models are capable of determining semantic entailment relations, or even of performing logical reasoning, some doubts concerning the current task and the associated data are justified. Nine such concerns are mentioned below, and developed in more detail in Appendix B:

1. *Natural logic.* The appealing simplicity of the natural logic calculus comes at the expense of other properties. Although intended to operate on fully natural language, the use of brackets in the data makes the logic less natural than it is intended to be. Results can be unconclusive or uninformative. The implemented system does not allow reasoning with multiple premises and can only be used to express some quantifier-based inferences. Moreover, it is provably incomplete (e.g. De Morgan's laws cannot be established) and not provably sound.

2. *Implementation and data anomalies.* The data contain some anomalies that cannot be explained by MacCartney's natural logic calculus, which suggests that there are issues with Bowman's implementation.

3. *Simplistic syntax.* Sentences are short and their structure is extremely rigid. Other than negation, no words can be used twice in a sentence and only unary predicates are used.

4. *Symmetric hierarchy.* The ordering of nouns in $\mathcal{N}^{\mathcal{L}_B}$ is symmetric: `warthogs` and `turtles` relate in exactly the same way as `mammals` and `reptiles`. This decreasess the diversity of the taxonomy, which can make the classification task even less challenging.

5. *Composed quantifiers.* As noted by Repplinger 2017, the negated quantifiers in $\mathcal{Q}^{\mathcal{L}_B}$ are composed for no clear reason. This is an unnatural simplification of the problem.

6. *Relative size of training data.* A large fraction of the total data space is observed during each training epoch: more than 4% (see page 20).

7. *Baseline performance.* As shown in Table 3.2, even the simplistic summing baseline obtains decent results on the current task. The fact that the more advanced recursive networks only slightly outperform this baseline suggests that the task might be trivial.

8. *Symbolic prerequisites.* The recursive network topology depends on the availability of syntactic information about the input sentences. The need for this symbolic knowledge makes the models more artificial and less scalable.

9. *Lack of interpretation.* Bowman, Potts, and Manning 2015 hardly pay any attention to interpretation and analysis of the models.

Chapter 4

First-Order Entailment Recognition

Despite the shortcomings of Bowman's line of research, the task in itself - classifying logical entailment with neural models - remains as interesting and relevant as it was before. For this reason, the remainder of the current project addresses the same general problem, but with data and models that are chosen so as to avoid the issues identified in Section 3.3.

In this chapter, the task is updated and complicated in order to solve old problems and meet new requirements. An alternative lexical taxonomy is introduced and a new data generation process is specified. Consequently, experiments are performed on the tree-shaped models and the summing baseline. An in-depth analysis of the results thus obtained is provided at the end of the chapter.

4.1 Data

The majority of the issues that have to be addressed are related to the data. In particular, the natural logic calculus and the toy language $\mathcal{L}_B$ that were used to generate Bowman's data sets are responsible for most of the problems listed in Section 3.3. Therefore, the first two measures that have to be taken are the adoption of a different logical system, and the adaptation of the artificial language. These steps are described in the next sections.

4.1.1 New logic

Problems As mentioned in Section 3.3 and explained in detail on the pages 87-90, the main difficulties of the natural logic underlying the old data are that:

1. It is not applied to natural language.
2. Several edit sequences may be possible.
3. Results are often uninformative.
4. No reasoning with multiple premises is possible.

5. The system is provably incomplete.
6. It is not provably sound.
7. Only some quantifier-based inferences can be expressed.

These problems provide a strong incentive to adopt a different logic. For the quantified propositions that have been the focus so far, first-order predicate logic seems a logical choice. The aim remains to study the ability of neural models to perform compositional generalization and recognize different entailment relations in (a somewhat) natural language, so the final data will not be formal expressions of FOL. Instead, the data should resemble the Bowman-format. This format could be called 'semi-natural' because the sentences are comprehensible without any need for translation, while the use of brackets and lack of auxiliary verbs are definitely artificial aspects. Hence, if sentences in a pair are generated combinatorially on the basis of some phrase structure grammar, they must first be formalized in FOL to infer the entailment relation according to this logic. This will be done with an automated theorem prover and model builder. Afterwards, the phrases must be translated back into the format used for the data. The resulting classification task is called 'First-Order Entailment Recognition'.

Referring back to Section 2.2.2, where some important properties of FOL were explained, we can confirm that this system is hardly affected by any of the above-mentioned issues. As for problem 1, the entire discrepancy between a natural logic and an unnatural language has disappeared. Previously, natural logic operated on an artificial language, whereas its most fundamental purpose is to act directly on natural language statements. In Appendix B, the recommendation was made to either apply natural logic to fully natural expressions, or to substitute the system with a well-understood formal logic. Here, I choose the latter option. Natural logic is abandoned in favour of traditional FOL, which is not applied to the expressions in the data, but to an intermediate representation in the correct formal syntax. The second issue is completely irrelevant in the context of FOL. Instead of a monotonicity calculus that allows for different possible edit sequences, thereby producing divergent results, there are deductive systems that can be used to infer entailment relations according to the procedure discussed below. Different proofs may be available for a single case, but this never yields different results. Also, results cannot be uninformative in the sense of problem 3. A deterministic programme (Algorithm 4.1 on page 37) will be provided that outputs one of the seven entailment relations of Table 2.1 for each sentence pair. It first checks for all six relations other than independence, and if none of these hold, it concludes that the sentences must be independent. Situations yielding unions of relations are thus excluded, so nothing has to be labelled 'unknown'. As for issue 4, reasoning with multiple premises is trivial in FOL. Although this option is currently not needed, it is an important property for the reasons discussed on page 88. Problems 5 and 6 are solved by Gödel's completeness theorem, which states that there exist deductive systems for FOL that are both sound and complete (see Section 2.2.2). This means that FOL has formal systems for which all provable statements are also valid, while all valid statements are also provable. Surely, the systems that will be

used in this project fall within this class. As a consequence, when working with FOL, one never has to face problems of the kind encountered in natural logic when it comes to e.g. De Morgan's laws. The equivalences of page 88 can easily be established, as is the case for all other valid inferences that natural logic was unable to handle.

Quantifier 'most' The seventh and final objection on page 31 demands a more nuanced response. It is true that natural logic only describes a part of all possible quantifier-based inferences, as explained on page 89. Much of the compositional ambiguity occurring in natural language cannot adequately be captured by natural logic without the admission of supplementary machinery such as brackets. It is also true that many of the propositions that prove challenging to natural logic are straightforwardly expressible in FOL. The formal apparatus of FOL then serves to eliminate the interpretative confusion that arises so often in natural language. However, it is not true that FOL describes all quantifier-based inferences.

Indeed, there exist quantifiers that face great expressibility issues in FOL. One such quantifier is part of the vocabulary of $\mathcal{L}_B$, namely `most` (together with its negation `not_most`). Because I first wish to produce data that are as similar to Bowman's as possible, only replacing the NL calculus by FOL deductions, it is preferable to maintain all quantifiers of his original data. Hence, five options to circumvent the issue of quantifier `most` are briefly considered below. Unfortunately, none of these options provides a satisfying solution, due to the problems indicated for each of them.

1. *Generalized quantifier.* Following Barwise and Cooper 1981, the quantifier `most` can be added as a generalized quantifier to the logical syntax of FOL, yielding the system FOL(`most`). `most` can be represented as a $\langle 1,1 \rangle$ quantifier with the following definition:

   $$\texttt{most}(A,B) \Leftrightarrow \mid A \cap B \mid > \mid A - B \mid$$

   This equivalence states that most A are B if and only if the set of objects that are A and B (the intersection $A \cap B$) contains more elements than the set of objects that are A but not B (the difference $A - B$). Hereby, it captures the understanding of `most` as applying to a strict majority of some (sub)domain of discourse.

   Problem: In FOL(`most`) it is possible to define the Härtig quantifier I_M, which expresses equinumerosity: $I_M(A,B) \Leftrightarrow \mid A \mid = \mid B \mid$. With this quantifier one can categorically characterize $(\omega, <)$, so that it follows from Tarski's theorem that there is no axiomatization (Weaver 2012). This implies that completeness has to be given up.

2. *Weak models.* In second-order logic (SOL), using weak semantics, it is possible to express `most`. This is because SOL allows quantification over relations in addition to quantification over individual objects in the domain. See Mostowski 1995 and van Benthem and Westerståhl 1995.

Problem: This is second-order. It is strongly preferable to maintain a first-order system, because automated theorem proving for SOL is problematic and several nice properties of the logic would have to be given up. Most importantly, it is a corollary of Gödel's incompleteness theorem that no deductive system for SOL can be sound, complete and effective at the same time with standard semantics (Gödel 1931; Shapiro 1991).

3. *Syllogistic system.* In a syllogistic system, `most` can be added as a quantifier with corresponding introduction and elimination rules. The resulting system can be complete, as is shown by Endrullis and Moss 2015.

   Problem: `most` is primitive in this approach, so no immediate representation in FOL is available. It could be possible to add the quantifier, with the corresponding inference rules, but this would either create the problems of option 1, or the number of axioms would become too high for a theorem prover to function properly.

4. *Majority digraphs.* In finite simple digraphs, the property 'most As are B' can be expressed as a binary relation, as shown by Lai, Endrullis, and Moss 2016. The results are applied to a propositional logic.

   Problem: This system is too weak. The goal is to find a FOL representation, not a propositional one.

5. *Constant domain.* Assume that the domain has a fixed size, say n. Then 'most x are A' is equivalent to 'more than $n/2$ x are A'.

   Problem: Although from a computational point of view it is justified to assume a finite domain, it does not seem reasonable to assume that the domain has a fixed size n. Even if this were the case, a particular n has to be chosen. This decision would always be arbitrary. And even if this were acceptable, not only the domain would need to have a fixed size; also the different predicates should have a fixed cardinality to account for statements such as 'most A are B'.

It appears that none of these five possibilities meets our requirements. For this reason, the quantifier `most` will be left out of consideration when generating the first new data set. All other quantifiers used by Bowman can be expressed in FOL and will be included. Table 4.1 shows how all of the positive ones can be translated to FOL representations. For negated determiners, nouns and verbs it suffices to prefix the FOL encoding of their positive counterparts with a negation symbol.

It turns out that substitution of NL with FOL effectively solves the issues identified in the preceding chapter. The only drawbacks are the inexpressibility of `most` and the computational overhead of the intermediate translation to a FOL representation and the automated theorem proving that is to replace the rather simple monotonicity calculus. Another possible objection is that the tokenization required by the formalization into FOL does not correspond to the cognitive processing of natural language. If one believes that human agents deduce semantic relations by processing utterances in their natural form, without resorting to some abstract language of thought, then NL has a definite

$\mathcal{L}_B$ (NL)	FOL
`( ( all warthogs ) move )`	$\forall x(warthog(x) \rightarrow move(x))$
`( ( some warthogs ) move )`	$\exists x(warthog(x) \wedge move(x))$
`( ( two warthogs ) move )`	$\exists x \exists y(warthog(x) \wedge warthog(y) \wedge move(x) \wedge move(y) \wedge x \neq y)$
`( ( three warthogs ) move )`	$\exists x \exists y \exists z(warthog(x) \wedge warthog(y) \wedge warthog(z) \wedge move(x) \wedge move(y) \wedge move(z) \wedge x \neq y \wedge y \neq z \wedge x \neq z)$

Table 4.1: Representations of quantifiers expressible in FOL with equality. $t_i \neq t_j$ abbreviates $\neg(t_i = t_j)$.

advantage over FOL. However, the NL expressions in Bowman's data set were not entirely natural to begin with (see page 87), so some degree of formalization was already present.

Data generation The data generation process has to be adapted according to the new logic. The monotonicity calculus of NL, with its jointables and projectivity matrices, is discarded. Instead, the relation between two expressions is established by means of an automated theorem prover for FOL with equality, Prover9. If Prover9 does not manage to find a proof in time, the model builder Mace4 takes over (McCune 2010, see Section 2.2.2). Thanks to the soundness and completeness of FOL, the notions of syntactic and semantic provability coincide. This means that (in the context of the current research) no distinction has to be made between the results obtained by Prover9 and Mace4.

The first step is to translate the lexical entailment relations between vocabulary items into FOL. This information represents the taxonomic background knowledge that is always true, and could therefore be treated as a set of axioms. The axioms are derived from the primitive entailment relations according to the mapping of Table 4.2.

lexical entailment relation	axiom representation in FOL
$A < B$	$\{\forall x(A(x) \rightarrow B(x))\}$
$A > B$	$\{\forall x(B(x) \rightarrow A(x))\}$
$A \mid B$	$\{\forall x(\neg(A(x) \wedge B(x))), \neg\forall x(A(x) \vee B(x))\}$
$A^{\wedge}B$	$\{\forall x(\neg(A(x) \wedge B(x))), \forall x(A(x) \vee B(x))\}$
$A \vee B$	$\{\forall x(\neg A(x) \rightarrow B(x))\}$

Table 4.2: FOL axiom representations of lexical entailment relations. (Recall from Table 2.1 that $<$ denotes forward entailment, $>$ backward entailment, $\mid$ alternation, $^{\wedge}$ negation and $\vee$ cover.)

The logical representations in Table 4.2 are the FOL analogues of the set-theoretic definitions provided for each entailment relation in Table 2.1. Note that no translation is provided for = (equivalence) and # (independence). The only notions that are lexically equivalent are so by self-identity. This is necessarily the case for all predicates, so no additional axioms representing such tautologies are required. Independence simply means that there is no particular semantic relation between two concepts, so for these cases there is nothing meaningful to encode. For the relations | (alternation) and ^ (negation), two axioms have to be addded, as shown in Table 4.2.

The FOL axioms are derived for all nouns in $\mathcal{N}^{\mathcal{L}_B}$ and verbs in $\mathcal{V}^{\mathcal{L}_B}$ according the translation of Table 4.2 and the lexical entailment relations visualized in Figures 3.1 and 3.2. Let the total set of axioms for Bowman's language $\mathcal{L}_B$ be denoted by $A^{\mathcal{L}_B}$. Provability in FOL with equality and the universally valid formulas of $A^{\mathcal{L}_B}$ can then be expressed by $\vdash_{A^{\mathcal{L}_B}}$. That is, if $\varphi \vdash_{A^{\mathcal{L}_B}} \psi$, then ψ is provable from φ in FOL with the correct axiomatization. Given soundness, completeness and the current research focus, the distinction between semantic and syntactic provability is not relevant enough to necessitate separate notation for both concepts, so $\vdash_{A^{\mathcal{L}_B}}$ also represents the equivalent notion $\models_{A^{\mathcal{L}_B}}$.

Not only logical, but also computational aspects have to be taken into account. The speed of Prover9 and Mace4 rapidly decreases as the number of axioms grows, so it is essential to keep the set of constraints considered per derivation as limited as possible. Hence, before computing whether $\varphi \vdash_{A^{\mathcal{L}_B}} \psi$, the collection of axioms is filtered in such a way as to retain the minimal set of formulas that could possibly be used in the proof or refutation of this particular entailment. This is done by dismissing all axioms containing predicates that do not occur in either φ or ψ. E.g., if $\varphi = \forall x(A(x) \rightarrow C(x))$ and $\psi = \forall x(B(x) \rightarrow D(x))$, then all constraints in $A^{\mathcal{L}_B}$ containing terms not in $\{A, B, C, D\}$ are omitted. As the first term in a FOL representation of a $\mathcal{L}_B$ sentence (c.q. A and B) is always a noun from $\mathcal{N}^{\mathcal{L}_B}$, while the second term (c.q. C and D) is always a verb from $\mathcal{V}^{\mathcal{L}_B}$, only those axioms are used that relate the noun predicates or the verb predicates of both sentences to each other. No axioms combining terms from $\mathcal{N}^{\mathcal{L}_B}$ and $\mathcal{V}^{\mathcal{L}_B}$ exist, so this is generally the case. Additionally, not only identical but also equivalent axioms are eliminated. That is, if e.g. $\forall x(A(x) \vee B(x))$ is already included, $\forall x(B(x) \vee A(x))$ is redundant and cannot be added as well.

Apart from filtering the axioms of $A^{\mathcal{L}_B}$ to the bare minimum, it is sometimes necessary to append valid formulas as well. This is the case for propositions whose first predicate is not necessarily instantiated, e.g. '((all warthogs) walk)'. At present, this expression does not entail that there is any object instantiating the term `warthogs`: the FOL translation $\forall x(warthog(x) \rightarrow walk(x))$ does not imply that any warthog actually exists. Bowman uses MacCartney's NL, which endorses the assumption of non-vacuity, according to which there can be no quantification over terms with empty (or universal) denotation (MacCartney 2009, p. 78). Hence, I do so too. In the current example, this means that the existence of some warthog is postulated. In general, it means that for a premise or hypothesis of the form $\forall x(A(x) \rightarrow B(x))$ or $\neg\forall x(A(x) \rightarrow B(x))$, the axiom $\exists x A(x)$ is added.

Now that all logical and computational prerequisites have been addressed, the final procedure to infer the entailment relation between two sentences can be provided in pseudocode as Algorithm 4.1. The input comprises the $\mathcal{L}_B$ statements S_l and S_r, where S_l is the left and S_r the right sentence of some pair (S_l, S_r). The output $\beta(S_l, S_r)$ is the entailment relation between them. The sentences are translated to their FOL encodings φ, ψ by the function `translate_to_fol`, which is just the mapping of Table 4.1. The axioms of $A^{\mathcal{L}_B}$ are filtered by the function `filter` as described in the preceding paragraphs, thus yielding the minimal set $A^{\varphi,\psi}$ required for proofs involving φ and ψ. Although all seven entailment relations can be returned as output, at most four proofs are required, namely those determining whether there S_l entails S_r ($\varphi \vdash_{A^{\varphi,\psi}} \psi$), whether S_r entails S_l ($\psi \vdash_{A^{\varphi,\psi}} \varphi$), whether S_l and S_r contradict each other ($\vdash_{A^{\varphi,\psi}} \neg(\varphi \wedge \psi)$) and whether the disjunction of S_l and S_r is always true ($\neg\varphi \vdash_{A^{\varphi,\psi}} \psi$). The results of these proofs are, respectively, the Boolean variables $forward\_entailment, backward\_entailment, conflict$ and *exhaustion*, all initialized as 'False'. Sometimes it suffices to only (dis)prove $forward\_entailment$ and *backward_entailment*. If at least one of these is true, it follows that the entailment relation is either $=, <$ or $>$. If not, $conflict$ and *exhaustion* must be checked in order to determine whether $|, ^\wedge$ or $\vee$ holds. If none of these relations is deducible, # is returned. Proofs are first attempted by Prover9. If the maximum time or recursive depth of the pending proof is exceeded, it is aborted and Mace4 takes over by searching a countermodel. E.g., to check whether $\varphi \vdash_{A^{\varphi,\psi}} \psi$, Mace4 tries to find a model $\mathcal{M}$ such that $\mathcal{M} \models \{A^{\varphi,\psi}, \varphi\}$, while $\mathcal{M} \not\models \psi$. If such a model is found, it follows that $\varphi \nvdash_{A^{\varphi,\psi}} \psi$. Upon completion, the sentence pair (S_l, S_r) is added to the data, labelled with the inferred entailment relation $\beta(S_l, S_r)$.

New data sets With the new deductive method in place, it is possible to start processing Bowman's data by substituting the NL labels with the FOL relations. The pairs with sentences containing `most` are removed. No other data instances are omitted or manipulated. I will not translate all of Bowman's five folds, but concentrate exclusively on `f1`, following Repplinger 2017, p. 61: 'no discernible differences could be found between single model runs on these sets, and no motivation of the splits is offered or could be derived by us.' Furthermore, translating all data in `f1` to `f5` and training all models for at least five runs on each of these folds would be excessively time-consuming.

The translated data set is called `f1_fol`. The training data contain some 22,000 sentences, which is approximately 6,000 fewer than the original `f1` training set, due to the omission of `most`. The test set is reduced to 4,500 pairs, as opposed to 6,500 in its NL predecessor. A preliminary, manual inspection of the data does not reveal any of the anomalies encountered in Bowman's. In particular, all of the five sample errors given on page 90 are labelled correctly by the new method.[1] A more general analysis of the conflicts between Bowman's `f1` and our `f1_fol` train set shows that 45.3% of the instances in `f1_fol` are labelled differently in `f1`. This is almost half of the entire training data, which is an extremely large fragment. The confusion matrix in Figure 4.1a provides

[1] To be precise, the computed FOL relations are $|, \#, >, >, \#$, respectively, which is in agreement with the intuitions articulated on page 90.

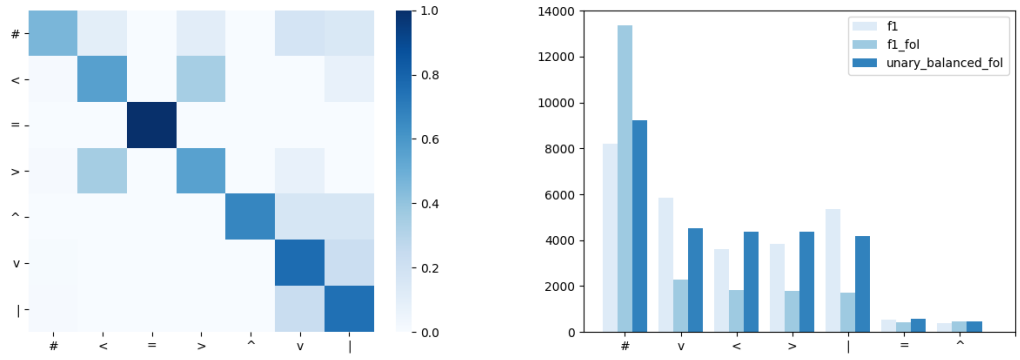
Input: left sentence S_l, right sentence S_r
Output: entailment relation $\beta(S_l, S_r)$

$forward\_entailment, backward\_entailment, conflict, exhaustion \leftarrow$ `False`
$\varphi, \psi \leftarrow$ `translate_to_fol`(S_l), `translate_to_fol`(S_r)
$A^{\varphi,\psi} \leftarrow$ `filter` $(A^{\mathcal{L}_B}, \varphi, \psi)$
if $\varphi \vdash_{A^{\varphi,\psi}} \psi$ **then**
  $forward\_entailment =$ `True`
if $\psi \vdash_{A^{\varphi,\psi}} \varphi$ **then**
  $backward\_entailment =$ `True`
if $forward\_entailment \wedge backward\_entailment$ **then**
  return $=$
else if $forward\_entailment$ **then**
  return $<$
else if $backward\_entailment$ **then**
  return $>$
if $\vdash_{A^{\varphi,\psi}} \neg(\varphi \wedge \psi)$ **then**
  $conflict =$ `True`
if $\neg\varphi \vdash_{A^{\varphi,\psi}} \psi$ **then**
  $exhaustion =$ `True`
if $conflict \wedge exhaustion$ **then**
  return $\wedge$
else if $conflict$ **then**
  return $|$
else if $exhaustion$ **then**
  return $\vee$
else
  return $\#$

Algorithm 4.1: Inference of FOL entailment relation.

a visualization. As the matrix is row-normalized, it is visible that the instances labelled with a specific FOL relation have the same relation as their most frequent label in the NL training data. However, the diagonal entries account for less than 55% of all data. So far, sample-wise checks of the individual conflicts have only brought more aberrations in the NL data to light. Figure 4.1b illustrates the distribution of both training sets in absolute frequencies. This reveals something the confusion matrix does not, namely the fact that the new data are very unbalanced. Some 61% of the instances are labelled with #, as opposed to 30% in Bowman's data. This bias is caused by the numerous cases where the NL calculus erroneously concludes a more 'interesting' relation than independence. The labels $\vee$, $<$, $>$ and $|$ remain evenly distributed, but have a much lower relative frequency than before ($\sim$ 10% vs $\sim$ 20% before). $=$ and $\wedge$ have a very low frequency, but this is because equivalence (de facto identity) and negation rarely occur.[2]

[2]This imbalance will be corrected on page 38.



(a) (b)

Figure 4.1: (a) Row-normalized confusion matrix visualizing the distribution of NL labels in Bowman's `f1` training set (columns) with respect to the FOL labels in the new `f1_fol` training set (rows). (b) Histograms showing the absolute frequency of each entailment relation in the training sets of `f1`, `f1_fol` and `unary_balanced_fol`.

The sumNN, tRNN and tRNTN models are trained on the new data. The training regime is the same as for the replication described in the preceding chapter. Also the network topologies remain the same. Word embeddings have 25 dimensions, the comparison layer 75. Results are shown in Table 4.3. Comparing these findings to Table 3.2 shows a dropped performance across the board. Especially the summing baseline suffers from the new data, but also the tree-shaped models deteriorate.

	`f1_fol`		`unary_balanced_fol`	
	train	test	train	test
25/75 sumNN	91.7	87.3	84.2	78.0
25/75 tRNN	96.5	95.4	98.0	98.0
25/75 tRNTN	99.6	98.4	99.5	98.4

Table 4.3: Training and testing accuracy scores on the FOL inference task for data `f1_fol` and `unary_balanced_fol`. Results are averaged over five runs.

Due to the unbalanced nature of the new data `f1_fol` it is not clear to what extent the decrease in accuracy is attributable to the substitution of NL with FOL. To assess this, a data set is needed that corrects for the imbalance and smaller size of `f1_fol`. I call these new data `unary_balanced_fol`. A training and test set are generated according to the described procedure, but no reference is made to the original Bowman data anymore. Sentences are independently constructed from the grammar of Table 3.1, pairs are formed and their FOL relations inferred. With a downsampling ratio, the fraction of #-labelled instances is controlled. Again, a partitioning between single statements guarantees that the train and test set have neither pairs nor sentences in common. The

absolute frequencies of the labels in the training set of `unary_balanced_fol` are shown in Figure 4.1b. The distribution seems acceptable again, with 33% of all instances labelled #. This still makes independence far more prevalent than any other relation, but at least a fair comparison is possible now. The data size has also been augmented in order to match Bowman's.

Results of training all three models on the `unary_balanced_fol` data are included in Table 4.3. The performance of the tensor model is hardly affected, but the average accuracy of the matrix model improves significantly, closely approaching the replication results of Table 3.2. These findings demonstrate that also with FOL as background logic, the tree-shaped models can learn to recognize the semantic relation between two sentences quite successfully. Apparently, the tree-shaped networks are not strongly (dis)advantaged by the use of the frequently erroneous NL data.

This is different for the sumNN. Both for the `f1_fol` and the `unary_balanced_fol` data, the baseline obtains much lower scores than reported in Table 3.2. For the `unary_balanced_fol`, accuracy scores drop most. How can this effect be explained? At first sight, it seems that there is no structural difference between the new data and Bowman's `f1` that could challenge the sumNN so much more than the other models. But perhaps this intuition is mistaken, and the use of FOL does constitute a serious complication that the tree-shaped models can cope with, thanks to their sophisticated architecture, while the rudimentary summing baseline has no such compensation. This would confirm once more that the FOL task is more interesting to consider than the NL one, because a problem that hardly challenges the baseline disincentivizes the examination of more complex models.

4.1.2 New language

By endorsing FOL instead of NL (or Bowman's version thereof), the first two problems identified in Section 3.3 have been solved. This comes at the cost of the computational overhead of FOL theorem proving and model building, as well as the omission of the quantifier `most`. This price is not prohibitive, because the data only have to be generated once and it is disputable whether a representation of the problematic determiner `most` constitutes a critical addition to a model that already captures the classical logical quantifiers. At the same time, many other issues remain, and cannot be addressed by only changing the underlying logic. More specifically, in order to complicate the syntax (problem 3), remove the ontological symmetry (problem 4), and decompose the quantifiers (problem 5), it is necessary to replace $\mathcal{L}_B$, design a new artificial language and use it as the basis for new data. This is the aim of the current section.

Desiderata Let $\mathcal{L}_N$ (with N for 'new') denote the new language, which comprises a phrase structure grammar, a vocabulary and taxonomy of terms that are at present still unspecified. Recalling problems 3-5 of Section 3.3 (elaborated on pages 92-94), $\mathcal{L}_N$ should ideally satisfy the following requirements:

1. $\mathcal{L}_N$ should be able to produce longer expressions than $\mathcal{L}_B$, which has a maximal sentence length of only five words, while many sentences are even shorter.
2. Different sentence constituents should not only be taken from mutually exclusive classes, so that the same words can be used more than once in a single statement. This also implies that word order becomes more important.
3. Not only unary, but also binary predicates must be included.
4. No taxonomic symmetries (such as the one illustrated in Table B.2) are allowed.
5. Negated quantifiers are not composed, but included as pairs of two separate lexical items: negation/adverb `not` together with a positive quantifier.

Additionally, it would be nice if the ontology of $\mathcal{L}_N$ were inspired by something else than biology, not just for the sake of variation but also to show that there exist other domains than the animal kingdom where such semantic hierarchies are applicable.[3] A positive side effect would be that we can then discharge the rather unrealistic assumption that there are people with pet warthogs.

First, the noun and verb classes of the new language are introduced: $\mathcal{N}^{\mathcal{L}_N}$ and $\mathcal{V}^{\mathcal{L}_N}$. For both these sets, care is taken to prevent the kind of symmetry observed in $\mathcal{N}^{\mathcal{L}_B}$, so that the fourth requirement is met. $\mathcal{N}^{\mathcal{L}_N}$ contains five nouns, all of which are unary predicates, as before. Instead of biological classes, the noun hierarchy now comprises four demonyms (identifying people from different geographical locations), and the independent class `children` (comparable to the class `pets` in $\mathcal{L}_B$). The new taxonomy of nouns is visualized in Figure 4.2. The four verbs in $\mathcal{V}^{\mathcal{L}_N}$ are all transitive, e.g. fearing and loving. That is, they take both a subject and an object argument, which makes them binary predicates. Thereby, the third requirement is satisfied. Both verb arguments are taken from $\mathcal{N}^{\mathcal{L}_N}$, so that the subject and the object of a sentence are not taken from mutually exclusive classes. This addresses the second requirement. $\mathcal{V}^{\mathcal{L}_N}$ is illustrated in Figure 4.3.[4]

The adverbs $\mathcal{A}^{\mathcal{L}_N}$ are not changed, and contain only negation and an empty string as dummy adverb. In order to meet the fifth requirement, all negated, composed quantifiers from $\mathcal{Q}^{\mathcal{L}_B}$ are discarded. Moreover, the numerical quantifiers `two` and `three` are no longer considered. There are several reasons for this decision. As the current project is FOL-oriented, it is natural to start by studying models' capacities to capture the notions that

[3] At first, the taxonomies were intended to instantiate all seven relations from Table 2.3. This turned out virtually impossible, unless very unnatural predicates were adopted. The cover relation proved particularly difficult, as already noted by MacCartney 2009, p. 90: 'Very few naturally-occurring pairs of terms belong to the $\vee$ relation. When we are obliged to produce such pairs for the purpose of an illustrative example, we often resort to using a general term and the negation of one of its hyponyms, such as *mammal* $\vee$ *nonhuman* or *metallic* $\vee$ *nonferrous*.' Hence, in order to maintain a somewhat natural vocabulary, this requirement was dropped.

[4] Note that the extension of a binary predicate P is the set of ordered pairs $\langle a, b\rangle$ for which $P(a, b)$ holds. Hence, in the Venn diagram of Figure 4.3, the circles do not represent sets of individual objects, but of ordered pairs. E.g., if a pair $\langle a, b\rangle$ is contained by `love`, this means that a loves b, implying that a likes b and $\langle a, b\rangle$ is also a member of `like`.

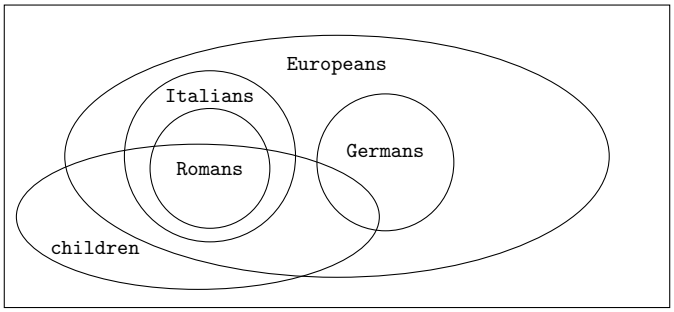


Figure 4.2: Venn diagram visualizing the taxonomy of nouns $\mathcal{N}^{\mathcal{L}_N}$ in $\mathcal{L}_N$

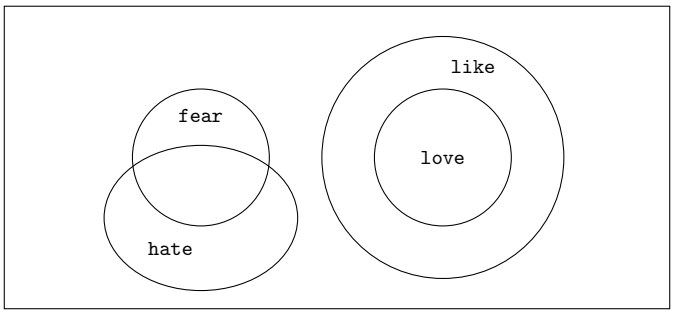


Figure 4.3: Venn diagram visualizing the taxonomy of verbs $\mathcal{V}^{\mathcal{L}_N}$ in $\mathcal{L}_N$.

are logically primitive in this framework. The only irreducible quantifiers of FOL are $\forall$ (c.q. `all`) and $\exists$ (c.q. `some`), and with equality, all numerical quantifiers are definable in terms of the existential one (see Table 4.1). Hence, from a logical point of view, the numerical quantifiers are far less interesting than `all` and `some`. Also, their inclusion causes computational issues. Table 4.1 shows how number-based statements increase the length of FOL encodings, thereby greatly slowing down the theorem proving process on the available computational resources. Although it would be worthwhile to study the numerical quantifiers at a later stage, they are currently omitted on these grounds. An overview of the complete vocabulary of $\mathcal{L}_N$ can now be provided:

$$\mathcal{L}_N \begin{cases} \mathcal{Q}^{\mathcal{L}_N} = \{\texttt{all}, \texttt{some}\} \\ \mathcal{N}^{\mathcal{L}_N} = \{\texttt{Romans}, \texttt{Italians}, \texttt{Germans}, \texttt{Europeans}, \texttt{children}\} \\ \mathcal{V}^{\mathcal{L}_N} = \{\texttt{fear}, \texttt{hate}, \texttt{like}, \texttt{love}\} \\ \mathcal{A}^{\mathcal{L}_N} = \{\texttt{not}, \epsilon\} \end{cases}$$

Of all five desiderata on page 39, only the first one still has to be addressed. The maximum sentence length in $\mathcal{L}_N$ depends on the new phrase structure grammar provided in Table 4.4. Sentences are now of the form '(((adverb quantifier) (adverb noun))

((adverb verb) (quantifier (adverb noun))))'.[5] As adverbs can be empty strings, this means that the minimal sentence length is five words, and the maximum nine, so statements can become almost twice as long as in $\mathcal{L}_B$. It is clear now that $\mathcal{L}_N$ satisfies all specified requirements for the new language.

S $\rightarrow$ NP VP	Det $\rightarrow$ Quant
NP $\rightarrow$ Det NP	Quant $\rightarrow \mathcal{Q}^{\mathcal{L}_\mathcal{N}}$
NP $\rightarrow$ Adv N	N $\rightarrow \mathcal{N}^{\mathcal{L}_\mathcal{N}}$
VP $\rightarrow$ VP NP	V $\rightarrow \mathcal{V}^{\mathcal{L}_\mathcal{N}}$
VP $\rightarrow$ Adv V	Adv $\rightarrow \mathcal{A}^{\mathcal{L}_\mathcal{N}}$
Det $\rightarrow$ Adv Quant	

Table 4.4: Phrase structure grammar for artificial language $\mathcal{L}_N$.

Data generation The data generation process is essentially the same as described in Section 4.1.1, but has to be extended to accommodate the new grammar. Specifically, the translation between representations of $\mathcal{L}_N$ phrases and their FOL encodings must be updated. As the syntactic backbone of FOL representations is determined by the quantifiers in the concerned expressions, it is the quantifier configuration that governs the translation. Because $\mathcal{L}_N$ has two unique quantifiers, there are only four such configurations, all of which are included in Table 4.5. The four examples do not contain negations, but these can simply be inserted as prefixes to the predicates or quantifiers they are meant to modify. FOL axioms are derived from the lexical entailment relations of the noun and verb ontologies as before, but the translation of Table 4.2 must be generalized to binary predicates. This is easily realized by adding a universal quantifier and an object argument to the FOL representations.

$\mathcal{L}_N$	FOL
`( ( all Europeans ) ( love ( all Germans ) ) )`	$\forall x(European(x) \rightarrow \forall y(German(y) \rightarrow love(x,y)))$
`( ( all Europeans ) ( love ( some Germans ) ) )`	$\forall x(European(x) \rightarrow \exists y(German(y) \wedge love(x,y)))$
`( ( some Europeans ) ( love ( all Germans ) ) )`	$\exists x(European(x) \wedge \forall y(German(y) \rightarrow love(x,y)))$
`( ( some Europeans ) ( love ( some Germans ) ) )`	$\exists x(European(x) \wedge \exists y(German(y) \wedge love(x,y)))$

Table 4.5: FOL representations of $\mathcal{L}_N$ statements.

With the updated translation and axiomatization functions, Algorithm 4.1 can be used to infer the FOL relation per pair of $\mathcal{L}_N$ sentences. Thus, a new data set is generated, which I call `binary_fol`. By downsampling the instances labelled with #, a balanced distribution is maintained. The train set contains some 30,000 instances, the test set

[5] Theoretically, it is also possible to negate the second quantifier in a sentence, but to keep the statements somewhat readable I decided not to do so in the current data.

7,500. As always, no unique sentences from the train set occur in the test set and vice versa. A small specimen is shown below:

```
<  ( ( all Europeans ) ( like ( some Italians ) ) )
                        ( ( ( not some ) Italians ) ( ( not like ) ( some Europeans ) ) )
v  ( ( all Germans ) ( ( not hate ) ( all ( not Italians ) ) ) )
                        ( ( ( not all ) ( not Italians ) ) ( love ( some ( not Italians ) ) ) )
#  ( ( all children ) ( ( not hate ) ( all Romans ) ) )
                        ( ( all ( not Italians ) ) ( ( not fear ) ( all Romans ) ) )
|  ( ( some ( not Europeans ) ) ( like ( all ( not Italians ) ) ) )
                        ( ( ( not some ) ( not Italians ) ) ( like ( all ( not Italians ) ) ) )
^  ( ( ( not all ) ( not Germans ) ) ( ( not fear ) ( all Europeans ) ) )
                        ( ( ( not some ) ( not Germans ) ) ( fear ( all Europeans ) ) )
```

The several negations can make these sentences cumbersome to read, but careful inspection shows that the relations assigned to the examples are as one would expect. Generally, it is helpful to read the quantifier `some` as 'any' in negated clauses. The last instance of the above specimen nicely demonstrates that complex instantiations of De Morgan's equivalences are handled correctly.

The data set `binary_fol` is based on a random sample from the total space of possible sentence pairs. Let $\mathcal{S}_{\mathcal{L}_N}$ denote the set of all sentences in $\mathcal{L}_N$. This set equals the Cartesian product $\mathcal{A}^{\mathcal{L}_N} \times \mathcal{Q}^{\mathcal{L}_N} \times \mathcal{A}^{\mathcal{L}_N} \times \mathcal{N}^{\mathcal{L}_N} \times \mathcal{A}^{\mathcal{L}_N} \times \mathcal{V}^{\mathcal{L}_N} \times \mathcal{Q}^{\mathcal{L}_N} \times \mathcal{A}^{\mathcal{L}_N} \times \mathcal{N}^{\mathcal{L}_N}$. The cardinality of this set is $2 \times 2 \times 2 \times 5 \times 2 \times 4 \times 2 \times 2 \times 5 = 6,400$. It follows that the number of different sentence pairs is $6,400 \times 6,400 = 40,960,000 \approx 40$M. The training set of `binary_fol` contains almost 30,000 pairs, which is only 0.07% of the total space of possibilities. This fraction is 60 times smaller than Bowman's 4.2%, so it seems that also problem 6 of Section 3.3, concerning the relative size of the training data, has been solved. It is worth noting that $\mathcal{L}_N$ has a nearly 800% larger space of admissible sentences than $\mathcal{L}_B$, whereas the vocabulary has become almost 40% smaller. Otherwise stated, $\mathcal{L}_N$ produces many more sentences with fewer words. In this respect, it reflects the (infinitely) generative capacity of human language more realistically than $\mathcal{L}_B$.

4.2 Recursive models

As shown, the `binary_fol` data solve problems 1-6 of Section 3.3. I now proceed to train the summing baseline and the recursive models on the new training set according to the same regime as before. Following a description of the results thus obtained, a more in-depth interpretation will be provided.

4.2.1 Results

Training and testing accuracies after 50 training epochs, averaged over five different model runs, are shown in Table 4.6.

Immediately striking about these results is the increased variation between the scores obtained by the different models. The tensor model still performs very well, with nearly perfect average training accuracy. Testing accuracy is 5 percentage points lower, but

	train	test
25/75 sumNN	55.9	51.3
25/75 tRNN	83.0	81.7
25/75 tRNTN	99.0	94.1

Table 4.6: Training and testing accuracy scores on the FOL inference task for the `binary_fol` data. Results are averaged over five runs.

still well above 90%. This is approximately 4% lower than the tensor testing scores of Table 4.3. Taking into account that the artificial language $\mathcal{L}_N$ is more complicated than $\mathcal{L}_B$, and that the relative size of the training data is 60 times smaller than before, it is remarkable that the tensor model only deteriorates this little. The matrix model suffers much more from the new data. Average testing accuracy is below 82%, while the tRNTN used to only slightly outperform the tRNN. It seems that for the new data, the tensor term in the composition and comparison functions of Equations (3.3) and (3.5) has gained importance. Apparently, only modelling the additive interactions between vectors suffices to obtain decent scores on the `unary_balanced_fol` data, whereas the complex `binary_fol` instances require something more. This is offered by the tensor term in the layer functions, which also pays attention to the multiplicative interactions between vectors. The summing baseline fails dramatically. Both training and testing accuracy are not far above 50%. This result clearly shows that a simple bag-of-words model is not at all capable of recognizing entailment relations for complex sentences whose constituents can always be arranged in several different orders. Previously, determining word order was only challenging in the case of a single negation. Now, it is always an issue. Nouns and quantifiers are both interchangeable, and there are four negatable positions per sentence. Models oblivious to word order, such as the sumNN, therefore face great uncertainty, which explains the current performance. The low baseline scores reported in Table 4.6 show that by adopting the new data, also problem 7 of Section 3.3 is addressed, which is extra evidence for the non-triviality of the task.

In general, the results suggest that sophisticated model properties become more important in the face of the new, complicated data. Following the replication of Bowman's experiments, all scores were still above 90% (see Table 3.2). Now, the testing accuracies of the sumNN and tRNTN are more than 40% apart. There can be no suspicion anymore that the recursive architectures of the matrix and tensor models simply serve as means to boost performance by a few percentage points. It is this design that enables the tRNTN in particular to successfully approach an entailment classification task with significantly less data and higher complexity than before. Conversely, the primitive sumNN set-up is not just a relatively minor handicap anymore, as suggested by Table 3.2, but the cause of drastic failure. Likewise, the tensor term in the layer functions no longer accounts for a marginal difference between testing accuracy scores of the matrix and tensor models, but for a gap of more than 12%.

4.2.2 Interpretation

In the remainder of this section, I focus on the most powerful model so far: the tRNTN. More specifically, of the five tensor models that were trained to obtain the results of Table 4.6, the one that performed best on the test set is selected for a more detailed analysis and interpretation. This model achieved final training and testing accuracy scores of 99.1% and 94.6%, respectively. Its row-normalized confusion matrix with respect to the `binary_fol` test data is shown in Figure 4.4a. For reference, Figure 4.4c contains a histogram visualizing the relative frequency of each entailment relation in the `binary_fol` train data.

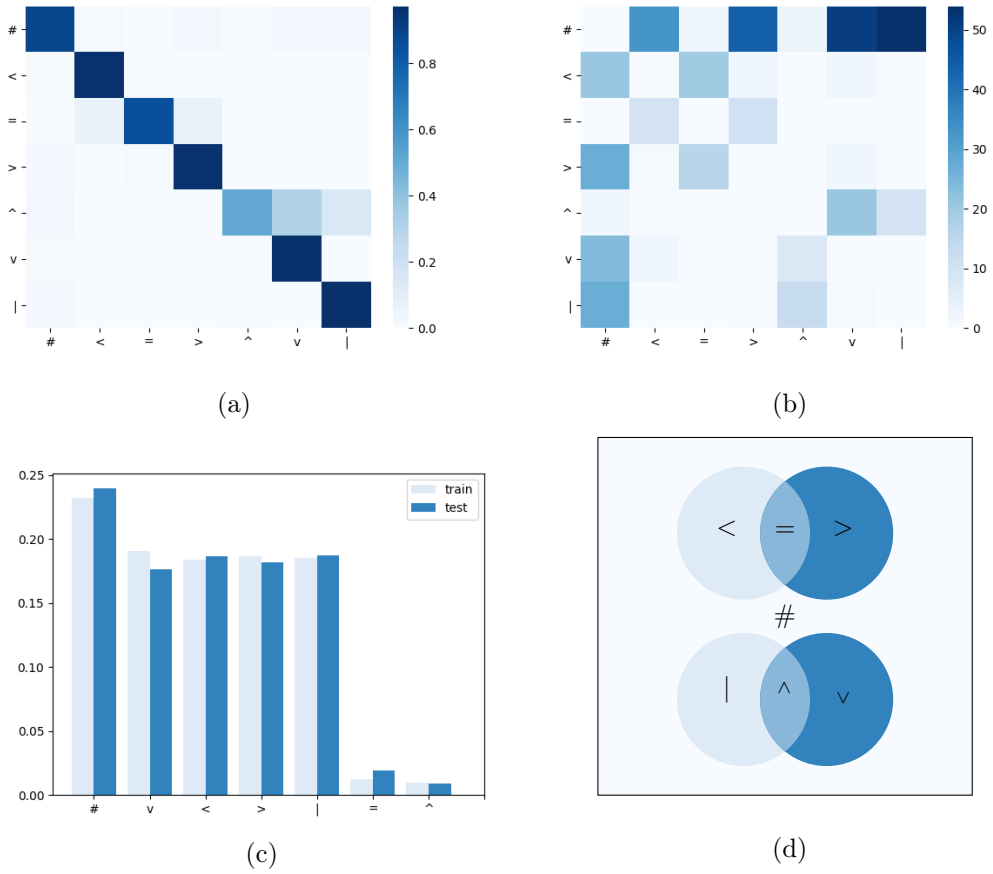


Figure 4.4: (a) Row-normalized confusion matrix with respect to the `binary_fol` test data for the best-performing tRNTN model. (b) The confusion matrix of (a), with no normalization or diagonal entries (correct classifications). (c) Histogram showing the relative frequency of each entailment relation in the train and test set of `binary_fol`. (d) Venn diagram visualizing the seven entailment relations, interpreted as the sets of sentence pairs for which they hold.

Confusion matrix The confusion matrix reveals some interesting facts about the error profile of the tensor model. Of course, because most testing instances are classified correctly, the matrix is strongly diagonal. Concentrating on the mistakes, two classes stand out: equivalence (=) and negation (^). As targets, these relations are most often misclassified. The distribution of the training data illustrated by Figure 4.4c shows that these are also the two least occurring classes in the training data. As stated on page 37, this imbalance is a logical matter that cannot be fixed as easily as the dominance of the independence relation, which was dealt with by simply downsampling #-labelled instances. Equivalence and negation are so rare that forcing a uniform distribution over all seven classes by similar means would yield a very small data set. Hence, they are included as smaller classes, as was the case in all earlier data sets (see Figure 4.1b). Figure 4.4c shows that the `binary_fol` test set has an almost identical distribution as the train set, so the errors made for targets = and ^ cannot account for most mistakes. To be precise, only 11% of all errors made by the model concern instances labelled = or ^. The confusion matrix of Figure 4.4a can be misleading in this respect, because its row-normalization seems to suggest that these are important error categories.

Indeed, the diagonal entries of the confusion matrix in Figure 4.4a are so large that they overshadow the nuances in the error profile. To avoid this, Figure 4.4b shows the same matrix, but this time without normalization and without diagonal entries. Thus, the error distribution becomes more distinct. Relative to the other error categories, the misclassifications of = and ^ are not frequent at all. Most errors are related to independence, either by misclassification (true negative, errors in the first row) or wrongful attribution (false positive, errors in the first column). The only two categories the model never confuses with independence are equivalence and negation, because these were seldom observed during training. It turns out that, even though independence is the most common class, it is not the easiest one to classify. A possible explanation is that, due to the semantic unrelatedness of #-labelled sentence pairs, independence has become a kind of container class, attributed to items whose true target the model fails to recognize. As opposed to the other relations, independence has no rigid logical definition, but is simply a class of 'other' instances. Hence, there are no relations that independence contradicts as clearly as is the case between e.g. equivalence and alternation. This is why # is more likely to show up as a mistake than the other relations.

The matrix of Figure 4.4b is quite symmetric. If a class A is often wrongfully assigned to instances labelled B, then probably B is also assigned to many A-labelled items. The details of the error pattern can all be understood with reference to the class definitions of Table 2.1, and the Venn diagram of Figure 4.4d, which visualizes the relations between the sets of sentence pairs assigned to each entailment class. Equivalence is often mistaken for either forward or backward entailment. This makes sense from a logical perspective, because the definition of equivalence is the conjunction of the definitions of forward and backward entailment. This is illustrated in Figure 4.4d, where = is the intersection of $<$ and $>$.[6] This also holds for negation, which is defined as the conjunc-

[6] In a personal meeting, Luciano Serafini (from FBK - Trento) suggested that attribution of either kind of entailment in the case of equivalence should not be treated as a full error, because it is at least

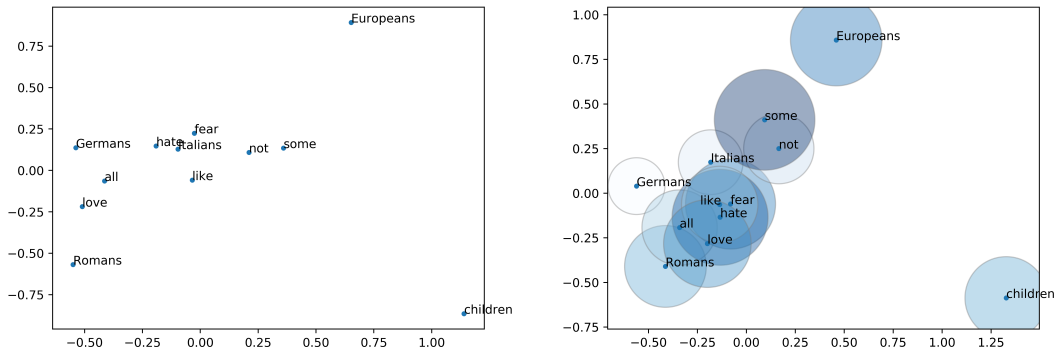
tion of cover and alternation. Indeed, negation is most often misclassified as either of these two classes. Conversely, forward and backward entailment are often misclassified as equivalence, and cover and alternation as negation, because definition-wise these are the nearest alternatives. $<$, $=$ and $>$ form the upper semantic cluster in Figure 4.4d, while the corresponding rows and columns in Figure 4.4b show that they are often confused. The same holds for $|$,$^\wedge$ and $\vee$. At the same time, sentence pairs belonging to the $<$,$=$,$>$-cluster are rarely mistaken for instances of $|$,$^\wedge$ or $\vee$, and vice versa. Thus, even in the mistakes made by this tRNTN there is logic, because the error profile reflects the definitions of the entailment classes.

Word embeddings Next, I look at the word embeddings learned by the winning tensor model. Principal Component Analysis (PCA) is applied to reduce the dimensionality of the vector representations from 25 to 2, while maintaining the largest possible variance.[7] The result is shown in Figure 4.5a. By now, it is well-known that unsupervised models can be trained on a corpus to produce a word vector space with meaningful linear substructures, as shown by e.g. Mikolov, Yih, and Zweig 2013 and Pennington, Socher, and Manning 2014. The word embeddings of our models are not taken from an existing data set, but learned during the training phase as regular network parameters (see Equation 3.1). A relevant question to ask is whether the geometry of the learned embeddings reflects the semantic interrelations between the represented words. It is difficult to assess this on the basis of Figure 4.5a alone. The verbs 'fear', 'hate' and 'like' form somewhat of a cluster. Nouns 'Germans' and 'Italians' are closer to each other than to 'Romans' or 'Europeans', which is understandable because both concern countries instead of a city or continent. The embedding for 'children' is an outlier, as this predicate is independent of all other ones.

Figure 4.5a confirms some semantic intuitions, but only weakly. As the word vector space is so large, a single PCA projection is hardly representative of general model dynamics. Therefore, Figure 4.5b shows the average of the two-dimensional PCA projections of the word embeddings learned by all five tRNTN models trained on the `binary_fol` data. Additionally, the radius of the concentric circle around each mean projected embedding indicates the average absolute distance to the individual model embeddings. It turns out that the configuration of learned word representations is surprisingly consistent across models. Each model's word embeddings are uniformly randomly initialized, so prior to

partially correct. This is justifiable, because equivalence between two sentences always implies that both backward and forward entailment are also the case, so strictly speaking there is no untruth in this kind of misclassification. But despite the semantic legitimacy, there are strong pragmatic objections against this kind of 'truth'. It is at best incomplete, because a stronger logical relation also holds. As such, it violates the scalar or quantity implicature, according to which a stronger claim is false unless explicitly claimed to be true. See Carston 1997, or the Maxim of Quantity in Grice 1975. Furthermore, forward and backward entailment are understood as exclusive notions during the data generation process, see Algorithm 4.1.

[7] Another popular dimensionality reduction algorithm is t-distributed Stochastic Neighbor Embedding (t-SNE), see Maaten and Hinton 2008. It is not used here because it is focused too much on local structure for our purposes. Moreover, due to its stochastic nature, some trial runs yielded results that were too divergent to interpret.



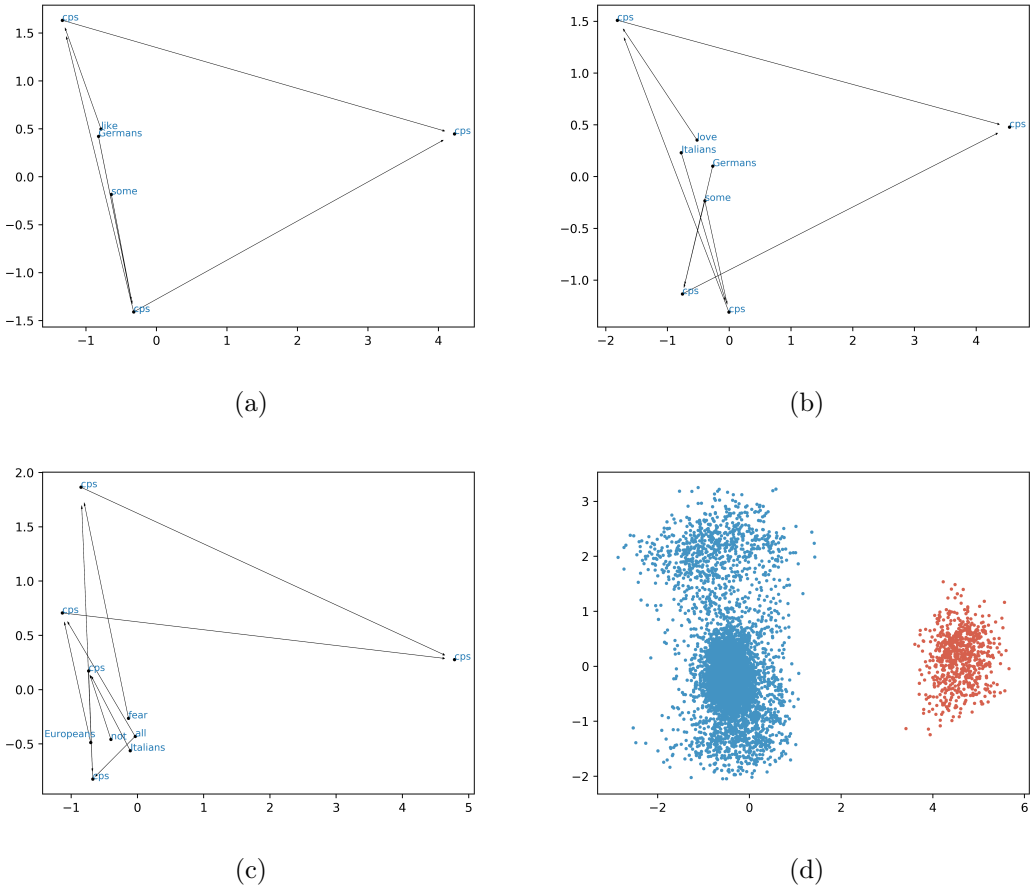
(a) (b)

Figure 4.5: (a) PCA projection of the word embeddings learned by the best-performing tRNTN model after training on `binary_fol`. (b) Average of the PCA projections of the word embeddings learned by all five tRNTN models trained on `binary_fol`, with the average absolute distance visualized as a radius around the mean locations.

training, the individual embedding matrices must be very different. Figure 4.5b shows that after 50 training epochs, the embedding matrices of the tRNTN models have converged, as the networks have learned to organize the vectors in very similar ways. Verbs clearly cluster together, 'Germans' and 'Italians' are closer to each other than to 'Romans' or 'Europeans', and the embedding for 'children' is an outlier. These observations already applied to Figure 4.5a, but their generalization to other models compensates for some of the uncertainty caused by the 25-to-2-dimensional projection. It is now more salient that the tensor models organize their word embeddings in a geometry that echoes important structural aspects of the vocabulary semantics.[8]

Composition vectors The tree-shaped models produce sentence vectors by recursively applying a learned composition function. The first arguments to this function are the embeddings of the words at the leaf nodes of the syntactic tree, the results of which then serve as new inputs until all branches have been collapsed. In order to gain more insight into this process, Figures 4.6a-4.6c show the 2-dimensional PCA projections of all stages in the best-performing tRNTN's composition of three sample expressions. These sentences are taken from the `binary_fol` test set, and belong to pairs correctly classified by the tRNTN. Projections of word embeddings are labelled with the corresponding vocabulary items from $\mathcal{L}_N$, and (activated) outputs returned by the composition function are annotated as 'cps'. The arrows indicate the direction in which the composition

[8]The project-sum-squash method, mentioned on page 28, was applied to the word embeddings and the composition matrix to check if an effect as described in Veldhoen 2015 takes place here as well. However, the method did not produce a more interpretable configuration than the projections of Figure 4.5a.



(a) (b)

(c) (d)

Figure 4.6: (a)-(c) PCA projections of all stages in the best-performing tRNTN's composition of the sentences (a) ((some Germans) (like (some Germans))), (b) ((some Germans) (love (some Italians))) and (c) ((all Europeans) (fear (all (not Italians)))). Projected embeddings are labelled with the corresponding vocabulary item from $\mathcal{L}_N$. Outputs from the composition function are annotated as 'cps'. (d) Projections of all composition stages for 1,000 sampled sentences. Blue points represent embeddings or intermediate compositions, red points denote complete sentence vectors.

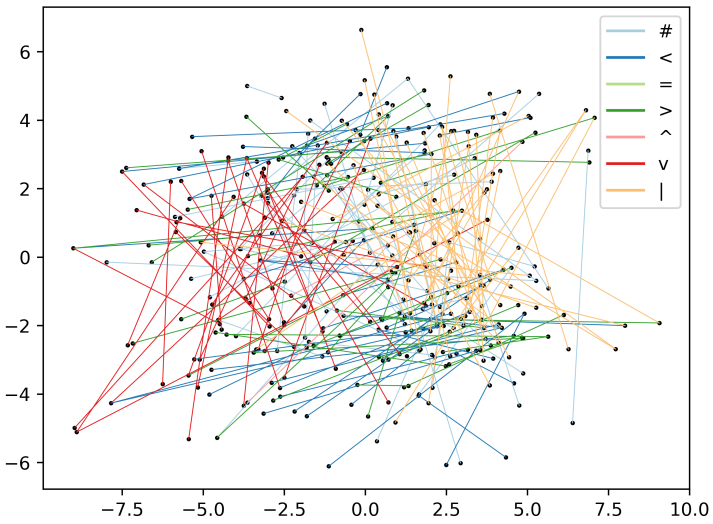
function is applied. E.g., in Figure 4.6a for the sentence ' ((some Germans) (like (some Germans)))', the composition of 'some' and 'Germans' is visualized by the arrows connecting the embeddings to a cps node representing the noun phrase '(some Germans). 'like' is composed with this cps node, which produces a new cps node representing the verb phrase '(like (some Germans))'. Finally, the composition function sends this cps node together with the '(some Germans)' node to a new cps node, which constitutes the sentence vector.

Although it is not straightforward to interpret the details of the plots in Figure 4.6,

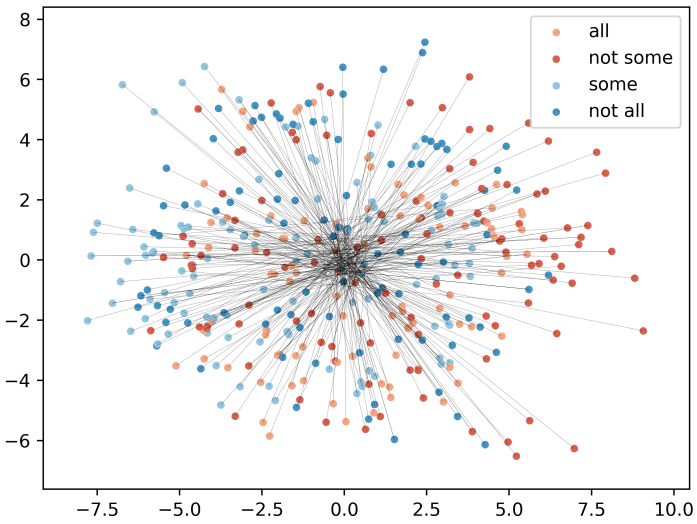
they expose some global facts about the composition method. Most importantly, they show that the distance between composition arguments and outputs increases proportionally to the complexity of the composed phrase. Embeddings represent the sentence constituents with the least complexity, namely primitive vocabulary items. When conjoining two word representations, the composition function never returns a vector in the region between the embeddings: it maps the input arguments to a different linear subspace. As the arguments become cps nodes themselves, the composition function increases the variance even more. This effect is clear in all plots 4.6a-4.6c. It is strongest at the final step, when the main noun and verb phrases of the sentence are combined. This produces the total sentence vector, which is situated in a region distant from all its recursive predecessors. Sentence vectors generally live in an isolated subspace, as is illustrated by Figure 4.6d. Here, the projections of all composition stages are plotted for 1,000 statements, sampled from the sentences in the `binary_fol` test set belonging to correctly classified pairs. Blue points represent embeddings or intermediate compositions, red points denote complete sentence vectors. The two regions are linearly separable. This strong geometric division suggests that the tRNTN recognizes sentences and their partial constituents as fundamentally different objects.

Sentence vectors Finally, I pay some more specific attention to the highest-level internal representations with a direct semantic interpretation: the sentence vectors. Here, focus is on the unactivated sentence vectors, which sum the final matrix and tensor composition results. Figure 4.7a shows the PCA projection of some 200 correctly classified sentence pairs sampled from the `binary_fol` test set. Points representing sentences that belong to the same pair are connected with a line whose color indicates their entailment relation. Figure 4.7a is not very elucidating, because it fails to show any obvious geometric characteristics distinguishing different entailment relations from each other. That is, the relations do not appear to correspond with fixed linear transformations in the sentence vector space. The plot provides only some meager evidence that relations belonging to the upper cluster of Figure 4.4d ($<, >, =$) have a preference for transformations orthogonal to those of the lower cluster ($|$,$^\wedge$, $\vee$).

Figure 4.7b singles out one relation, namely negation ($^\wedge$). Because only few instances in the `binary_fol` data are labelled with $^\wedge$, a new data set containing only negations was constructed by taking single sentences from the existing data and negating them. This new data set is called `only_negations`. Figure 4.7b shows a sample of 200 correctly classified sentence pairs from this data set, with lines connecting statements belonging to the same pairs. Only pairs of the form $\langle \varphi, \neg\varphi \rangle$ are considered, i.e. sentences together with their trivial logical complement, which prefixes the positive expression with a negation. In our case, negation of the entire sentence amounts to negation of the first quantifier. For this reason, Figure 4.7b also colors each included point according to its main determiner, understood as the (possibly negated) first quantifier. There seems to be a weak clustering of universal statements (starting with 'all' or 'not some') and existential ones ('some' or 'not all'). More importantly, the plot shows that negation dictates a very specific geometry, where sentence vectors are mirrored with respect to a kind of centroid.



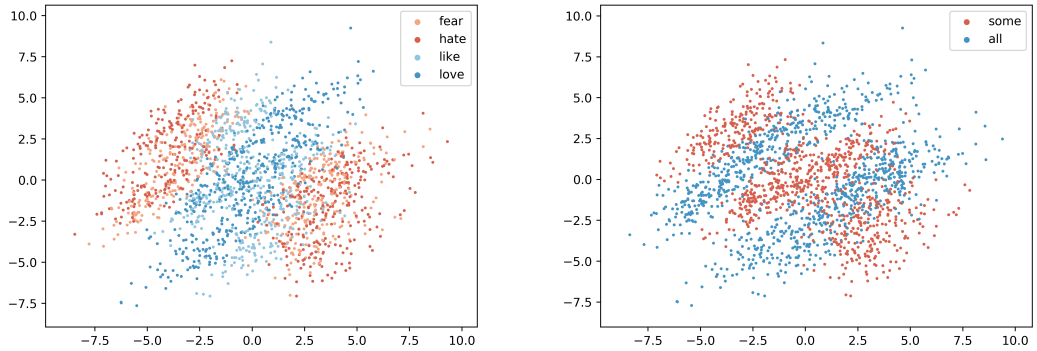
(a)



(b)

Figure 4.7: PCA projections of unactivated sentence vectors produced by the best-performing tRNTN. (a) 200 pairs of sentences connected according to their entailment relation. (b) 200 pairs of negated sentences, colored according to their first (possibly negated) quantifier.

This geometric mirroring appears to be a non-local equivalent of the semantic mirroring effectuated by logical negation: the inversion of a sentence's position relative to a specific point can serve as a metaphor for the inversion of its truth value.



(a) (b)

Figure 4.8: PCA projections of unactivated sentence vectors produced by the best-performing tRNTN. (a) 4,000 sentences, colored according to their verb. (b) 4,000 sentences, colored according to their second quantifier.

Not only the relations between sentences, but also their internal structure can reveal geometrical patterns. Figures 4.8a and 4.8b include the projections of some 4,000 sentences belonging to correctly classified pairs in the `binary_fol` test set, colored according to their verbs and second quantifiers, respectively. (Similar experiments were performed for other individual words and complex sentence components, but interesting results were only obtained for the verb and the second quantifier.) In Figure 4.8a, the colors of the projected sentences represent their verbs. Each sentence has only one verb, taken from the set $\mathcal{V}^{\mathcal{L}_N}$. In this set, 'fear' and 'hate' semantically contrast 'like' and 'love'. See also the Venn diagram of Figure 4.3. The configuration of the projections in Figure 4.8a reflects this dichotomy. Sentences containing 'like' or 'love' are clustered together in the central cloud of blue points. On either sides of this area are red clusters of sentences containing 'fear' or 'hate'. It is not clear why there are two such clusters, or what distinguishes them from each other. The overlap between the blue and red regions is not negligible, but seems largely due to sentences containing 'like'. Those containing 'love' are packed together more closely inside the blue region. Perhaps, this is explained by the fact that 'love' is a hyponym of 'like', and the corresponding extensions relate as sub- and superset.

Figure 4.8b provides a similar visualization, but now with coloring relative to the second quantifier. As this quantifier is never negated in the `binary_fol` data, there are only two values to consider: 'some' and 'all'. The plot shows five clusters, three of which contain sentences with second quantifier 'some', while the other two contain sentences with quantifier 'all'. As with the previous plot, there is no obvious explanation for the

number of clusters, but it is observable that sentences tend to group together if they share their second quantifier. So far, this has not been established for any other sentence components than the verb and the second quantifier. Apparently, these constituents prove more useful to the tRNTN as a basis for geometrical partitioning than any of the other ones, including subject and object noun phrases. It is likely that there are other factors capable of determining an equally meaningful division of the sentence vector space, but until now these are the only ones that have been identified.

Chapter 5

A recurrent approach

By now, most issues listed in Section 3.3 have been addressed. Endorsing FOL instead of NL solves problems 1 and 2, while replacing language $\mathcal{L}_B$ with $\mathcal{L}_N$ solves problems 3 to 5. Issues 6 and 7 are overcome by the new `binary_fol` data set. Problem 9, the methodological objection concerning a lack of interpretation, does not apply anymore either, given the attention paid to analysis in the preceding section. The only issue remaining is the eighth: the symbolic prerequisites of the models. It is problematic that the tree-shaped architectures depend on the availability of linguistic information specifying the syntactic structure of the input sentences. Especially the tensor models have proven successful in addressing the current entailment recognition task, but as long as they owe their achievements to symbolic background knowledge that guides their entire topology, they are neither scalable nor natural. Additionally, as mentioned on page 25, the tree-shaped set-up is notoriously inefficient to train, especially when there are tensors involved.

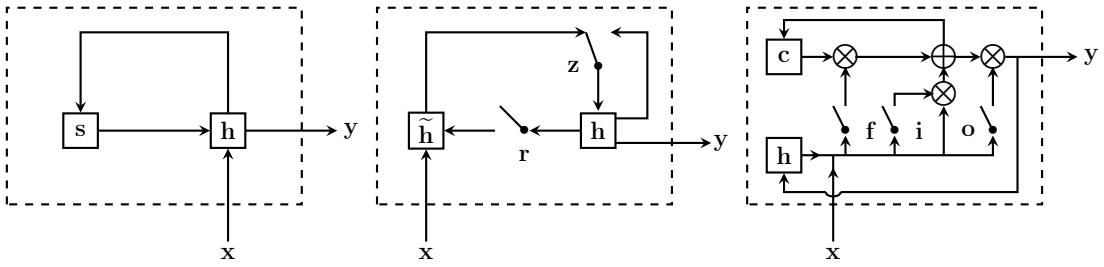
The only solution is to consider a different class of models. The aim is to learn whether they are able to address the same challenge without relying on access to parse trees. Given the poor performance of the summing baseline on the new task, a bag-of-words model is insufficient. Required is a model that processes sentences according to their internal order, without using supplementary linguistic structure of any kind to direct the computational process. This is exactly what a recurrent model does, by treating sequences of inputs in a linear fashion. Therefore, I will now focus on recurrent models, and examine whether they manage to successfully address the FOL entailment classification task in a sequential manner.

Recurrent neural networks (RNNs) are defined as neural models with feedback connections (Fausett 1994; Medsker and Jain 1999). They were introduced in the 1980s by, among others, Rumelhart, Hinton and Williams, who used them to learn strings of characters (Rumelhart, Hinton, and Williams 1985). In the 1990s they were further developed and applied to a range of real-world problems by e.g. Giles, Lawrence, and Tsoi 1997, Liang, Su, and Lin 1999 and Costa et al. 1999. Largely thanks to an increase in computational resources, it has become easier to train sophisticated RNNs in recent years. This has caused a surge in publications on recurrent architectures, which turn out

to be remarkably successful in a wide variety of tasks, ranging from machine translation (Sutskever, Vinyals, and Le 2014) to speech recognition (Graves and Jaitly 2014) and joke generation (Ren and Yang 2013).

5.1 Three recurrent units

There are many different ways of designing the feedback connections in an RNN. Recurrency is obtained by any computational unit that accepts output from a previous time step as input. In the current project, three types of recurrent cells are considered. All three are schematized in Figure 5.1, and will now be introduced in more detail.



(a) SRN (b) GRU (c) LSTM

Figure 5.1: Schematic visualization of three recurrent units. Switch signs represent gates, $\otimes$ element-wise product and $\oplus$ summation. $\mathbf{x}$ and $\mathbf{y}$ are input and output vectors.

5.1.1 Simple Recurrent Network

The most basic recurrent unit is the Simple Recurrent Network (SRN), also known as the Elman network because it was introduced by Elman 1990. See Figure 5.1a. At a time t, an SRN has a hidden state $\mathbf{h}_t$, which is computed from the input $\mathbf{x}_t$ at the same time step and the previous hidden state, $\mathbf{h}_{t-1}$. The hidden state of time $t-1$ constitutes the state $\mathbf{s}_t$ at time t:

$$\mathbf{s}_t = \mathbf{h}_{t-1} \tag{5.1}$$

$\mathbf{s}_t$ is also known as the context unit. It can be interpreted as the memory of the cell. The new hidden state is calculated as:

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{s}_t + \mathbf{b}), \tag{5.2}$$

where $\mathbf{W}$ denotes the $h \times d$-dimensional weight matrix connecting the d-dimensional input $\mathbf{x}_t$ to the h-dimensional hidden unit $\mathbf{h}_t$. $\mathbf{U}$ is the $h \times h$-dimensional matrix containing the internal hidden layer weights and $\mathbf{b}$ is a bias term. A tanh nonlinearity is applied to the entire sum.

5.1.2 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a more advanced recurrent cell introduced by Chung et al. 2014. It is schematized in Figure 5.1b. The switch signs in the figure represent additional internal states known as 'gates'. They are used to organize the information flow in a more controlled way than the rudimentary feedback loop of the SRN. Individually, all gates are just regular feed-forward neurons that apply a nonlinearity to a weighted sum of the input vectors. In the GRU case, $\mathbf{r}$ is a reset gate used to compute the h-dimensional preliminary hidden state $\widetilde{\mathbf{h}}_t$ at time t:

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{5.3}$$

As before, for d-dimensional input $\mathbf{x}_t$ at time t, $\mathbf{W}_h$ is the $h \times d$-dimensional weight matrix to the candidate hidden state. $\mathbf{U}_h$ is the $h \times h$-dimensional matrix storing the internal hidden weights, $\mathbf{r}_t$ is the reset gate value at time t, $\mathbf{h}_{t-1}$ is the previous hidden state and $\mathbf{b}_h$ an h-dimensional bias vector. $\otimes$ is the element-wise product, also known as the 'Hadamard product'. $\mathbf{r}$ is called the 'reset gate' because it determines how much of the previous hidden state is remembered and used in computation. At time t it takes the following value:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{5.4}$$

$\mathbf{W}_r$ are $\mathbf{U}_r$ are weight matrices of dimensionality $h \times d$ and $h \times h$, respectively. $\mathbf{b}_r$ is another bias term. σ is the logistic sigmoid function that is used as nonlinear activation at the gates. The value of gate $\mathbf{z}$, the update gate, at time t is the result of a similar computation:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{5.5}$$

$\mathbf{W}_r$ and $\mathbf{U}_r$ are weight matrices of the same dimensions as before, $\mathbf{b}_z$ is a new bias term. The gate value $\mathbf{z}_t$ establishes the extent to which the previous hidden state $\mathbf{h}_{t-1}$ is retained, or updated with $\widetilde{\mathbf{h}}_t$:

$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \widetilde{\mathbf{h}}_t \tag{5.6}$$

5.1.3 Long Short-Term Memory

Finally, here follows a description of the unit shown in Figure 5.1c: the Long Short-Term Memory (LSTM). The LSTM is a predecessor of the GRU, introduced by Hochreiter and Schmidhuber 1997. It has more gates, and therefore more parameters, than the GRU. To be precise, it has a forget gate $\mathbf{f}$, an input gate $\mathbf{i}$ and an output gate $\mathbf{o}$. It also has an extra component $\mathbf{c}$, which serves as memory cell. As with the GRU, all gates are traditional neurons, which apply the logistic sigmoid function σ to a linear transformation of the input vectors. At time t, they take the following values:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{5.7}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{5.8}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{5.9}$$

As usual, $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ denote the input vector at time t and the hidden state at time $t-1$. For d-dimensional input and h hidden units, weight matrices $\mathbf{W}_f, \mathbf{W}_i$ and $\mathbf{W}_o$ are $h \times d$-dimensional, and internal hidden-to-hidden matrices $\mathbf{U}_f, \mathbf{U}_i$ and $\mathbf{U}_o$ are $h \times h$-dimensional. $\mathbf{b}_f, \mathbf{b}_i$ and $\mathbf{b}_o$ are h-dimensional bias vectors for the three gates. The memory cell $\mathbf{c}$ is computed at time t as:

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{5.10}$$

with $\mathbf{W}_c$ an $h \times d$- and $\mathbf{U}_c$ an $h \times h$-dimensional weight matrix, and $\mathbf{b}_c$ a bias vector of length h. The new hidden state is now computed as:

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t) \tag{5.11}$$

The functions of the different LSTM components can best be understood with reference to Figure 5.1c. The $\otimes$-symbols in the diagram all denote element-wise multiplication. $\oplus$ is ordinary matrix addition. The forget gate $\mathbf{f}$ can selectively obliviate information from the previous cell state, up to total annihilation in the hypothetical case of a weight matrix containing only zeroes. With a similar mechanism, the input gate $\mathbf{i}$ determines which values of the cell state must be updated. In the GRU, the update gate has a role that is comparable to a combination of the LSTM forget and input gates. The output gate $\mathbf{o}$ modifies the final result before it is returned and passed on to the next cell state.

So far, all gates were characterized as single-layer neurons, but their definitions can be straightforwardly generalized to multi-layer analogues by adding sets of parameters for extra layers. The weight matrices and biases of all recurrent units are trainable by means of an adapted version of traditional backpropagation. This method is called 'backpropagation through time' and was independently established in several publications. See Robinson and Fallside 1987, Werbos 1988 and Mozer 1989.

5.2 New architecture

In PyTorch, a new architecture is implemented that replaces the tree-shaped sentence composition of the tRNN and tRNTN models with a sequence of recurrent units. An abstract visualization of the new set-up is given in Figure 5.2. The upper part of the model is identical to Bowman's (see Figure 3.3 for comparison). It consists of a comparison layer and a classification layer, after which a softmax function is applied to determine the most probable target class. As before, the comparison layer takes the concatenation of two sentence vectors as input. However, these vectors are no longer produced by a composition function, but returned as final output of a series of recurrent cells.

The number of cells equals the number of words, so it differs per sentence. The new set-up resembles the Siamese architecture for learning sentence similarity of Mueller and Thyagarajan 2016b and the LSTM classifier described in Bowman, Angeli, et al. 2015, although the latter contains more upper layers and fewer output units.

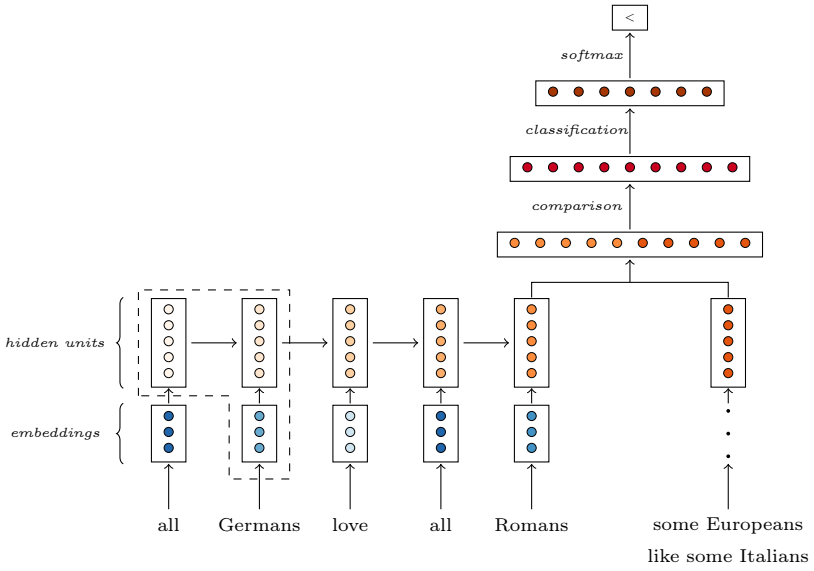


Figure 5.2: Visualization of the general recurrent model. The region in the dashed box represents any of the recurrent cells schematized in Figure 5.1, which is repeatedly applied until the final sentence vector is returned.

In the diagram, the dashed box indicates the location of an arbitrary recurrent unit. The specific computations that take place in the unit depend on its type, which can be any of those listed in Section 5.1. Recall the schematizations of the SRN, GRU and LSTM in Figure 5.1, all of which can be inserted in the dashed region of the high-level overview in Figure 5.2. The generic input vector $\mathbf{x}$ in the diagrams of the individual recurrent units now takes the shape of a word embedding. More precisely, at each time step t, the input $\mathbf{x}_t$ is the vector representation of the word at position t in the sentence. The final output $\mathbf{y}$ is the sentence vector, which is returned after an entire sentence has been processed.

The number of hidden units (the dimensionality of the hidden vectors, denoted by h in Section 5.1) is fixed at 128. The dimensionality of the word embeddings is 25, as before, and the comparison layer remains 75-dimensional. By default, all recurrent units are single-layered.[1] Prior to training, all hidden units are initialized as zero vectors. Network parameters are initialized according to the default PyTorch routine. Weights of the recurrent units are drawn from the uniform distribution $\mathcal{U}(-1/\sqrt{h}, 1/\sqrt{h})$. In our case $h = 128$, so the lower bound is $-1/\sqrt{128}$ and the upper bound $1/\sqrt{128}$. Non-recurrent parameters, belonging to the linear comparison and classification layers, are initialized

[1]Experiments were conducted with multi-layered recurrent units, but this did not improve results.

uniformly randomly according to distribution $\mathcal{U}(-1/\sqrt{fan_{in}}, 1/\sqrt{fan_{in}})$, where fan_{in} denotes the number of input units. The comparison layer is initialized according to $\mathcal{U}(-1/\sqrt{50}, 1/\sqrt{50})$, because its input is the concatenation of two 25-dimensional sentence vectors, and the initial classification layer weights are drawn from $\mathcal{U}(-1/\sqrt{75}, 1/\sqrt{75})$, because the comparison layer outputs are 75-dimensional. Word embeddings are initialized according to a normal distribution with mean 0 and standard deviation 1, i.e. $\mathcal{N}(0, 1)$.

During training, AdaDelta is used as optimizer. L2 is not applied as regularization method for the recurrent units, because it reduces their memorizing capacities (Pascanu, Mikolov, and Bengio 2013). Experiments were conducted with dropout layers, which perform regularization by zeroing elements of input vectors with some preset probability p during training (Bluche, Kermorvant, and Louradour 2015). However, hyperparameter tuning showed that best results were obtained for probability $p = 0$, which equals no dropout, so no such layers are used in the final experiments.

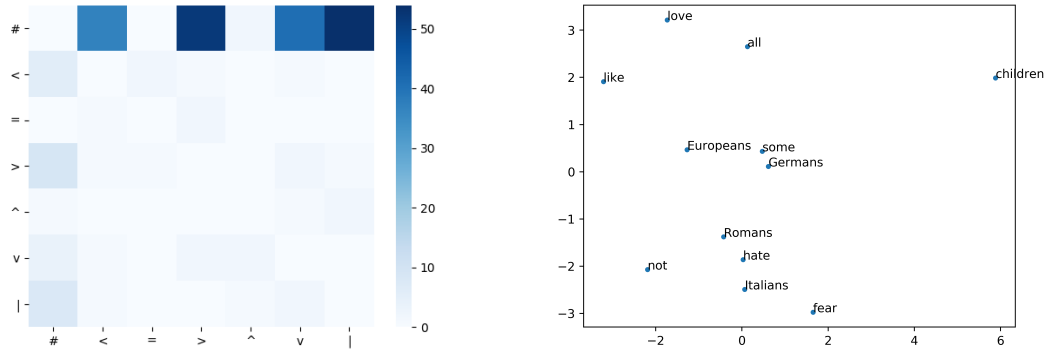
Three types of models, containing the three different kinds of recurrent units, are trained for five runs on the `binary_fol` data, while completely ignoring brackets. Training is always continued for 50 epochs, and results are reported after termination. The final scores, averaged over all five runs, are given in Table 5.1.

	train	test
128/25/75 SRN	97.9	87.3
128/25/75 GRU	100.0	96.0
128/25/75 LSTM	100.0	94.6

Table 5.1: Training and testing accuracy scores of models with the three different recurrent units on the FOL inference task for the `binary_fol` data. Results are averaged over five runs. An $h/n/m$ model has h hidden units, n-dimensional word embeddings and an m-dimensional comparison layer.

Comparison with the results in Table 4.6 shows that all recurrent models outperform the summing baseline. Even the simplest recurrent network, the SRN, achieves higher training and testing accuracy scores than the tree-shaped matrix model. The GRU and LSTM even beat the tensor model. The LSTM obtains slightly lower scores than the GRU, which is unexpected given its more complex design, but perhaps the current challenge does not require separate forget and input gates. It must be noted that comparison between the training accuracies of Table 5.1 and earlier ones can be misleading. More regularization was applied while training the tree-shaped models, which tends to decrease the training accuracy.

The consistently higher testing accuracy, however, provides evidence that the recurrent networks are not only capable of recognizing FOL entailment relations between unseen sentences, but that they can also outperform the tree-shaped models on this task. This is a surprising result, because the recurrent models do not use any of the symbolic structure that seemed to explain the success of their recursive predecessors. The recurrent classifiers have learned to apply their own strategy. In what follows, I will



(a) (b)

Figure 5.3: (a) Unnormalized confusion matrix with respect to the `binary_fol` test data for the best-performing GRU model, with diagonal entries (correct classifications) omitted. (b) PCA projection of best-performing GRU's learned word embeddings.

investigate the details of this method, and juxtapose it with the recursive mechanism of the tree-shaped networks.

5.3 Interpretation

As before, I first focus on the most successful single model. This is a GRU, which obtained a final testing accuracy of 96.8% and a training accuracy of 100.0%.[2] Recurrent neural networks are notoriously hard to interpret, which is why they are often referred to as 'black boxes'. In what follows I will attempt to shed some light on their intricate dynamics.

Confusion matrix The winning GRU's confusion matrix is so strongly diagonal that it does not even reveal which categories are relatively most error-prone. For that reason, it is not included here. Figure 5.3a is more informative in this respect, because it is not normalized and omits the correct classifications of the diagonal entries, as was done for the best-performing tRNTN in Figure 4.4b. The manifested error pattern is totally different from the one produced by the winning tensor model. The vast majority (90%) of all errors made on the test set is related to #, comprising misclassifications of items labelled with independence, and attribution of independence to sentence pairs with a different target. Relative to this class, other error categories are virtually negligible. It

[2]To verify that the recurrent units are responsible for the success of the GRU, and not the feed-forward upper layers of the network, a random GRU was tested. For this experiment, the recurrent weights of the best-performing GRU were randomized, but the trained upper layers remained unchanged. The mean testing accuracy of five such models was no more than 22.0%, confirming that the positive results are not attributable to the upper layers alone.

seems that the GRU has learned to distinguish quite accurately between the relations within the semantic clusters of Figure 4.4d, which proved to be a source of confusion for the tensor model. The remaining #-related errors are most difficult to overcome, because of their diversity. The GRU hardly ever confuses independence with equivalence or alternation. From the distribution in Figure 4.4c, recall that these are the two rarest labels.

Word embeddings Next, I examine the word embeddings. Figure 5.3b shows the PCA projection of the representations learned by best-performing GRU after 50 training epochs. Much of the geometric distribution matches the semantic order. Consider, for instance, that 'love' and 'like' are clustered relatively far away from 'fear' and 'hate', that 'Romans' and 'Italians' are close together, and that 'children' is an outlier. Some of these observations also applied to the word embeddings of the tRNTN shown in Figure 4.5a. An important difference between the tRNTN and the GRU is consistency across models. Figure 4.5b demonstrated that the distribution of the final word embeddings did not differ a lot across model runs for the tRNTN. This is not the case for the GRU. Although similar clusters are often discernible, their distribution over the word vector space is completely different for each fully trained model. In this respect the GRU models are far less convergent than the tRNTNs.

Hidden vectors Figure 5.4 is the recurrent analogue of Figure 4.6. It shows the PCA projection of the vectors representing the subsequent stages of processing a single sentence. Only now, the plots do not show how a composition function recursively maps word embeddings to a sentence vector, but how a sequence of recurrent units outputs hidden vectors, the last one of which represents the entire statement. Arrows indicate the direction in which the recurrent units are applied, following the order of the sentence.

The four examples in Figure 5.4 are taken from the sentence pairs in the `binary_fol` test set that are correctly classified by the best-performing GRU. Because this model has 128 hidden units, applying PCA to find a two-dimensional projection requires that only two out of 128 principal components are maintained. This is a heavy reduction, so it would be premature to base any major conclusions on these plots. They do, however, clarify some details that are particularly interesting in relation to the tRNTN sentence composition visualized in Figure 4.6. For instance, the trajectories do not map subsequent vectors into increasingly remote regions of the space, as opposed to the tensor model's composition function. Indeed, there seems to be no rigid separation between the subspaces containing sentence vectors and those containing intermediate or initial representations. In this regard, the GRU appears less geometrically sensitive to grammatical distinctions than the tRNTN. Also, looking at the distances between the items per recurrent path, large steps are usually caused by quantifiers, verbs and negation, while nouns tend to have a smaller impact. It remains to be seen whether this signals a proportional contribution to the classifier's performance.

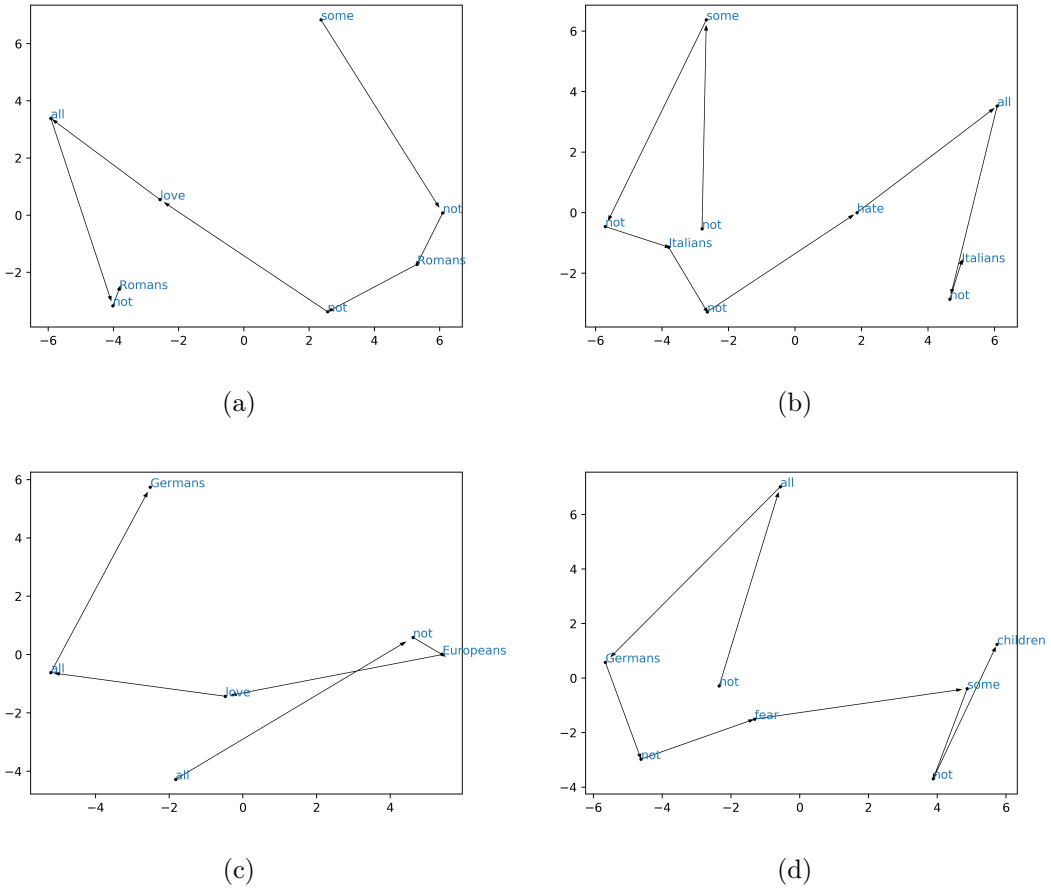
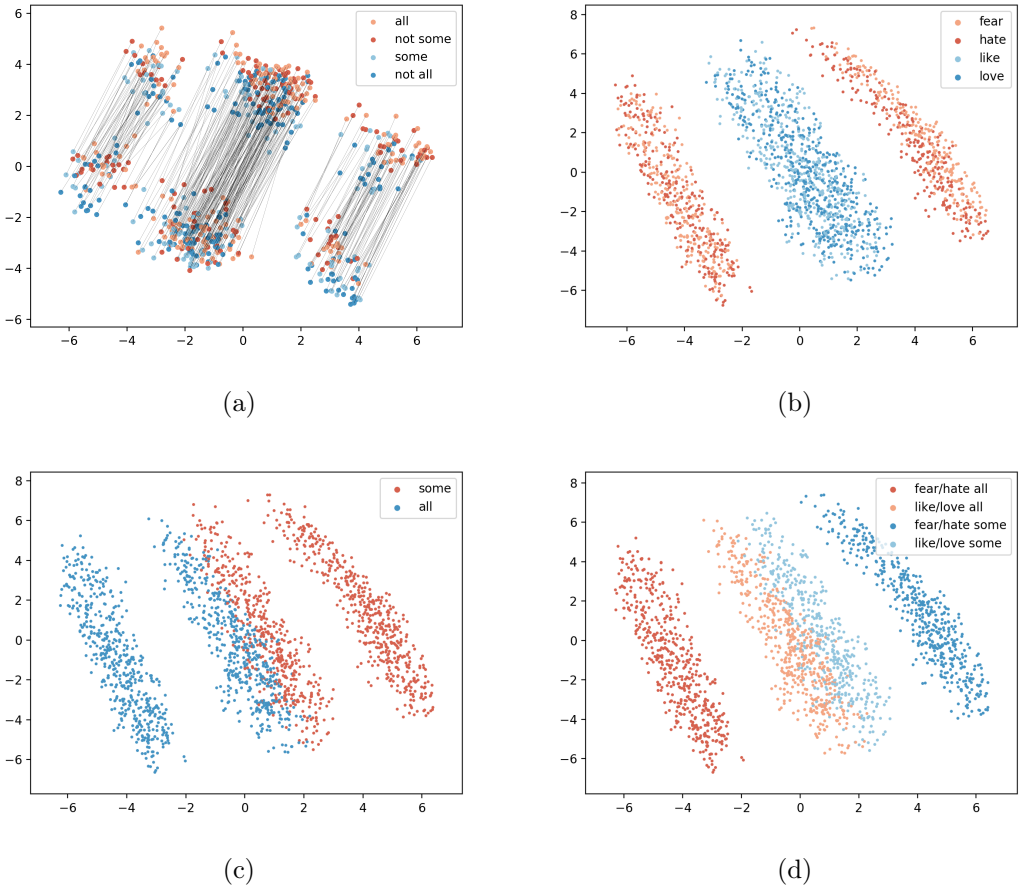


Figure 5.4: PCA projections of the hidden vectors generated by the best-performing GRU while processing the sentences (a) some not Romans not love all not Romans, (b) not some not Italians not hate all not Italians, (c) all not Europeans love all Germans, and (d) not all Germans not fear some not children. Projections are annotated with the corresponding vocabulary items from $\mathcal{L}_N$.

Sentence vectors Figure 5.5 includes four plots showing PCA projections of sentence vectors generated by the best-performing GRU. The first one, Figure 5.5a, shows a sample of approximately 500 correctly classified sentence pairs from the `only_negations` data set, which only contains pairs of the form $\langle \varphi, \neg\varphi \rangle$, labelled with $^\wedge$. Projected vectors are colored according to the negatable first quantifier of the represented sentence. Apart from some minor clustering of existentially and universally quantified propositions, an even stronger mirroring effect takes place than with the tRNTN (shown in Figure 4.7b). Negated sentences are not symmetric with respect to a point, but with respect to a line that forms an almost perfect linear boundary between propositions and their logical complements. In this sense, the GRU has realized a geometry that captures the semantics



(a) (b)

(c) (d)

Figure 5.5: PCA projections of sentence vectors produced by the best-performing GRU. (a) 400 pairs of negated sentences, colored according to their first (possibly negated) quantifier. (c) 4,000 sentences, colored according to their verb. (d) 4,000 sentences, colored according to their second quantifier. (e) 4,000 sentences, colored according to the combination of their verb and second quantifier.

of negation even more intuitively than the tree-shaped tensor model managed to do. This is especially remarkable when taking into consideration that negation of a sentence currently reduces to negation of its first quantifier, which is just a matter of adding (or removing) 'not' as first word of the sentence. At the same time, this modification swaps the truth value of the complete sentence. In other words, we have here a single word at the very beginning of a statement, capable of manipulating the semantics at sentence level. Even though the GRU processes expressions sequentially, it has managed to recognize the global importance of this first word, which is memorized during the recurrent processing, and assigned sufficient weight to effectuate the strongly symmetric ordering of Figure 5.5a. This is evidence that the GRU not only remembers important

historic inputs, but that it also distinguishes items that are semantically significant and organizes the sentence space according to such items, even if they are observed at the very beginning of a sequence.

Figure 5.5a already suggests that the sentence vectors, irrespectively of their first quantifier or its sign, form three major clusters. This is confirmed when a higher number of vectors is projected, representing sentences belonging to correctly classified pairs in the `binary_fol` test set. Figures 5.5b-5.5d include samples of approximately 4,000 such sentence vectors. Even more clearly than those produced by the tRNTN (shown in Figure 4.8), these projections form three distinct, linearly separable clusters. Coloring the individual points according to a range of grammatical properties of the represented sentences provides more insight into the nature of these configurations.

In particular, it turns out that sentences containing 'like' or 'love' as verb are situated in a different subspace than those containing 'fear' or 'hate', as illustrated in Figure 5.5b. This is reminiscent of Figure 4.8a, but then without overlap between the different regions. Furthermore, there now seems to be an explanation for the division between the two 'fear/hate' clusters. Consider Figure 5.5c, which colors the plotted vectors according to the second quantifier of the underlying sentences. This is clearly an ordering factor of the sentence vector geometry, with overlap between the two classes only occurring in the middle region, which has two totally separate clusters located on either side of it. Because Figures 5.5b and 5.5c visualize different samples, it is not necessary (although likely) that the three major clusters in Figure 5.5c coincide with the verb clusters of Figure 5.5b. To test this, Figure 5.5d visualizes another sample, with vectors colored according to the *combination* of the verb and second quantifier of the represented sentences. For clarity, 'fear' and 'hate' are considered together, as are 'like' and 'love'. The new plot confirms the suspicion that the three clusters of Figure 5.5c are indeed the verb clusters identified in Figure 5.5b, and shows that a grammatical feature has been found that appears to capture all major clusters in the sentence vector space. So far, no other sentence constituents (or combinations thereof) have been found to correspond with notable patterns in the sentence space. Apparently, the GRU is similar to the tRNTN in this respect, that the verb and the second quantifier are the grammatical features with most impact on the sentence vector distribution, although the geometry as a whole is better understood in the case of the GRU.

5.4 Diagnostic classification

The PCA projections of word embeddings, hidden units and sentence vectors shed light on certain aspects of the internal mechanisms of the GRU. Most of all, they facilitate a qualitative analysis of clustering and partitioning among learned representations, as performed in the previous section. Although this revealed some interesting facts, it has not taught us much about the general 'method' used by the GRU to classify entailment. Due to their highly specific topology, the tree-shaped networks have no option but to process sentences recursively, collapsing their syntactic structure branch by branch until a sentence vector is returned. The recurrent models have no such guidance, but also

perform very well on the entailment recognition task. I would like to know more about the strategy they apply 'under the hood', in order to explain their success.

Understanding the internal dynamics of a recurrent model is not straightforward, due to the high numbers of parameters and hidden units that are involved. One way of opening the black box is to study the hidden units in terms of specific symbolic hypotheses to find out if they have learned to encode details corresponding with particular linguistic facts. In order to do so as comprehensively as possible, I make use of a technique called *diagnostic classification*, introduced by Hupkes, Veldhoen, and Zuidema 2017:

> To test whether a network is computing or representing a certain variable or feature, I determine the sequence of values that this variable or feature takes at each step while processing a sentence. I then train an additional classifier – a diagnostic classifier – to predict this sequence of variable values (representing our hypothesis) from the sequence of hidden representations a trained network goes through while processing the input sentence. (Hupkes, Veldhoen, and Zuidema 2017, p. 14)

Diagnostic classifiers can provide the basis for a quantitative analysis of recurrent models that is more precise than the visual, often somewhat handwavy interpretation of plotted projections. The method is also more systematic than the approach of e.g. Karpathy, Johnson, and Fei-Fei 2015, who visualized cell and gate activations as part of a qualitative investigation. The only required assumption is that predictability of some linguistic hypothesis by a diagnostic classifier trained on the hidden representations of a recurrent model indicates that this information is truly encoded by that model. This should be an uncontroversial supposition, given the fact that diagnostic classifiers are very simple, linear tools. If such basic models can successfully predict a symbolic hypothesis from a sequence of hidden vectors, this serves as evidence that the recurrent model actually computes that information. If they are unable to do so, this indicates that the model does not encode such information, and is therefore unlikely to base any internal strategy on it.

Logistic regression is used as diagnostic classification method in all experiments reported in this section. The entire `binary_fol` data set is presented to the best-performing GRU, in order to retrieve the hidden vectors produced following each word of every sentence in the data. These vectors are labelled according to some symbolic feature (the hypothesis) that is heuristically derived from the data. The result is a separate data set for each hypothesis, containing 128-dimensional vectors with their corresponding labels. Duplicate items are removed from the data, and the order is randomized. 80% of the remaining instances are used to train the diagnostic classifier ($\sim$ 11,000 items). The other 20%, unseen during training, is used for testing ($\sim$ 3,000 items). Four hypotheses are considered, and discussed below.

Recursive depth The tensors models owe their success to their recursive structure, so it is surprising to see that GRUs, which are wholly sequential, perform at least as well on the same entailment classification task. Could it be that the recurrent models

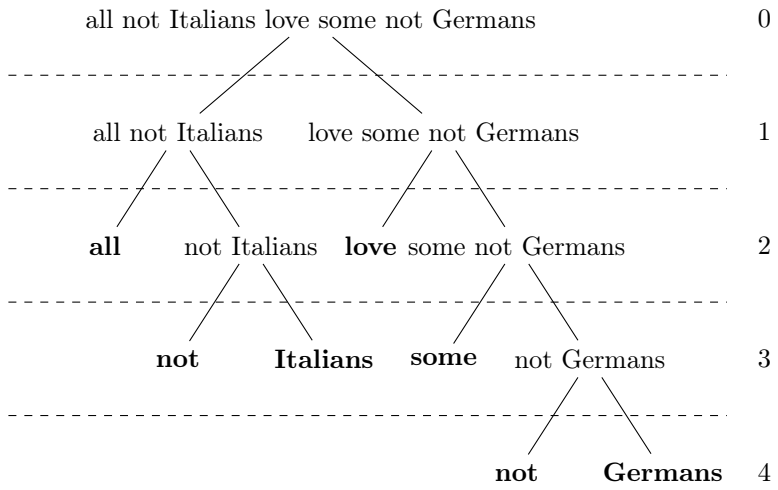


Figure 5.6: Illustration of recursive depth for sample sentence 'all not Italians love some not Germans'.

are internally applying a recursive strategy of their own? In Hupkes, Veldhoen, and Zuidema 2017, this hypothesis is tested on GRUs that have learned to evaluate arithmetic expressions. Consider e.g. the statement '$(5+(2-3))$'. There are two symbolic strategies to solve such a sum: a cumulative and a recursive one. A cumulative computer calculates the tentative result at each point in the expression. In this example, she would thus produce the intermediate outcomes 5, 7 and 4. In the recursive strategy, expressions at different branches (between brackets) are first evaluated separately. Here, this means that the initial value 5 has to be remembered while '$(2-3)$' is assessed and concluded to equal -1. Next, the computation '$5-1$' is performed and the final result 4 is returned.

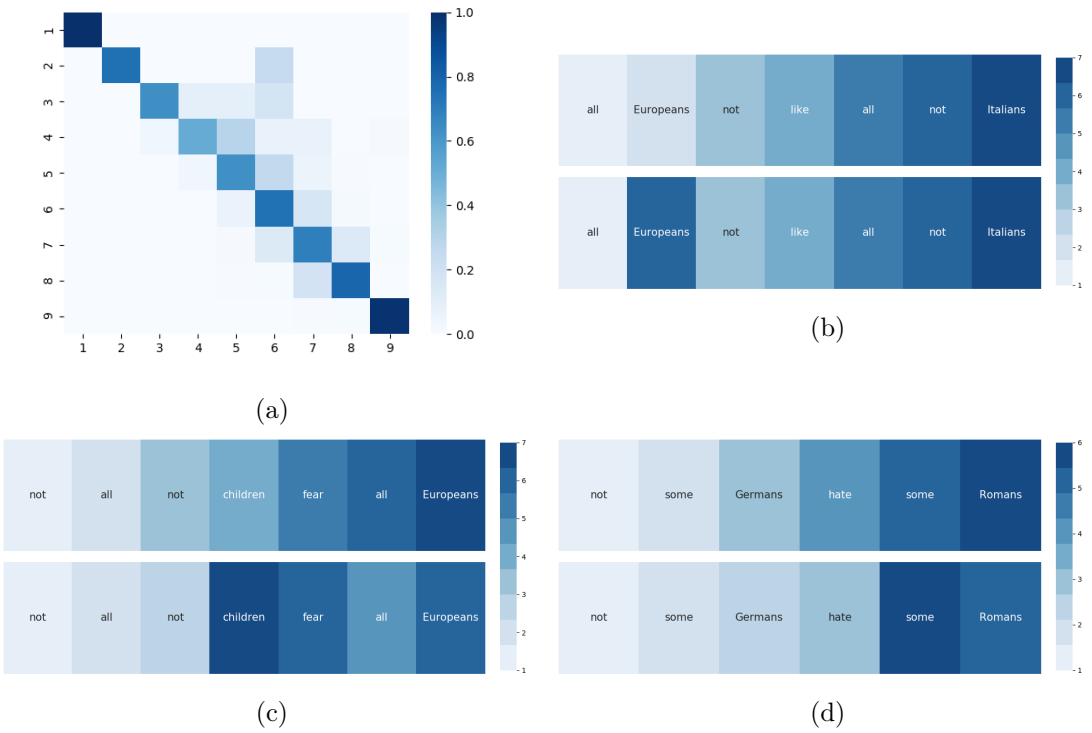
In the case of the `binary_fol` data, there is no direct equivalent of intermediate outcomes associated with either a cumulative or a recursive strategy. It still makes sense, however, to wonder if the GRUs apply an analogue of the recursive method. If so, this would mean that they have developed an internal parsing mechanism, without the supervision of the brackets needed by the tree-shaped models to structure their computation graph. Moreover, a 'proclivity for tree structures' has been considered 'central to human cognition' (Fitch 2014), so it is worthwhile to investigate if GRUs also have such a tendency.

By lack of a more precise evaluation method to label the individual branches in a sentence tree, what I test here is recursive awareness, understood as predictability of recursive depth. Figure 5.6 illustrates this notion for a sample sentence. Every sentence constituent is located at a specific location in the binary parse tree, whose depth is indicated by the numbers on the right hand side. What matters here is only the depth of the individual words, shown in bold typeset, because these are the units presented to the GRU. Each hidden vector is labelled with the recursive depth of the token that produced it. If a diagnostic classifier is able to predict that depth for unseen instances,

this is regarded as evidence that the GRU has some recursive awareness.

It turns out that a diagnostic classifier can predict recursive depth very well: the mean squared error (MSE) on the test set is 0.01, and the accuracy 99.1%. The fact that simple logistic regression suffices to derive recursive depth from the hidden vectors with nearly perfect accuracy indicates that the GRU is not indifferent with respect to this information. Yet, it would be exaggerated to conclude that the model performs a recursive strategy from this result alone. The number of classes is low, because individual words can only be located at depth 2, 3 or 4, and encoding of recursive information does not imply that the entire process is guided by a tree-oriented mechanism.

Word index The next hypothesis to be considered is word index. I test if a diagnostic classifier can infer the sentence position of a token from the hidden vector that was produced upon its observation. If this is possible, it suggests that, while processing a sentence, the GRU keeps track of its location in a linear fashion. This property seems best associated with a cumulative approach.



(a) (b) (c) (d)

Figure 5.7: (a) Row-normalized confusion matrix of the diagnostic classifier trained to predict word indices from hidden vectors of the best-performing GRU, with respect to the test set. (b)-(d) Visualization of the diagnostic classifier's performance on three sample sentences, with correct labelling (hypothesis) shown in the upper rows, and predictions below.

Sentence length ranges from five words (with no negations) to nine (with four negations), so classification of word index is a 9-class problem. The diagnostic classifier obtains an MSE of 0.36 on the test set. The testing accuracy is 75.3%, which is not extremely high, but the confusion matrix in Figure 5.7a shows that wrongly predicted positions do tend to be close to the targets. For an index i, the most common mistakes are attribution of either $i - 1$ or $i + 1$, so errors made by the classifier are usually not very serious.

The examples in Figures 5.7b-5.7d confirm this. For three sample sentences, these plots show the correct label (the hypothesis) per word in the upper row. The lower row shows the same sentence, but this time with coloring according to the labels predicted by the diagnostic classifier. Thus, they provide a more small-scale illustration of the kind of errors made by the classifier. The figures demonstrate that, indeed, most of the errors are quite close to the hypothesis, and that the predicted index generally increases while iterating over the sentence words from start to finish. All in all, it seems clear that the GRU roughly keeps track of its position in an expression while processing it.

Semantic type I also check if a diagnostic classifier can be trained to predict the semantic type of a sentence constituent from the corresponding hidden vector. Four types are distinguished: quantifiers, nouns, verbs and negations, which are the four groups of vocabulary items in $\mathcal{L}_N$ specified on page 41. If a simple classifier manages to accurately infer the type of a word from the hidden vector produced with that word as input, this suggests that the GRU is 'conscious' of basic grammatical categories.

As it turns out, the diagnostic classifier is hardly challenged by this task. On the test set it achieves an accuracy of 99.9%. This means that virtually all testing instances have been labelled correctly according to their semantic type, so the confusion matrix is almost totally diagonal. It appears that this information is easily derivable from the hidden vectors. Then again, sentence structure is still rather rigid, so words of a given type usually end up in the same regions of a statement. Therefore, it is impossible to exclude the possibility that the classifier has learned to recognize correlations with particular positions rather than semantic categories. If sentence structure were more flexible, diagnostic classification would prove more conclusive in this respect.

Monotonicity direction Another interesting phenomenon to address diagnostically is monotonicity. As described in Section 2.2.1, monotonicity concerns the truth-preserving replaceability of quantifier arguments with weaker or stronger notions. The sentences in $\mathcal{L}_N$ have quantifiers at two positions. The first, at the beginning of a sentence, takes the first noun and the verb as its arguments. The second, towards the end of a sentence, has only the second noun as an argument. Both quantifiers are taken from the set $\mathcal{Q}^{\mathcal{L}_N} = \{\texttt{all}, \texttt{some}\}$, and only the first one is negatable. Examples of the four most basic configurations that can thus be constructed (without negations) are shown in Table 4.5. For these standard sentences, I evaluate the monotonicity of the quantifier arguments in Table 5.2. Upward monotonicity is denoted by an upward arrow (↑), downward monotonicity by a downward arrow (↓). Brackets are omitted, as these are not considered by

the recurrent models.

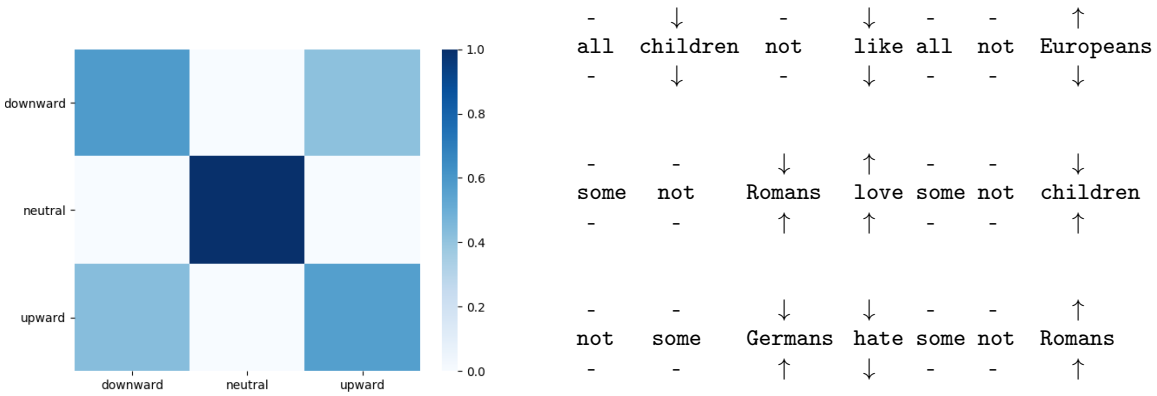
	↓	↑		↓
all	Europeans	love	all	Germans
	↓	↑		↑
all	Europeans	love	some	Germans
	↑	↑		↓
some	Europeans	love	all	Germans
	↑	↑		↑
some	Europeans	love	some	Germans

Table 5.2: Monotonicity properties of quantifiers in basic $\mathcal{L}_N$ sentences.

As indicated in the table, a universal quantifier in the first position is downward monotone in its first and upward monotone in its second argument, while an existential quantifier at the beginning of a sentence is upward monotone in both its arguments. This is not surprising, because it is generally the case for the natural language analogues of these quantifiers (Keenan 1996), and $\mathcal{L}_N$ is supposed to resemble natural English. A universal quantifier in the second position is downward monotone, and an existential one upward monotone. Both nouns and the verb can be prefixed with a negation that modifies only these individual words. Negation reverses monotonicity, so it can be integrated in any of the patterns of Table 5.2 by simply reversing the direction of the arrow above the negated word. The second quantifier is not negatable, but the first one is. As the entire sentence is in the scope of the first quantifier, the full expression is semantically affected upon its negation. This is accommodated by reversing the direction of all arrows.

I now have a heuristic procedure to determine the monotonicity direction of the quantifier arguments in all $\mathcal{L}_N$ sentences. It is used to construct a new diagnostic data set, which labels hidden vectors according to the monotonicity direction of the token it took as input. Words with no monotonicity properties (without arrows in Table 5.2) are assigned label 'neutral'.

The diagnostic classifier does not excel at predicting monotonicity directions. On the test set, it obtains an accuracy score of only 62.5%. It manages to recognize most neutral words, but constantly confuses upward and downward monotone quantifier arguments. This is illustrated by the confusion matrix of Figure 5.8a, and by the sample sentences in Figure 5.8b. Given a target of either direction, the classifier only has a slight preference for the correct label. The high accuracy for neutral terms is as expected, because the distinction between neutral and monotone items reduces to the distinction between $\mathcal{Q}^{\mathcal{L}_\mathcal{N}} \cup \mathcal{A}^{\mathcal{L}_\mathcal{N}}$ (quantifiers and negations) on the one hand, and $\mathcal{N}^{\mathcal{L}_\mathcal{N}} \cup \mathcal{V}^{\mathcal{L}_\mathcal{N}}$ (nouns and verbs) on the other hand. The results of the diagnostic classifier trained to predict semantic type already showed that the difference between these categories is encoded in the hidden vectors, so distinguishing their unions should be even easier. The inability of a linear model to discriminate between upward and downward monotonicity indicates that this logical-linguistic feature does not contribute to the GRU's success. The GRU may have integrated an analogue of some symbolic system under the hood, but most probably not



(a) (b)

Figure 5.8: (a) Row-normalized confusion matrix of the diagnostic classifier trained to predict monotonicity directions from hidden vectors of the best-performing GRU, with respect to the test set. (b) Visualization of the diagnostic classifier's performance on three sample sentences, with correct labelling (hypothesis) shown above each sentence, and predictions shown below.

a monotonicity calculus.

Scope of negation Finally, a data set is constructed in which the hidden vectors are labelled according to the sign of the corresponding words. In this context, 'sign' indicates whether or not a word is negated. In the syntax of $\mathcal{L}_N$, negation acts on individual lexical items, so for each word one can ask if it is in the scope of a negation. The heuristics to determine whether a word is negated (and label the data) are simple: it suffices to check if the word 'not' immediately precedes it. The diagnostic classifier will show if this information can be inferred as easily from the GRU's hidden vectors.

I find that a logistic regression model has no difficulty to predict whether a hidden vector is produced within or outside the scope of a negation. The diagnostic classifier's testing accuracy is 99.5%, so only a handful of errors is made and the confusion matrix is practically diagonal. This demonstrates that negation marks the hidden state, and leaves a trace that is maintained while the GRU is processing the item in its scope. This is not surprising in the lights of Figure 5.5a, which already showed that the GRU is very aware of negations at sentence level. Vectors representing pairs of sentences and their negations are almost linearly separable, and a similar partitioning appears to hold between hidden vectors representing negated words on the one hand, and unnegated items on the other.

5.5 Compositional learning

The diagnostic classifier analysis has shown that the GRU is more aware of some logical-linguistic notions than of others. This indicates that, if the model internally applies a strategy that could be mapped to some symbolic analogue, such a method is probably describable in terms of these or related concepts. Another question about the recurrent models' functioning does not concern the way in which they learn, but the type of learning that is applied. Research of recent years has made it clear that these models have a strong generalizing potential, as is confirmed by the results reported in this project. However, the great amount of data required for training raises questions about the kind of generalization that takes place.

An important question to ask is whether neural models in general, and recurrent ones in particular, apply any compositional skills in their learning processes. Compositionality is the ability to interpret and generate a possibly infinite number of constructions from known constituents, and is commonly understood as one of the fundamental aspects of human learning and reasoning (Chomsky 1957; Montague 1970). It has often been claimed that neural networks operate on a merely associative basis, lacking the compositional capacities to develop systematicity without an abundance of training data. See e.g. Fodor and Pylyshyn 1988, Marcus 1998, Calvo and Symons 2014. Especially recurrent models have recently been regarded quite sceptically in this respect, following the negative results established by Lake, Ullman, et al. 2017 and Lake and Baroni 2017. Their research suggests that recurrent networks only perform well provided that there are no systematic discrepancies between train and test data, whereas human learning is robust with respect to such differences thanks to compositionality.

In this section, the recurrent architectures introduced at the beginning of the chapter are studied in terms of their compositional abilities, in order to assess whether the negative results obtained in the above-cited research apply in our case. Because compositionality is such a broad concept, the models are subjected to three separate experiments, each of which focuses on a different aspect.

5.5.1 Recursive cues

So far, all recurrent models have been trained and tested without access to the parse trees of the sentences. This was done by removing all brackets from the `binary_fol` data. The results of Table 5.1, and especially those obtained by the GRU, prove that recurrent models can successfully address this entailment classification task, even if no symbolic guidance in the form of brackets is available. This means that there exists a sequential approach to the challenge, which contrasts the recursive strategy applied by the tree-shaped networks of Chapter 4.

The compositional nature of language $\mathcal{L}_N$, used to construct the `binary_fol` data, is manifest in the bracketed data. The topology of the tree-shaped networks can be regarded as an imposed form of compositional learning, because it forces the models to construct the meaning of complex expressions from individual components according to their explicitly provided syntactic form. The recurrent models are free to design their

own internal algebra. Yet, under the hood, they may well have learned to distinguish compositional patterns resembling those specified by the parse trees. It is known that recurrent models can perform parsing, as shown by e.g. Vinyals et al. 2015. Not only the high accuracy scores on the test set indicate that the recurrent models have internalized structural properties of the sentences in $\mathcal{L}_N$. Also the diagnostic classifier that predicts recursive depth with more than 99% accuracy demonstrates that compositional structure of input expressions is latently present in the hidden vectors of the GRU, as discussed in Section 5.4.

Could this mean that the recurrent models apply compositional learning by uncovering and exploiting the syntax of their input expressions? If so, they should profit from a more explicit representation of this structure. In order to test this, the SRN, GRU and LSTM models are now trained and tested on the `binary_fol` data without ignoring brackets. Additionally, the data are adapted by randomly omitting only certain bracket pairs. Thus, a modified version of the `binary_fol` data is created, with only 50% of all bracket pairs included. The models are also trained and tested on this half-bracketed data set. All results are shown in Table 5.3. The previously obtained scores for the data with no brackets are also included for easier reference. Figure 5.9 visualizes all results in a factor plot.

	0%		50%		100%	
	train	test	train	test	train	test
128/25/75 SRN	97.9	87.3	79.0	72.0	84.8	82.4
128/25/75 GRU	100.0	96.0	100.0	95.2	100.0	96.2
128/25/75 LSTM	100.0	94.6	100.0	90.6	99.3	90.6

Table 5.3: Training and testing accuracy scores of the three recurrent models on the FOL inference task for the `binary_fol` data, with percentage of included bracket pairs in train and test set indicated in the upper row. Results are averaged over five runs.

The GRU obtains the best average score for each data set, and seems least affected by the differing bracket percentages. There is no significant difference between its testing performance on the 100%, 50% and 0% data sets. Training accuracy after the 50th epoch is always 100%. The GRU does not profit from the syntactic cues, but it does not suffer from them either, unlike the SRN and the LSTM. Both these models deteriorate when only 50% of the bracket pairs are shown. Apparently the brackets do not seduce them to learn and use the underlying hierarchic structures - the partial bracketing just seems to confuse them. Especially the SRN cannot cope with the 50% data at all, as the high standard deviation in Figure 5.9 shows. One SRN only obtains a testing accuracy of 37.5%. Upon inclusion of all remaining brackets, performance improves significantly for the SRN, but not for the LSTM. LSTM training accuracy even decreases.

All in all, none of the considered models seems to profit from the bracketing. This either indicates that their strategy does not depend on a notion of syntactic hierarchy, or that they do not recognize the brackets as relevant signals in this regard.

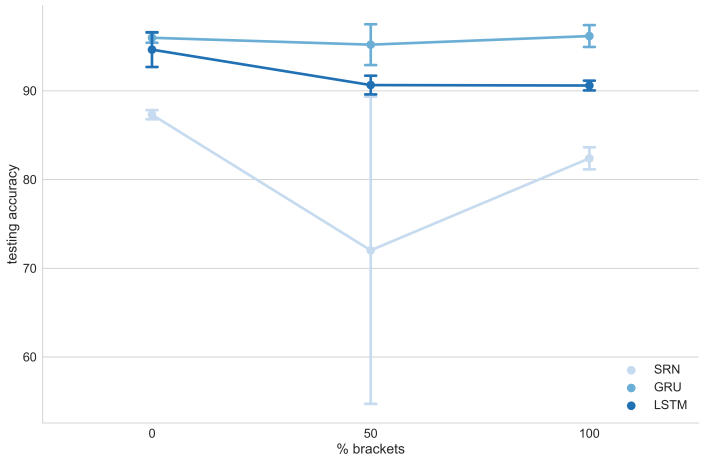


Figure 5.9: Factor plot visualizing the results of the bracket experiments reported in Table 5.3, together with the standard deviation per model per data set.

5.5.2 Unseen lengths

The next challenge that I consider is generalization to unseen lengths. Compositional agents use knowledge of primitive semantic components to infer the meaning of complex expressions. In theory, there is no limit to the length of interpretable sequences as long as all individual constituents have been encountered and internalized before. E.g., if someone knows the meaning of the words 'walk', 'stop' and 'and', then she knows the meaning of 'walk and stop', but also of 'walk and stop and walk', 'walk and stop and walk and stop' and all other grammatically admissible combinations of these items. Likewise, if she also knows the meaning of 'not', she does not need to witness all its possible instantiations to make sense of new combinations containing the word. The meanings of 'not walk' and 'not stop' are equally clear, given a correct understanding of the applied primitives.

Neural models are often considered incapable of such generalization, allegedly because they are limited to the training space. See Marcus 2003, Kaiser and Sutskever 2015, Reed and De Freitas 2015 or Evans and Grefenstette 2018. I want to test if this is the case for the recurrent models studied in this project. The language $\mathcal{L}_N$ licenses a heavily constrained set of grammatical configurations, but it does allow the sentence length to vary according to the number of included negations. A perfectly compositional model should be able to interpret statements containing any number of negations, on condition that it has seen an instantiation at least once at each position where this is allowed If this condition is not met, the training data provide statistical evidence against the negatability of particular sentence components. If a model is consequently confronted with items that do contain negations at these positions, its failure is due to an adequate fit rather than limited generalizability.

In a new experiment, the models are trained on pairs containing sentences with

a fixed maximum length, which is set to 7. The minimum length is 5, for sentences containing no negations. Thus, during training the models encounter negations at all possible positions, but never more than two in a single statement. After 50 epochs, they are tested on a set of pairs containing only sentences with the complementary lengths 8 and 9. These expressions contain three or four negations, which is something the models have not seen before, but if they learned to apply negation at all individual positions, compositionality should enable them to also accommodate the longer combinations in the test data. The train and test set are generated according to the procedure described in Chapter 4. They contain approximately 30,000 and 5,000 instances, respectively. Due to the division between sentence lengths they are necessarily mutually exclusive.

	train	test
128/25/75 SRN	98.9	47.9
128/25/75 GRU	100.0	51.9
128/25/75 LSTM	100.0	54.5

Table 5.4: Accuracy scores on the FOL inference task for models trained on pairs of sentences with lengths 5, 6 or 7 and tested on pairs of sentences with lengths 8 or 9. Results are averaged over five runs.

The results are shown in Table 5.4. The difference between mean training and testing performance is dramatic for all three recurrent models, with a drop of almost 50 percentage points on average. This shows that none of the considered architectures fully masters the compositional ability required to generalize from the sentences with at most two negations to those containing three or four. Let N denote the set of negatable sentence positions. During training, the models did not only encounter sentences with a single negation at position n for each $n \in N$, but also sentences with double instantiations at positions n, m for all $n, m \in N$ such that $n \neq m$. The testing results in Table 5.4 prove that this is not enough to reach a decent score on the test set. Nevertheless, it must be noted that the reported accuracies are well above chance level. The test data follow a distribution similar to the one visualized in Figure 4.4c, with the most prevalent label assigned to 25.8% of the instances. The fact that the models reach an accuracy of approximately 50% on this 7-class problem shows that they certainly outperform any classifier with a single or random output. This suggests that some limited form of compositional learning does take place.

It can be objected that the training regime of this experiment actually discourages models from generalizing to unseen sentence lengths, because the lengths that are observed form an uninterrupted succession on the natural number line. By only confronting a model with sentences containing 5, 6 or 7 words, the training data suggest that this is the range of valid sentence lengths. From this point of view, model failure for longer sentences need not indicate poor generalizing potential, but can also be due to appropriate specialization with respect to the statistical properties of the training data. To really test whether models can cope with new sentence lengths, they should at least be provided with a cue that such lengths exist. For this reason, a new experiment is conducted in

which the models are trained on pairs of sentences with length 5, 7 or 8, and tested on pairs of sentences with lengths 6 or 9. As before, the training and test sets contain some 30,000 and 5,000 sentence pairs, respectively. Results are shown in Table 5.5.

	train	test
128/25/75 SRN	98.2	28.8
128/25/75 GRU	100.0	90.4
128/25/75 LSTM	100.0	88.8

Table 5.5: Accuracy scores on the FOL inference task for models trained on pairs of sentences with lengths 5, 7 or 8 and tested on pairs of sentences with lengths 6 or 9. Results are averaged over five runs.

As before, all recurrent models obtain (near-)perfect training accuracy scores. What happens on the test set is interesting. It turns out that the GRU and LSTM can generalize from lengths 5, 7 and 8 to 6 and 9 very well, while the SRN has even more difficulty with this task than with the previous one. Apparently, the training corpus of lengths 5, 6 and 7 did bias the models to specialize on a specific range of lengths. The new data suggest that other lengths are possible by skipping length 6 during training, thereby enabling the GRU and LSTM to generalize to unseen sentence lengths 6 and 9. From witnessing expressions with no, two or three negations, the models extrapolate to sentences with one or four negations. It appears that the GRU and LSTM gates play a crucial role, because the results show that the SRN does not have this capacity at all.

5.5.3 One-shot learning

Typical to compositional reasoning is that it enables agents to perform one-shot learning: correct classification of examples that have not been observed more than a handful of times before. Provided that atomic constituents and production rules are understood, compositionality does not require that abundantly many instances embodying a semantic category are observed. In a new experiment, I assess if this property applies to the GRU, the recurrent unit that performed best in all preceding experiments. In particular, the aim is to determine whether the model generalizes from entailment relations between sentences containing words observed during training to sentences of the same structure with words that were never seen before, but whose lexical semantics resemble those of known vocabulary items.

The current set-up cannot deal with unknown words, so the training regime is modified. Instead of randomly initializing an embedding matrix $\mathbf{M}_{emb}$ that is updated by backpropagation, pretrained word embeddings are used that were generated by the GloVe algorithm from the combined Wikipedia and Gigaword corpus (see Section 2.1.2 and Pennington, Socher, and Manning 2014). The 50-dimensional GloVe representations of vocabulary words are stored and kept constant. After training, model behavior with unseen words can be tested by simply adding the corresponding GloVe vector to the embedding matrix. All other parameters are treated as before.

On the unbracketed `binary_fol` data, the GRU that uses 50-dimensional GloVe embeddings, 128 hidden units and a 75-dimensional comparison layer obtains a mean training accuracy of 100.0% and a testing accuracy of 95.9% (averaged over five runs). The best GRU achieves 100.0% training and 97.1% testing accuracy. This performance is on a par with earlier experimental results, for which the word embeddings were trained alongside the regular network parameters. Because no optimization is applied, the pre-trained embeddings prove to be well suited for this task as they are. This indicates that GloVe somehow encodes the lexical entailment relations that are required to perform logical classification at sentence level, as already established for word2vec by Henderson and Popa 2016.

In the following experiments, the best-performing GRU is provided with knowledge about new individual words in the form of their GloVe embeddings. Together with the information it acquired about sentence structure during training, this should be enough for a compositional model to accommodate certain types of new sentence pairs. Three such cases are investigated below.

Synonyms One of the most basic relations on the level of lexical semantics is synonymy, which holds between words with equivalent meanings. The `binary_fol` data contain no hyperintensionality or propositional attitudes, so a word can be substituted with one of its synonyms without altering the entailment relation assigned to the sentence pairs. If the GRU manages to perform well on such a modified data set after receiving the pretrained word embedding of the unseen word, this is a first piece of evidence for its one-shot learning skills. I test this for several pairs of synonymous words. The best-performing GRU is first evaluated with respect to the fragment of the test data containing the original word w, and consequently with respect to that same fragment after replacing the original word with its synonym $s(w)$. The pairs of words, the cosine distance $cos\_dist(w, s(w))$ between their GloVe embeddings and the obtained results are listed in Table 5.6.

			accuracy on `binary_fol` test data	
w	$s(w)$	$cos\_dist(w, s(w))$	containing w	containing $s(w)$
children	kids	0.21	98.3	91.9
love	adore	0.57	97.0	92.5
fear	dread	0.39	97.2	91.3
hate	detest	0.56	97.4	48.2

Table 5.6: Effect on best-performing GRU of replacing words w by unseen synonyms $s(w)$ in the `binary_fol` test set and providing the model with the corresponding GloVe embedding.

For the first three examples in Table 5.6, substitution only decreases testing accuracy by a few percentage points. Apparently, the word embeddings of the synonyms encode the lexical properties that the GRU needs to recognize that the same entailment relations apply to the sentence pairs. This does not prove that the model has distilled essential information about hyponymy from the GloVe embeddings. It could also be that the word

embeddings of the replacement words are geometrically very similar to the originals, so that it is an algebraic necessity that the same results arise. However, this suspicion is inconsistent with the result of changing 'hate' into 'detest'. The cosine distance between these words is 0.56, so according to this measure their vectors are more similar than those representing 'love' and 'adore' (which have a cosine distance of 0.57). Nonetheless, replacing 'hate' with 'detest' strongly confounds the model, whereas substitution of 'love' into 'adore' only decreases testing accuracy by 4.5 percentage points. This illustrates that robustness of the GRU in this respect is not a matter of simple vector similarity. In those cases where substitution into synonyms does not confuse the model it must have recognized a non-trivial property of the new word embedding that licenses particular inferences.

Ontological twins Relative to the ontology of language $\mathcal{L}_N$, certain pairs of concepts fill an identical semantic 'niche'. This is to say that there are words whose extensions are not necessarily equivalent, but that can be substituted with each other without affecting the entailment relation between any pair of sentences in which they feature. This trivially applies to self-identical terms or synonyms, but in the strictly defined hierarchy of $\mathcal{L}_N$ it is also the case for pairs of terms w, v that maintain the same lexical entailment relations to all other terms in the taxonomy. I call such terms 'ontological twins'. Technically, if $\odot$ is an arbitrary lexical entailment relation and $\mathcal{O}$ is an ontology, then w and v are ontological twins if and only if $w, v \in \mathcal{O}$ and for all $u \in \mathcal{O}$, if $u \notin \{w, v\}$ then $w \odot u \Leftrightarrow v \odot u$.

Examples of ontological twins in the taxonomy of nouns $\mathcal{N}^{\mathcal{L}_N}$ are 'Romans' and 'Venetians' . This can easily be verified in the Venn diagram of Figure 4.2 by replacing 'Romans' with 'Venetians' and observing that the same hierarchy applies. The same holds for e.g. 'Germans' and 'Polish' or for 'children' and 'students'. For several such word-twin pairs the GRU is evaluated with respect to the fragment of the test data containing the original word w, and with respect to that same fragment after replacing the original word with ontological twin $t(w)$. Results are shown in Table 5.7.

The examples in Table 5.7 suggest that the best-performing GRU is largely robust with respect to substitution into ontological twins. Replacing 'Romans' with other urban Italian demonyms hardly affects model accuracy on the modified fragment of the test data. As before, there appears to be no direct correlation with vector similarity because the cosine distance between the different twin pairs has a much higher variation than the corresponding accuracy scores. 'Germans' can be changed into 'Polish' without significant deterioration, but substitution with 'Dutch' greatly decreases testing accuracy. The situation is even worse for 'Spanish'. Again, cosine similarity provides no explanation - 'Spanish' is still closer to 'Germans' than 'Neapolitans' to 'Romans'. Rather, the accuracy appears to be negatively correlated with the geographical distance between the national demonyms. After replacing 'children' with 'students', 'women' or 'linguists', testing scores are still decent.

Alternative hierarchies So far, I replaced individual words in order to assess whether the GRU can generalize from the vocabulary to new notions that have comparable se-

			accuracy on `binary_fol` test data	
w	$t(w)$	$cos\_dist(w, t(w))$	containing w	containing $t(w)$
Romans	Venetians	0.28	97.3	97.3
	Milanese	0.72		95.4
	Neapolitans	0.57		95.4
Germans	Polish	0.37	96.8	96.3
	Dutch	0.40		78.3
	Spanish	0.50		62.9
children	students	0.27	98.3	94.6
	women	0.24		91.8
	linguists	0.92		86.3

Table 5.7: Effect on best-performing GRU of replacing words w by unseen ontological twins $t(w)$ in the `binary_fol` test set and providing the model with the corresponding GloVe embedding.

mantics in the context of this entailment recognition task. The examples have illustrated that the model tends to do this quite well. In the last one-shot learning experiment, I do not replace single words, but sets of nouns, thereby also studying the flexibility of internalized relational semantics. The substitute noun sets generalize the understanding of ontological twins, in the sense that they must not alter the hierarchic structure of the original noun ontology $\mathcal{N}^{\mathcal{L}_N}$. This is to maintain correct entailment labels. Formally, the replacement can be regarded as a function r, mapping words w to substitutes $r(w)$. Not all items have to be replaced. For an ontology $\mathcal{O}$, the function r must be such that for any $w, v \in \mathcal{O}$ and lexical entailment relation $\odot$, $w \odot v \Leftrightarrow r(w) \odot r(v)$. The result of applying r can be called an 'alternative hierarchy'.

An example of an alternative hierarchy is the result of the replacement function r_1 that maps 'Romans' to 'Parisians' and 'Italians' to 'French'. Performing this substitution in the Venn diagram of Figure 4.2 shows that the taxonomy remains structurally intact. The best-performing GRU is evaluated on the fragment of the `binary_fol` test data containing 'Romans' or 'Italians', and consequently on the same fragment after implementing replacement r_1 and providing the model with the GloVe embeddings of the unseen words. Replacement r_1 is incrementally modified up until replacement r_4, which substitutes all nouns in $\mathcal{N}^{\mathcal{L}_N}$. The results of applying r_1 to r_4 are shown in Table 5.8.

The results are positive: the GRU obtains 86.7% accuracy even after applying r_4, which substitutes the entire ontology $\mathcal{N}^{\mathcal{L}_N}$ so that no previously encountered nouns are present in the test set anymore. Testing scores are above 87% for the intermediate substitutions r_1 to r_3. This outcome clearly shows that the classifier does not depend on a strongly customized word vector distribution in order to recognize higher-level entailment relations. Even if all nouns are replaced by alternatives with embeddings that have not been witnessed or optimized beforehand, the model obtains a high testing accuracy. This establishes obvious compositional capacities, because familiarity with structure and information about lexical semantics in the form of word embeddings are enough for the

						accuracy on `binary_fol` test data	
	Romans	Italians	Germans	Europeans	children	before substitution	after substitution
r_1	Parisians	French	⋮	⋮	⋮	97.2	93.5
r_2	⋮	⋮	Polish	⋮	⋮	97.1	93.9
r_3	⋮	⋮	⋮	Eurasians	⋮	97.1	87.6
r_4	⋮	⋮	⋮	⋮	students	97.1	86.7

Table 5.8: Effect on best-performing GRU of replacing noun ontology $\mathcal{N}^{\mathcal{L}_N}$ with alternative hierarchies as per the replacement functions r_1 to r_4. Vertical dots indicate that cell entries do not change on the next row.

model to accommodate configurations of unseen words.

These findings must be relativized by noting that the functions r_1 to r_4 map nouns in $\mathcal{N}^{\mathcal{L}_N}$ to alternatives that are thematically quite similar to the original taxonomy. What happens upon considering ontologies with the same structure, but with an altogether different subject matter? Three such alternative hierarchies are considered: $r_{animals}$, $r_{religion}$ and $r_{America}$. Each of these functions relocalizes the noun ontology in a totally different domain of discourse, as indicated by their names. Table 5.9 specifies the functions and their effect.

	Romans	Italians	Germans	Europeans	children	accuracy
$r_{animals}$	rabbits	rodents	cats	mammals	pets	59.0
$r_{religion}$	calvinists	protestants	catholics	christians	orthodox	56.3
$r_{America}$	Clevelanders	Ohioans	Californians	Americans	women	58.2

Table 5.9: Effect on best-performing GRU of replacing noun ontology $\mathcal{N}^{\mathcal{L}_N}$ with alternative hierarchies as per the replacement functions $r_{animals}$, $r_{religion}$ and $r_{America}$. Accuracy is measured on the `binary_fol` test set after applying the respective replacement functions.

Testing accuracy decreases drastically, which indicates that the model is sensitive to the changing topic. Variation between the scores obtained after the three transformations is limited. Although they are much lower than before, they are still far above chance level for a seven-class problem. This suggests that the model is not at a complete loss as to the alternative noun hierarchies. Possibly, including a few relevant instances during training could already improve the results.

The experiments performed in the last few sections suggest that the GRU and LSTM architectures apply at least basic forms of compositional learning. Especially the results of the one-shot learning experiments are difficult to explain if no such mechanism is at

work within the GRU. It remains to be investigated exactly how the internal representations of the GRU achieve this. Many other experiments can be designed to test different aspects of the compositionality of the recurrent models. Appendix C describes one such experiment, which assesses the extent to which the GRU exploits logical structures in the training data.

Chapter 6

Conclusion

6.1 Summary

In this thesis I studied the recognition of entailment, understood as a semantic issue with a linguistic appearance and a logical basis. To this end, I used the research of Bowman, Potts, and Manning 2015 as a starting point, because it is the only major project of recent years that challenged semantic models with (semi-)natural data generated by explicit inference rules. Other studies have tended to focus on fully natural or fully formal data, lacking either a clear theory of meaning or applicability to natural language contexts. I reimplemented Bowman's tree-shaped matrix and tensor models, as well as a bag-of-words baseline, and replicated his experiments using the original data sets. These were produced by the NL calculus of MacCartney 2009 and the artificial language $\mathcal{L}_B$, with information about syntactic structure provided by brackets. The obtained results are comparable to those reported by Bowman.

A number of issues with Bowman's research came to light. I identified some general problems with NL, as well as issues issues concerning the specific data sets used in the earlier research. To overcome these issues, I proposed a new classification task: First-Order Entailment Recognition. NL was replaced by FOL as underlying logic, and a first-order theorem prover and model builder were adopted to generate new data sets. I translated Bowman's data into FOL in order to deduce the first-order entailment relation between the premise-hypothesis sentence pairs, yielding the `f1_fol` data. To avoid class disbalance, the new data set `unary_balanced_fol` was generated as well. On both of these sets, the tree-shaped models obtained high accuracy scores.

To increase the difficulty of the task, I abandoned Bowman's language $\mathcal{L}_B$ in favor of the new language $\mathcal{L}_N$, with a more complex grammar and taxonomy of terms, longer sentences, binary predicates and varying word order. The new data set `binary_fol` was produced using $\mathcal{L}_N$ with FOL as background logic. It turned out that the recursive networks are able to generalize from the new training data, while the summing baseline failed dramatically because it does not pay attention to word order. Detailed analysis of the best-performing tensor model showed that the error pattern has a logical explanation. I observed some semantic clustering of word embeddings, which was consistent across

model runs. Projections of the different composition stages demonstrated that the tensor model localizes sentences and their constituents in different subspaces. I showed that negation effectuates a mirroring effect on sentence level, and discovered that sentence vectors cluster according to their verb and their second (object) quantifier.

Next, I discarded the recursive models and shifted attention to a recurrent architecture in order to investigate if the same entailment classification task can be handled sequentially, without cues about syntactic structure. Three different recurrent units were included: SRN, GRU and LSTM. All of these set-ups successfully faced the task, with the GRU and LSTM outperforming the tensor model. This suggests that compositional distributional models can learn to apply first-order semantics, even if they process the input linearly and are not provided with background knowledge about sentence structure. Qualitative analysis of sentence vectors of the best-performing GRU revealed that negation mirrors representations with respect to a hyperplane, while clear clusters are observable for verbs, second quantifiers and verb-second quantifier combinations. I used diagnostic classifiers to perform quantitative analysis, and demonstrated that the best-perorming GRU encodes information about recursive depth, word index, semantic type and scope of negation, but not about the monotonicity direction of the words it processes.

I conducted additional experiments to test whether the success of the recurrent models can be attributed to compositional learning. By using partially bracketed data sets, I studied the models' inclination to profit from syntactic information, but on average this decreased testing accuracy. Next, I focused on generalization to unseen lengths and found that the GRU and LSTM successfully extrapolate from training data containing sentences with 5, 7 or 8 words to test data that only contain sentences of lengths 6 or 9. Finally, I assessed the GRU in terms of its one-shot learning skills. After training with fixed GloVe embeddings, the model proved capable of accommodating not only new words, but also sets of unseen nouns if the corresponding GloVe embeddings were supplied.

6.2 Contributions

All in all, the main contributions of the thesis can be summarized by the following points:

- Critical assessment of research in the field of entailment recognition, focusing on the influential work of Bowman, Potts, and Manning 2015.
- Description of the First-Order Entailment Recognition Task, together with an automated data generation protocol that labels sentence pairs in (semi-)natural language according to their FOL entailment relation.
- Motivation and specification of language $\mathcal{L}_N$ to produce several FOL-labelled corpora of premise-hypothesis pairs, the most important of which is the `binary_fol` data set.
- Extensive qualitative analysis of the performance of tree-shaped neural networks on the First-Order Entailment Recognition Task.

- Implementation of a recurrent neural architecture to address the same classification task sequentially, with SRN, GRU and LSTM units.
- Qualitative interpretation of the best-performing GRU on the level of word embeddings, hidden units and sentence vectors.
- Quantitative interpretation by means of diagnostic classification in order to test the GRU's awareness of recursive depth, word index, semantic type, monotonicity direction and scope of negation.
- Several experiments addressing compositional generalization, in particular the ones concerning unseen lengths and one-shot learning, the results of which suggest that recurrent models possess at least rudimentary compositional skills.

6.3 Future work

In addition to these conclusions, the project has given rise to a number of follow-up questions. The most important next step would be to complicate the data sets. Longer sentences could be included, connectives such as disjunction, conjunction or implication, predicates with higher arity, function symbols, lambda expressions etc. Instead of a single premise for each hypothesis, multiple premises could be used. One could extend the vocabulary, replace FOL with a more expressive logic, or pay more attention to the link between artificial data sets such as `binary_fol` and SICK or SNLI, to see if the implemented models also function on wholly natural corpora.

As for the models that were used, it is most interesting to further pursue the recurrent approach. More sophisticated architectures could be investigated, containing e.g. an attention mechanism. With supplementary computational resources, better hyperparameter tuning could be performed to realize the classifiers' full potential. More tests and experiments could be designed to interpret the models, their internal representations and obtained results. In particular, the suspected compositional learning skills of the recurrent models deserve more attention. In the lights of the evidence that GRUs apply compositional generalization on the level of logic, sentence length and unseen words, it is worth assessing if these observations apply to other contexts and how they could impact training regimes. At the end of this thesis, one entailment is easily recognized: a lot remains to be done.

Appendix A

Trace of Prover9 and Mace4

Suppose that we have the following set of axioms A:

$$\begin{aligned} A = \{&\forall x(German(x) \rightarrow European(x)),\\ &\forall x(Roman(x) \rightarrow Italian(x)),\\ &\forall x \forall y(love(x,y) \rightarrow like(x,y)),\\ &\exists x German(x),\\ &\exists x Roman(x)\} \end{aligned}$$

Let premise φ and hypothesis ψ be the following formulas:

$$\begin{aligned} \varphi &:= \forall x(German(x) \rightarrow \forall y(Roman(y) \rightarrow love(x,y)))\\ \psi &:= \exists x(European(x) \wedge \exists y(Italian(y) \wedge like(x,y))) \end{aligned}$$

After formatting the formulas in $A \cup \{\varphi, \psi\}$ according to the Prover9 and Mace4 conventions, computation can start to determine whether the axioms A together with premise φ entail hypothesis ψ. The most relevant fragments from the produced traces are shown below:

```
formulas(assumptions).
    all x (German(x) -> all y (Roman(y) -> love(x,y))).
    all x (German(x) -> European(x)).
    all x (Roman(x) -> Italian(x)).
    all x all y (love(x,y) -> like(x,y)).
    exists x German(x).
    exists x Roman(x).
end_of_list.

formulas(goals).
    exists x (European(x) & exists y (Italian(y) & like(x,y))).
end_of_list.
```

```
============================== Prover9 ===============================

============================== PROOF =================================
Length of proof is 22.
Level of proof is 5
1 (all x (German(x) -> (all y (Roman(y) -> love(x,y)))))
    # label(non_clause).  [assumption].
2 (all x (German(x) -> European(x))) # label(non_clause).  [assumption].
3 (all x (Roman(x) -> Italian(x))) # label(non_clause).  [assumption].
4 (all x all y (love(x,y) -> like(x,y))) # label(non_clause).  [assumption].
5 (exists x German(x)) # label(non_clause).  [assumption].
6 (exists x Roman(x)) # label(non_clause).  [assumption].
7 (exists x (European(x) & (exists y (Italian(y) & like(x,y)))))
    # label(non_clause) # label(goal).  [goal].
8 German(c1).  [clausify(5)].
9 -German(x) | -Roman(y) | love(x,y).  [clausify(1)].
10 -German(x) | European(x).  [clausify(2)].
11 Roman(c2).  [clausify(6)].
12 -Roman(x) | Italian(x).  [clausify(3)].
13 -Roman(x) | love(c1,x).  [resolve(8,a,9,a)].
14 love(c1,c2).  [resolve(13,a,11,a)].
15 -love(x,y) | like(x,y).  [clausify(4)].
16 European(c1).  [resolve(8,a,10,a)].
17 -European(x) | -Italian(y) | -like(x,y).  [deny(7)].
18 -Italian(x) | -like(c1,x).  [resolve(16,a,17,a)].
19 Italian(c2).  [resolve(11,a,12,a)].
20 -like(c1,c2).  [resolve(18,a,19,a)].
21 like(c1,c2).  [resolve(14,a,15,a)].
22 $F.  [resolve(20,a,21,a)].
============================== end of proof ==========================

============================== Mace4 =================================

============================== DOMAIN SIZE 2 =========================
=== Mace4 starting on domain size 2. ===
============================== STATISTICS ============================
For domain size 2.
Current CPU time: 0.00 seconds (total CPU time: 0.00 seconds).
Ground clauses: seen=18, kept=18.
Selections=2, assignments=3, propagations=10, current_models=0.
Rewrite_terms=3, rewrite_bools=21, indexes=3.
Rules_from_neg_clauses=0, cross_offs=0.
============================== end of statistics =====================
```

```
============================== DOMAIN SIZE 3 =========================
=== Mace4 starting on domain size 3. ===
============================== STATISTICS ============================
For domain size 3.
Current CPU time: 0.00 seconds (total CPU time: 0.01 seconds).
Ground clauses: seen=35, kept=35.
Selections=2, assignments=3, propagations=10, current_models=0.
Rewrite_terms=3, rewrite_bools=25, indexes=3.
Rules_from_neg_clauses=0, cross_offs=0.
============================== end of statistics =====================
============================== DOMAIN SIZE 4 =========================
=== Mace4 starting on domain size 4. ===
============================== STATISTICS ============================
For domain size 4.
Current CPU time: 0.00 seconds (total CPU time: 0.01 seconds).
Ground clauses: seen=58, kept=58.
Selections=2, assignments=3, propagations=10, current_models=0.
Rewrite_terms=3, rewrite_bools=29, indexes=3.
Rules_from_neg_clauses=0, cross_offs=0.
============================== end of statistics =====================
```

See Section 2.2.2 for a short explanation of the rules used in the proof.

Appendix B

Objections to Bowman e.a.

This appendix provides a more detailed specification of all nine concerns about the research of Bowman, Potts, and Manning 2015 summarized in Section 3.3.

B.1 Natural logic

The first issue actually encompasses a set of problems, all of which are caused by the adopted calculus of natural logic. As explained in Section 2.2.1, the main advantage of this system is that it allows for semantic derivations without requiring formalization to an abstract 'language of thought'. Instead, inference rules are applied directly to natural language expressions. Irrespectively of this system's plausibility from a cognitive or linguistic perspective, its simplicity certainly makes it an appealing option from an engineering point of view. No translations to abstract representations are needed to perform inferences, so that large sets of artificial data can easily be generated. However, the intuitive convenience of a logic operating directly on natural language expressions comes at the expense of other properties. The most important such shortcomings are listed below.

- The natural logic used by Bowman is not as natural as it seems, or at least not as natural as it is intended to be. Natural logic owes its name to the fact that it applies directly to natural language. It is questionable how natural the language $\mathcal{L}_B$ really is. Indeed, it is admitted that the 'data consist of pairs of sentences generated from a grammar for a simple English-like *artificial* language' (Bowman, Potts, and Manning 2015, 6), whereas the aim of the research is to 'introduce a novel NN architecture for *natural* language inference' (idem, 1). Although sentences such as '((all warthogs) (not walk))' are immediately comprehensible, they do not comply with English syntax. This is not only due to the inclusion of the parse tree by means of brackets, but also to the omission of auxiliary verb 'to do'. The result is a data set of sentence pairs in an unnatural language, whose semantic relations are inferred with a logic specifically designed to operate on a natural one. Consistency in this respect would be preferable, either by applying natural logic to

fully natural expressions, or by abandoning all naturality restrictions in favour of some well-understood formal system.

- Bowman uses the version of natural logic proposed by MacCartney and Manning 2009. This system comes with its own calculus for determining the entailment relation between a premise p and a conclusion or hypothesis h. See Section 2.2.1 for a more detailed description. The core of this method is finding an appropriate edit sequence that connects p to h. In MacCartney 2009, MacCartney himself already recognizes that this approach faces several important limitations. Particularly problematic is the fact that several edit sequences are usually possible. Sometimes they lead to divergent results, which introduces a troublesome degree of uncertainty.

- Another concern about the adopted natural logic calculus is that 'the usefulness of the result is sometimes limited by the tendency of the join operation toward less informative entailment relations' (MacCartney 2009, 106). This is to say that the calculus joins atomic entailment relations in a way that often yields unions of relations, which are weak or even meaningless. Such cases are labelled 'unknown'. As a consequence they are excluded from Bowman's data, notwithstanding the possible existence of a straightforward semantic relation between the constituent sentences, which was just not captured by the calculus.

- MacCartney also concedes that 'the inference method provides no general mechanism for combining information from multiple premises' (MacCartney 2009, 106). This is no problem for the experiments described so far, because they only deal with a single premise and hypothesis p and h. It is a problem if we wish the experiments to be somewhat scalable. Clearly, the aim is not to fully understand the dynamics between an artificial miniature language and a class of neural networks, but to study the logical capacities of such models as generally as possible. The fact that the highest degree of generality may not be realistic at present does not justify the exclusive use of a calculus that fundamentally impedes this ambition. It must be taken into account that logic involves reasoning with multiple premises, as is reflected by the classical inference rules (e.g. modus ponens, modus tollens, disjunction elimination). Therefore, a system capable of modelling this prevalent phenomenon should be strongly preferred to one that has no such means.

- A major issue is that the natural logic calculus is provably incomplete: there exist valid inferences for which the method is not able to produce the desired entailment relation. Some of these inferences are rudimentary, everyday deductions that a natural logic, if anything, should be able to explain. De Morgan's laws are an important example (see Repplinger 2017). The first-order representation of their quantified forms is as follows:

  (a) $\neg(\forall x P(x)) \Leftrightarrow \exists x(\neg P(x))$
  (b) $\neg(\exists x P(x)) \Leftrightarrow \forall x(\neg P(x))$

The formulas of (a) and (b) are illustrated by the following instantiations in $\mathcal{L}_B$:

(c) `( ( not_all warthogs ) walk )` $\Leftrightarrow$ `( ( some warthogs ) (not walk ) )`
(d) `( ( no warthogs ) move )` $\Leftrightarrow$ `( ( all warthogs ) ( not move ) )`

Such equivalences are almost trivial, so it is reasonable to expect the logic governing the data generation process to accommodate them. Nevertheless, the adopted calculus fails to do so. MacCartney describes several attempts to derive the desired relation for De Morgan sentences, but none are successful. At best, a less informative relation is derived, but equivalence can never be inferred. The negative conclusion that inference rules as basic as De Morgan's laws apparently cannot be handled by the current system is strong evidence that a different logic is required for the task. (Intuitionistic or other non-classical arguments against the plausibility of (a) - (d) may be defensible from a purely theoretical viewpoint, but cannot be expected to outweigh the instinctive reasoning patterns ingrained in the natural language use that we eventually hope to model.) It is hard to interpret Bowman's results, knowing that they were obtained with 'natural' data produced by a system effectively violating some of our most basic deductive principles.

- Also, the natural logic calculus is not provably sound. In this context, soundness must be understood as the guarantee that informative results of the inference method are also correct. Pursuing a proof of soundness would contradict the purpose of natural logic, which is to provide a model of informal reasoning. This means that, on the one hand, the formal apparatus required to lend decisive, technical support to a soundness claim is simply not available. On the other hand, there are no known counterexamples to the system's possible soundness, so it cannot be concluded that it is unsound either.

- Finally, according to van Benthem 2007, natural logic is weaker than first-order predicate logic because 'it only describes a part of all possible quantifier-based inferences'. This is to say that there are fragments of predicate logic that are not expressible in purely natural terms. In particular, compositional structure proves to be a problem for natural logic, if it is really understood as a system that functions without the intervention of any formalism whatsoever. Consider e.g. the sentence 'Some warthog is not fat and fast'. Without any specification of compositional structure, natural logic is at a loss. For the sentence in its current form, two different interpretations are possible: the negation can modify only 'fat' (a), or the conjunction 'fat and fast' (b). This point can be clarified by providing the first-order logic translation for both senses:

  (a) $\exists x(warthog(x) \wedge \neg fat(x) \wedge fast(x))$
  (b) $\exists x(warthog(x) \wedge \neg(fat(x) \wedge fast(x)))$

  The first-order renderings of (a) and (b) indicate precisely the scope of the negation. Due to structural ambiguity, this scope cannot be taken at face value, which is

exactly what natural logic aims to do. This example illustrates that natural logic is severely disadvantaged by the ubiquitous structural ambiguity of natural language, unless additional machinery is allowed. This is the case in Bowman's data, where parse trees are explicitly provided by means of bracketed expressions. It could be argued that the use of brackets is unnatural, and thus defeats the purpose of adopting a natural logic. Their inclusion is even more puzzling if one takes into consideration that ambiguity does not occur for the simple syntax and exclusively unary predicates of $\mathcal{L}_B$. However, because we wish to study compositionality more generally than only with respect to a toy language, this issue must be acknowledged.

B.2 Implementation and data anomalies

Even if we assume that all aforementioned, general weaknesses of MacCartney's calculus can be overcome, issues concerning the particular data generated by Bowman remain:[1]

- The implemented system is only a simplified version of MacCartney's. Fundamental lexical entailment relations are included for the nouns and verbs in the adopted taxonomies (see Figures 3.1 and 3.2), as well as projectivity matrices for negation and the relevant quantifiers. This is all done in accordance with the method presented in MacCartney 2009. The more sophisticated elements of this system, however, have not been implemented. This includes the semantic composition rules for all logical connectives other than negation, as well as monotonicity and projectivity properties of specific classes of verbs such as implicatives and factives. Yet, as the language $\mathcal{L}_B$ has no other connectives than negation, and no other verbs than the four in class $\mathcal{V}^{\mathcal{L}_B}$, none of which are factive or implicative, these simplifications are justifiable. It is mostly with an eye on scalability and generalizability that they can be considered problematic.

- More importantly, manual inspection shows that the data sets used by Bowman contain a great number of sentence pairs with incorrect labels. Testing the code that was used to produce the data with new expressions confirmed the suspicion that the generation process contains errors. A few examples are listed below, together with the entailment relation that Bowman's implementation of the natural logic calculus assigns to the respective sentence pairs:

1. `#  ( ( all turtles ) walk )  ( ( all turtles ) ( not move ) )`
2. `>  ( ( all warthogs ) move )  ( ( not_all ( not warthogs ) ) swim )`
3. `<  ( ( some ( not warthogs ) ) ( not walk ) )  ( ( some turtles ) ( not walk ) )`
4. `<  ( ( all turtles ) swim )  ( ( all ( not warthogs ) ) swim )`
5. `|  ( ( all ( not pets ) ) ( not walk ) )  ( ( some pets ) ( not walk ) )`

[1] All code and data used by Bowman are available on www.github.com/sleepinyourhat/vector-entailment. The data generation script to which this section refers can be found in the file quantifiers/quantgen.py.

In all of the cases (a) - (e), the assigned relation seems incorrect, or at least undesirable if there is anything natural about natural logic. Example (a) shows a sentence pair that is labelled with the independence relation. This can only be the case if none of the other relations of Table 2.1 is derivable. However, the premise clearly contradicts the conclusion. Walking implies moving, so if all turtles walk it cannot be the case that all turtles do not move and vice versa. The sentences are not complementary, so there should be an alternation between them. Instance (b) is labelled with backward entailment, which means that premise '((not_all (not warthogs)) swim)' should entail conclusion '((all warthogs) move)'. This is clearly not the case: if some turtle X doesn't swim, the premise is validated. This does not imply that all warthogs must move, so the conclusion can still be false. For (c), Bowman's code outputs forward entailment, which means that '((some (not warthogs)) (not walk))' implies '((some turtles) (not walk))'. This is also false. The class 'turtles' is contained by 'not warthogs', and not the other way around, so the monotonicity calculus should yield a backward entailment instead. The same reasoning holds for (d). Finally, for (e), an alternation is assigned. This is to say that '((all (not pets)) ((not walk))' and '((some pets) (not walk))' are mutually exclusive, but that their disjunction is not a tautology. This is unexpected because both sentences can be true at the same time. Consider the scenario where some (but not all) pets are the only animals that walk. Then it is both the case that all non-pets do not walk, and that some pets do not walk, so the premise and the conclusion are simultaneously true.

The instances (a) - (e) are only five of the numerous anomalies appearing in Bowman's data sets, and as a result of further experimentation with the code that was used to generate them. These results are not uninformative, but demonstrably incorrect. This means that either MacCartney's inference method is unsound, or Bowman's implementation is erroneous. As noted before, no proof of soundness is available, so neither of these options can be excluded at once. However, it seems unlikely that MacCartney is to blame, because the method described in Section 2.2.1 can be used to infer the correct relation for at least some of the examples (a)-(e). Consider instance (a). The edit sequence shown in Table B.1 outputs the correct entailment relation.

i	e_i	$x_i = e_i(x_{i-1})$	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
0		((all turtles) walk)			
1	SUB(walk,move)	((all turtles) move)	$\sqsubset$	$\sqsubset$	$\sqsubset$
2	INS(not)	((all turtles) (not move))	$\wedge$	$\wedge$	|

Table B.1: Analysis of a natural logic inference using the calculus of MacCartney 2009. See Section 2.2.1 for an explanation of method and notation.

In only two edit steps, the hypothesis can be transformed into the conclusion, with alternation as unique output. This is the entailment relation that we hoped to find, but

which was not inferred by Bowman's script. Similar analyses can be provided for the other examples. Even if the correct relation cannot be inferred, MacCartney's method never seems to yield wrong results. In the worst case they are uninformative, which means that a union of relations is returned (interpreted as 'unknown').[2]

B.3 Simplistic syntax

Another issue concerns the nature of the artificial language $\mathcal{L}_B$ that is used to generate the data. The grammar (see Table 3.1) is highly restricted. Although it is a sensible decision to study a constrained language in detail before attempting to handle higher degrees of complexity, there is a limit to the amount of simplification that can be permitted without drifting too far away from the ultimate objective of modelling natural language expressions. It is disputable whether Bowman exceeds this limit. Even if we just consider the syntax of $\mathcal{L}_B$, there are reasons to believe that he does.

Sentence structure is rigidly defined to always follow the order 'quantifier adverb noun adverb verb', which means that the maximum length is only five words (not counting brackets). Many of the included sentences have one or no negations, so that they are even shorter: no more than three or four words. These limitations on sentence length and syntactic variation make the language $\mathcal{L}_B$ even more artificial. Also, the different sentence constituents are taken from mutually exclusive vocabulary classes ($\mathcal{Q}^{\mathcal{L}_B}$, $\mathcal{N}^{\mathcal{L}_B}$, $\mathcal{V}^{\mathcal{L}_B}$ and $\mathcal{A}^{\mathcal{L}_B}$, see Section 3.1.1). This is a result of the rudimentary grammar, which allows for no (indirect) objects or other options to use the same word twice in one phrase. As a consequence, given a group of words constituting a sentence with unknown order, the only factor causing uncertainty about their correct placement is the position of a single negation. This is hardly ever the case with natural language expressions, whose constituents can usually be placed in several orders to produce different, but equally admissible statements. Lastly, the absence of grammatical objects is explained by the exclusive use of unary predicates, in the form of the intransitive

[2]These findings strongly suggest that the problem lies in the implementation. In the lights of all additional objections, no conclusive debugging on Bowman's code was performed, but it was examined thoroughly enough to identify the most probable cause of the anomalous results. The matrices containing the compositionality and projectivity relations are all identical to the ones specified by MacCartney. This is not the case for the interpretation function, which seems to be only loosely based on the latter's algorithm. Bowman's semantic interpretation function has only one way of guiding the edit sequence. The recursive structure is mechanically unfolded in the same, left-branching way for all sentence pairs, thus determining the order of the join operations, the application of the projectivity matrices, and thereby the eventual output. This accounts for the large number of unknowns for sentence pairs that might have been labelled with an informative relation if another edit sequence had been pursued. It does not explain the fact that incorrect relations are sometimes derived. This appears to be a result of the recursive structure of Bowman's interpretation function. MacCartney's algorithm, described in Section 2.2.1, requires that for each edit e_i, the atomic entailment relation be determined by projecting the lexical entailment relation upward through the semantic composition tree of the *entire* preceding expression. Bowman collapses the parse tree leaf by leaf and branch by branch, thereby inferring the atomic entailment relation with respect to fragments of the sentence, and not with respect to the entire sentence at each stage. Hence, the interpretation function is only a partial implementation of MacCartney's algorithm.

verbs of category $\mathcal{V}^{\mathcal{L}_B}$. All of these verbs only take one argument, namely the subject of the phrase, which is in turn taken from class $\mathcal{N}^{\mathcal{L}_B}$. This is an extremely basic syntax, which is reminiscent of Aristotle's syllogistic (McKeon et al. 2009; Leszl 2004). Yet, although there is discussion about the expressibility of binary predication in Aristotelian terms (van Benthem 2007), it is certain that only monadic predicates are allowed in $\mathcal{L}_B$. In this respect, it is severely limited.

B.4 Symmetric hierarchy

The terms in the classes $\mathcal{N}^{\mathcal{L}_B}$ and $\mathcal{V}^{\mathcal{L}_B}$ of $\mathcal{L}_B$ are hierarchically related as illustrated by the Venn diagrams in Figures 3.1 and 3.2. Predicates whose visualization is contained in another one relate to the concerned concept as hyponyms. Conversely, they are hypernyms of the terms that they themselves contain. Now, what is striking about the taxonomy of $\mathcal{N}^{\mathcal{L}_B}$ is a kind of symmetry in Figure 3.1. Warthogs form a subset of mammals, are disjoined from all reptiles (including turtles), and independent of pets. Similarly, turtles form a subset of reptiles, are disjoined from all mammals (including warthogs), and independent of pets. For a clearer overview, the lexical entailment relations between the different nominal terms are shown in Table B.2.

	warthogs	turtles	mammals	reptiles	pets
warthogs	=	|	<	|	#
turtles	|	=	|	<	#
mammals	>	|	=	|	#
reptiles	|	>	|	=	#
pets	#	#	#	#	=

Table B.2: Lexical entailment relations between nouns of $\mathcal{N}^{\mathcal{L}_B}$. (Recall from Table 2.1 that = denotes equivalence, < forward entailment, > backward entailment, | alternation and # independence.)

The exact kind of symmetry occurring here is perhaps best illustrated by the arrows in Table B.2. The part of the table containing the actual entailment relations can be regarded as the 5×5-dimensional matrix $\mathbf{M}$. There is a permutation mapping the first column of $\mathbf{M}$ to its second column, which we call σ. If the entries per column are identified by means of their row index, then this permutation can be written in cyclic notation as $\sigma = (12)(34)(5)$. Now, we observe that the same permutation σ maps the third column of $\mathbf{M}$ to its fourth column. This is shown by the identical arrow pattern between the respective column pairs.

Symmetries of this kind decrease the 'diversity' of a taxonomy. Because of the noted analogies, the hierarchy can effectively be partitioned into two spaces with great

set-theoretic similarities. It is preferable to use a hierarchy with more variation. The symmetries in such a small class as $\mathcal{N}^{\mathcal{L}_B}$ could enable a model to apply set-theoretic shortcuts in the learning process that are unavailable in usual contexts. Potentially, such a process could in itself qualify as a form of applied logic, but because it would be the result of a highly specific taxonomy it could not rightfully serve as a basis for general conclusions.

B.5 Composed quantifiers

The quantifiers of $\mathcal{L}_B$ together form the set $\mathcal{Q}^{\mathcal{L}_B}$ (see Section 3.1.1). As noted before, $\mathcal{Q}^{\mathcal{L}_B}$ comprises the positive quantifiers `all`, `some`, `most`, `two`, `three` and their negated counterparts `not_all`, `no`, `not_most`, `lt_two`, `lt_three`. The negated quantifiers constitute individual lexical items, even though their semantics allow for decomposition into the combination of negation (`not`) with a positive quantifier.

Bowman does not explain this design choice, but it is evident that it simplifies the classification task. Consider the challenge currently facing a model: for each quantifier in $\mathcal{Q}^{\mathcal{L}_B}$, an individual word embedding is learned. Compare this to the task when the negated quantifiers are decomposed. The set of quantifiers would then be limited to `all`, `some`, `most`, `two`, `three`. Vector representations of negated quantifiers would be the result of an application of the composition function to the word embeddings of negation and a positive quantifier. In this alternative set-up, negation is represented by the same (trained) embedding, irrespectively of the quantifier that it is combined with. Likewise, the representations for positive quantifiers are applied both in negated and unnegated contexts. As there are no specific lexical items anymore to capture the particular nuances of all ten quantifiers in $\mathcal{Q}^{\mathcal{L}_B}$, a model dealing with decomposed determiners has to learn fewer representations with higher generalizability. This is probably most challenging in the case of negation, which can be prefixed to all other vocabulary items: the nouns in $\mathcal{N}^{\mathcal{L}_B}$, the verbs in $\mathcal{V}^{\mathcal{L}_B}$, and now also the positive quantifiers in $\mathcal{Q}^{\mathcal{L}_B}$.

In short, the inclusion of synthetic quantifiers as primitive vocabulary items simplifies the problem. At the same time, it makes the language $\mathcal{L}_B$ more artificial than an analogue with only positive determiners and a universal negation symbol that can be combined with all of them. Hence, decomposing quantifiers would not only make the task more interesting, but also more natural.

B.6 Relative size of training data

The grammar of $\mathcal{L}_B$ only produces a finite set of admissible formulations. This allows us to express the collection of sentence pairs per training set as a fraction of the total space of possible instances. As computed on page 20, the training sets used by Bowman contain approximately 4.2% of all possible pairs. Here we do not even take into consideration that pairs labelled 'unknown' are discarded. Phrased differently, this means that more than one in twenty-five of all possible cases is seen during each training

epoch. This is a large fragment of the total space. Of course, deep learning methods are known for their 'data hunger', and analogies with human learning processes are always precarious, but observing every twenty-fifth instance of a problem seems excessive at any rate. It certainly is an infeasible percentage to maintain for even slightly more complex languages. Assuming a combinatorial generation process as mentioned on page 18, the number of possible sentences grows exponentially with respect to the vocabulary size. A training set containing 4% (or any constant fraction) of the total space of possibilities would soon become unmanageable.

B.7 Baseline performance

There is something remarkable about the results reported in Table 3.2. The matrix and tensor models obtain almost perfect scores, but the sumNN performs very well too. In fact, the difference between the accuracies of the tree-shaped models and the simple baseline is so small that it raises suspicion. In Bowman's experiments, the baseline reaches a training accuracy that is, on average, only 2.9 percentage points below the score obtained by the recursive models (4.3 in the replication). For the testing accuracy, the average difference is 5.6 percentage points (5.6 in the replication). The superiority of the tree-shaped models is clearly not negligible. Nevertheless, their complexity and training times are so much higher than the baseline's that a difference of only a few percentage points is quite meager. It demonstrates that the features distinguishing them from such a basic model as the summing neural network only account for a minimal fraction of the obtained scores. The fact that sophisticated models only manage to slightly outperform a rudimentary baseline on this problem indicates that the task might be trivial. Instead, we would prefer a more challenging task, so that increased model complexity becomes indispensable, and not just a means to boost performance by a few percents.

B.8 Symbolic prerequisites

The presented models are essentially data-driven, or statistical. This is to say that they are not endowed with an integrated representation of the problem, but learn to address it from the large number of pre-labelled instances in the training sets. In this sense they differ from symbolic models, which handle problems by means of a knowledge base shaped according to human perception, interpretation or strategy. The main advantage of statistical models is that they do not require such a thorough understanding of a problem as their symbolic counterparts in order to achieve a high robustness in the face of new information or unexpected nuances. Instead of relying on the exhaustive description of a scenario, they develop their own, implicit representations, which are updated on the basis of new experiences as learning progresses. This increases their flexibility and scalability.

Although Bowman's recursive models are *essentially* data-driven, they are not completely so. This is because the network topology, as described in Section 3.1.2, is determined by the parse trees of the input sentences. This syntactic structure is symbolic

information, which is assumed to be present in all the data. Therefore, although the models as such do not encode any linguistic information other than the syntactic binarity encapsulated in the composition function, they do require that the data provide additional symbolic details. In this sense, it may be more appropriate to call the recursive models 'semi-symbolic': they learn from data, but require the data to be extended with a great number of linguistic cues in the form of brackets. This method could be considered as a form of symbolic guidance, by its way of guiding the computational process according to symbolic prompts.

Thanks to the automated data generation process and the limited variation between grammatical configurations, the syntactic shape of the input sentences can easily be added to the data constructed in language $\mathcal{L}_B$. Because this is not generally the case, it adds another artificial factor to the current data. When dealing with natural language, the issue could be overcome by implementing an automated parser before the entailment classifier, as is done in the final section of Bowman, Potts, and Manning 2015, but this comes at the price of great computational overhead and propagation of uncertainty. It would be worthwhile to pursue an approach that is less dependent on such pre-processing or symbolic prerequisites.

B.9 Lack of interpretation

The final issue is no objection to Bowman's experiments. It concerns something that is missing in the methodology, namely a more thorough interpretation of the obtained results, the models and their learned representations. Now that we know that the tree-shaped models perform well on this particular task, we would like to know more precisely *how* they manage to do so. This involves 'opening the black box' of the networks and assessing their internal dynamics. Research focusing on such matters is part of the field of interpretable machine learning. For other studies taking this approach see e.g. Vellido, Martín-Guerrero, and Lisboa 2012, Veldhoen, Hupkes, and Zuidema 2016 or Doshi-Velez and Kim 2017.

Here we list a few examples of interpretative questions that are worth addressing in this particular case:

- What is the relation between the learned word embeddings? Are vector representations of semantically related items clustered together?
- How exactly does the composition function combine and transform word embeddings while recursively processing a sentence?
- Do entailment relations determine geometric regularities in the sentence vector space?

Many other interpretative issues can be thought of. In general, the internal model representations (all intermediate entities between input and output) deserve to be studied in further detail to understand more clearly how the networks do what they do, and to what extent their individual components contribute to the over-all performance.

Appendix C

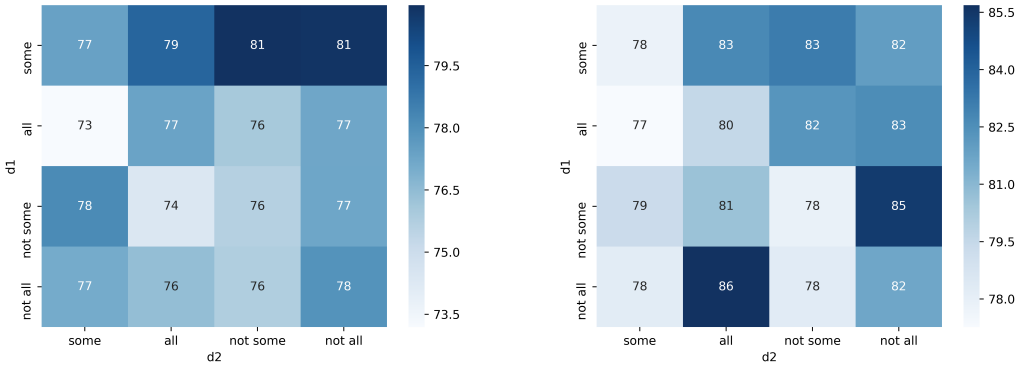
Logical generalization experiment

When dealing with the standard `binary_fol` data, much of the generalization performed by the models is combinatorial. Words of language $\mathcal{L}_N$ can be joined to form a finite number of possible combinations (sentences), and a finite number of combinations of these combinations (sentence pairs). A small fraction of this set is witnessed during training in order to extrapolate to new configurations with the same structural properties. Opposed to combinatorial generalization is hierarchic generalization, which is not just a matter of combining data as a basis for abstraction, but also of exploiting systematic connections between implicitly related instances. Models that learn through hierarchic generalization do not exclusively rely on the amount of processed data to succeed, but achieve an effect of synergy by recognizing and utilizing particular stratifications in the data.

One kind of hierarchic generalization that is interesting to consider here is logical generalization: do the models capture aspects of FOL that allow them to learn more from the available data than classifiers lacking such structural awareness? We will investigate this by means of an experiment focusing on the first quantifier of $\mathcal{L}_N$ sentences. In the `binary_fol` data, models are faced with several kinds of quantification at the beginning of a sentence: it can be universal or existential, and negated or unnegated. Thus, if we let the initial determiner refer to the initial quantifier together with its optional negation, it has four possible values: 'some', 'all', 'not some' and 'not all'. It follows that, considering sentence pairs, there are 16 possible combinations of first determiners. The data can be partitioned into 16 corresponding classes. We name each of these classes $C(d_1, d_2)$, where d_1 denotes the first determiner of the first sentence of the pairs, and d_2 the first determiner of the second sentence of the pairs in the class.

If a model applies logical generalization, it exploits the systematic relation between different classes $C(d_1, d_2)$ to improve its performance. E.g., observing instances from class C(some, not all) does not only increase accuracy for this class, but also for the class C(some, all) or C(all, not some) by virtue of the internalized logical associations between the quantifier combinations. In `binary_fol`, all classes are collected together, so that it is unclear if this effect takes place. To investigate if it does, the `binary_fol` train and test data are subdivided into the sixteen distinct (d_1, d_2)-categories. The resulting train and test sets respectively contain some 2,000 and 600 instances on average. Models

are trained and tested on these subsets. Only the SRN and the GRU are considered in this experiment, because the GRU has proven to consistently outperform the LSTM, while the latter's training times are longer. The SRN remains relevant to consider as a recurrent baseline. The final achievements on the test set, averaged over five runs, are visualized in the heatmaps of Figures C.1.



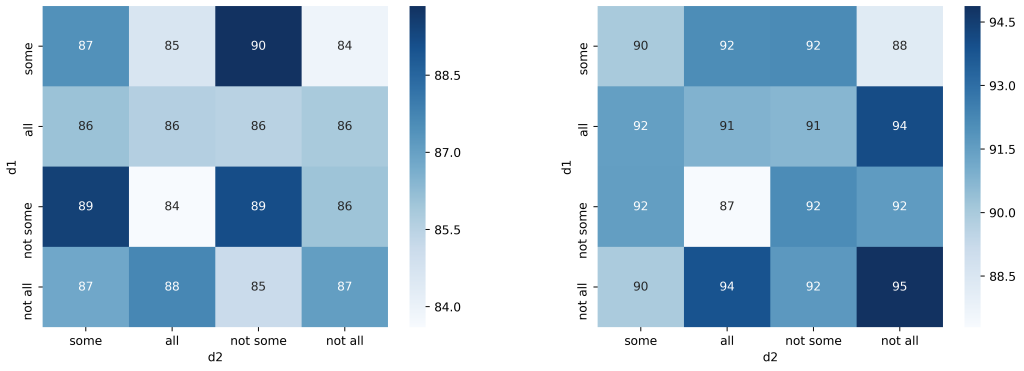
(a) SRN (b) GRU

Figure C.1: Heatmaps visualizing the mean testing accuracy scores on the FOL inference task for models trained and tested on the classes $C(d_1, d_2)$, isolated from the standard `binary_fol` data ($\sim$ 2,000 train and 600 test instances per class). d_i denotes the first determiner of the ith sentence in each pair. Results are averaged over five runs.

Recalling the results in Table 5.1, the SRN obtained a mean testing accuracy of approximately 87% on the `binary_fol` test set. The average of the testing scores on the isolated $C(d_1, d_2)$ classes is 77%, which is 10 percentage points lower. The GRU realized a mean testing accuracy of 96% on the `binary_fol` data, whereas its current average is 81%, which is 15 percentage points lower. Evidently, performance deteriorates when training and evaluation are relative to the 16 (d_1, d_2)-classes. These subclasses do not only restrict the original `binary_fol` data in terms of quantifier configurations, but also in size. Therefore, it cannot be concluded that the higher scores on the `binary_fol` are purely caused by logical generalization. But the difference cannot be explained by data size alone either. If this were the case, the SRN and GRU should suffer just as much, but the observed deterioration is stronger for the GRU (15 percentage points or 16%) than for the SRN (10 percentage points or 11%). This shows that the GRU makes more efficient use of the extra data in the `binary_fol` set than the SRN, which is possibly due to logical generalization.

It is difficult to determine the degree to which superior performance on data combining the $C(d_1, d_2)$ classes is a consequence of logical generalization or of the fact that there are simply more training instances. The difference between the SRN and the GRU suggests that, at least for the GRU, it cannot be a matter of training size alone. To rule out this option more confidently, a follow-up experiment is conducted. If the classifiers

perform better on `binary_fol`, just because there are more training instances than in the individual $C(d_1, d_2)$ sets, then the data statistics must be more or less homogeneous with respect to the (d_1, d_2)-subclasses. If not, they would represent 16 unique and unrelated classification problems, so that combination into `binary_fol` should not yield higher testing scores. To assess this, sixteen new $C(d_1, d_2)$ data sets are generated, with approximately 7,000 training and 1,500 testing instances per class. The `binary_fol` data are downsampled to a comparable size. The SRN and GRU are trained on all sixteen new $C(d_1, d_2)$ sets, and on the downsampled `binary_fol` data. If the data really are homogeneous across (d_1, d_2)-subclasses, the final testing scores should be similar.



(a) SRN (b) GRU

Figure C.2: Heatmaps visualizing the mean testing accuracy scores on the FOL inference task for models trained and tested on the classes $C(d_1, d_2)$ of newly generated data ($\sim$ 7,000 train and 1,500 test instances per class). d_i denotes the first determiner of the ith sentence in each pair. Results are averaged over five runs.

This is not the case. Mean scores per subclass are shown in Figure C.2. For the SRN, the average testing accuracy obtained is approximately 86%. For the GRU, it is 91%. After training on data with the same size that combine all these classes, the mean testing accuracy is 78% for the SRN, and 83% for the GRU. The significant difference between the mean accuracy scores for the individual $C(d_1, d_2)$ sets and the combined class of the same size shows that the data are not uniform across the subclasses $C(d_1, d_2)$. This means that the sets $C(d_1, d_2)$ are not just random samples from the combined data, but that they represent subproblems with a unique structure. As the partitioning is based on the initial determiners of the sentences, the distinction between the different classes is a logical one. Originally, the GRU and SRN were trained on the `binary_fol` data, which has approximately 2,000 training instances per $C(d_1, d_2)$ class. This is far below the 7,000 instances seen during training by the models whose heatmaps are shown in Figure C.2. Nevertheless, the mean testing accuracy on these smaller sets is lower than the one on the `binary_fol` set. The difference is marginal for the SRN (86% vs 87%), but significant for the GRU (91% vs 96%). It thus appears that the GRU exploits the

relation between the different classes, which is a logical one, in a way that cannot be explained by the amount of data alone. This is evidence that the model performs at least a rudimentary form of logical generalization.

Bibliography

Allamanis, Miltiadis et al. (2016). "Learning continuous semantic representations of symbolic expressions". In: *arXiv preprint arXiv:1611.01423*.

Bankova, Desislava et al. (2016). "Graded entailment for compositional distributional semantics". In: *arXiv preprint arXiv:1601.04908*.

Baroni, Marco, Raffaella Bernardi, et al. (2012). "Entailment above the word level in distributional semantics". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 23–32.

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski (2014). "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1, pp. 238–247.

Barwise, Jon and Robin Cooper (1981). "Generalized quantifiers and natural language". In: *Philosophy, Language, and Artificial Intelligence*. Springer, pp. 241–301.

van Benthem, Johan (1987). "Meaning: interpretation and inference". In: *Synthese* 73.3, pp. 451–470.

— (2007). *A brief history of natural logic*. ILLC Prepublication Series. URL: `https://www.illc.uva.nl/Research/Publications/Reports/PP/`.

van Benthem, Johan and Dag Westerståhl (1995). "Directions in generalized quantifier theory". In: *Studia Logica* 55.3, pp. 389–419.

Bluche, Theodore, Christopher Kermorvant, and Jérôme Louradour (2015). "Where to apply dropout in recurrent neural networks for handwriting recognition?" In: *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, pp. 681–685.

Boolos, George S, John P Burgess, and Richard C Jeffrey (2007). *Computability and Logic*. 5th ed. Cambridge: Cambridge University Press.

Bos, Johan and Katja Markert (2005). "Recognising textual entailment with logical inference". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 628–635.

Bowman, Samuel R (2013). "Can recursive neural tensor networks learn logical reasoning?" In: *arXiv preprint arXiv:1312.6192*.

Bowman, Samuel R., Gabor Angeli, et al. (2015). "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Bowman, Samuel R, Christopher Potts, and Christopher D Manning (2015). "Recursive neural networks can learn logical semantics". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*. (July 26–31, 2015). Association for Computational Linguistics. Beijing, China, pp. 12–21.

Bullinaria, John A and Joseph P Levy (2007). "Extracting semantic representations from word co-occurrence statistics: A computational study". In: *Behavior research methods* 39.3, pp. 510–526.

Calvo, Paco and John Symons (2014). *The Architecture of Cognition: Rethinking Fodor and Pylyshyn's Systematicity Challenge*. MIT Press.

Carston, R (1997). "Language of Thought". In: *Concise Encyclopedia of Philosophy of Language*, pp. 68–70.

Chen, Danqi et al. (2013). "Learning new facts from knowledge bases with neural tensor networks and semantic word vectors". In: *arXiv preprint arXiv:1301.3618*.

Chomsky, Noam (1957). *Syntactic structures*. Mouton, Berlin.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.

Church, Alonzo (1936). "A note on the Entscheidungsproblem". In: *Journal of Symbolic Logic* 1, pp. 40–41.

Coecke, B, M Sadrzadeh, and S Clark (2010). "Mathematical Foundations for a Compositional Distributional Model of Meaning". In: *Linguistic Analysis* 36, pp. 345–384.

Costa, Mario et al. (1999). "Short term load forecasting using a synchronously operated recurrent neural network". In: *Neural Networks, 1999. IJCNN'99. International Joint Conference on*. Vol. 5. IEEE, pp. 3478–3482.

Delavenay, Emile (1960). *An Introduction to Machine Translation*. New York, NY: Thames and Hudson.

Doshi-Velez, Finale and Been Kim (2017). "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608*.

Dowty, David (2007). "Compositionality as an empirical problem". In: *Direct compositionality* 14, pp. 23–101.

van Eijck, Jan (2005). "Natural logic for natural language". In: *International Tbilisi Symposium on Logic, Language, and Computation*. Springer, pp. 216–230.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Endrullis, Jörg and Lawrence S Moss (2015). "Syllogistic logic with 'most'". In: *International Workshop on Logic, Language, Information, and Computation*. Springer, pp. 124–139.

Evans, Richard and Edward Grefenstette (2018). "Learning Explanatory Rules from Noisy Data". In: *Journal of Artificial Intelligence Research* 61, pp. 1–64.

Evans, Richard, David Saxton, et al. (2018). "Can Neural Networks Understand Logical Entailment?" In: *arXiv preprint arXiv:1802.08535*.

Fausett, Laurene (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc.

Firth, John Rupert (1957). *Papers in Linguistics 1934-1951*. London: Oxford University Press.

Fitch, W Tecumseh (2014). "Toward a computational framework for cognitive biology: unifying approaches from cognitive neuroscience and comparative cognition". In: *Physics of life reviews* 11.3, pp. 329–364.

Fodor, Jerry (1998). "There are no recognitional concepts; not even RED". In: *Philosophical issues* 9, pp. 1–14.

Fodor, Jerry A and Zenon W Pylyshyn (1988). "Connectionism and cognitive architecture: A critical analysis". In: *Cognition* 28.1-2, pp. 3–71.

Frege, Gottlob ([1914] 1980). "Letter to Jourdain". In: *Philosophical and Mathematical Correspondence*. Ed. by Gottfried Gabriel et al. Trans. by Hans Kaal. Chicago: Chicago University Press, pp. 78–80.

Giles, C Lee, Steve Lawrence, and Ah Chung Tsoi (1997). "Rule inference for financial prediction using recurrent neural networks". In: *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*. IEEE, pp. 253–259.

Gödel, Kurt (1929). "Uber die Vollstandigkeit des Logikkalkuls". PhD thesis. Vienna.

— (1930). "Die Vollständigkeit der Axiome des logischen Funktionenkalküls". In: *Monatshefte für Mathematik und Physik* 37.1, pp. 349–360.

— (1931). "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I". In: *Monatshefte für mathematik und physik* 38.1, pp. 173–198.

Goldberg, Adele E (2015). "Compositionality". In: *The Routledge handbook of semantics*. Ed. by Nick Riemer. Routledge. URL: `https://adele.princeton.edu`.

Graves, Alex and Navdeep Jaitly (2014). "Towards end-to-end speech recognition with recurrent neural networks". In: *International Conference on Machine Learning*, pp. 1764–1772.

Grice, H. Paul (1975). "Logic and conversation". In: *Syntax and Semantics, Volume 3: Speech Acts*. Ed. by P. Cole and J. Morgan. Academic Press, New York, pp. 41–58.

Groenendijk, J and M Stokhof (2004). "Why Compositionality?" In: *The Partee Effect*. Ed. by G N Carlson and J F Pelletier. Stanford, CA: CSLI Publications, pp. 1–20.

Hackbusch, Wolfgang (2012). *Tensor spaces and numerical tensor calculus*. Vol. 42. Springer Science & Business Media.

Harris, Zellig S (1954). "Distributional structure". In: *Word* 10.23, pp. 146–162.

Henderson, James and Diana Nicoleta Popa (2016). "A vector space for distributional semantics for entailment". In: *arXiv preprint arXiv:1607.03780*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hupkes, Dieuwke, Sara Veldhoen, and Willem Zuidema (2017). "Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure". In: *arXiv preprint arXiv:1711.10203*.

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

Kaiser, Łukasz and Ilya Sutskever (2015). "Neural gpus learn algorithms". In: *arXiv preprint arXiv:1511.08228*.

Karpathy, Andrej, Justin Johnson, and Li Fei-Fei (2015). "Visualizing and understanding recurrent networks". In: *arXiv preprint arXiv:1506.02078*.

Keenan, Edward (1996). "The semantics of determiners". In: *The handbook of contemporary semantic theory*. Ed. by Shalom Lappin. Blackwell Oxford, pp. 41–63.

Kolda, Tamara G and Brett W Bader (2009). "Tensor decompositions and applications". In: *SIAM review* 51.3, pp. 455–500.

Lahav, Ran (1989). "Against compositionality: the case of adjectives". In: *Philosophical studies* 57.3, pp. 261–279.

Lai, Tri, Jörg Endrullis, and Lawrence S Moss (2016). "Majority Digraphs". In: *Proceedings of the American Mathematical Society* 144.9, pp. 3701–3715.

Lake, Brenden M and Marco Baroni (2017). "Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks". In: *arXiv preprint arXiv:1711.00350*.

Lake, Brenden M, Tomer D Ullman, et al. (2017). "Building machines that learn and think like people". In: *Behavioral and Brain Sciences* 40.

Le, Phong and Willem Zuidema (2015). "Compositional distributional semantics with long short term memory". In: *arXiv preprint arXiv:1503.02510*.

Lee, Lillian (1999). "Measures of distributional similarity". In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pp. 25–32.

Leszl, Walter (2004). "Aristotle's logical works and his conception of logic". In: *Topoi* 23.1, pp. 71–100.

Letsche, Todd A and Michael W Berry (1997). "Large-scale information retrieval with latent semantic indexing". In: *Information sciences* 100.1-4, pp. 105–137.

Liang, Sheng-Fu, Alvin WY Su, and Cheng-Teng Lin (1999). "A new recurrent-network-based music synthesis method for chinese plucked-string instruments-pipa and qin". In: *Neural Networks, 1999. IJCNN'99. International Joint Conference on*. Vol. 4. IEEE, pp. 2564–2569.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. ICML*. Vol. 30.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

MacCartney, Bill (2009). "Natural language inference". PhD thesis. Stanford University. URL: `https://nlp.stanford.edu/~wcmac/papers/nli-diss.pdf`.

MacCartney, Bill and Christopher D Manning (2009). "An extended model of natural logic". In: *Proceedings of the eighth international conference on computational semantics*. Association for Computational Linguistics, pp. 140–156.

MacQueen, James et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, pp. 281–297.

Marcus, Gary F (1998). "Rethinking eliminative connectionism". In: *Cognitive psychology* 37.3, pp. 243–282.

— (2003). *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

Marelli, Marco et al. (2014). "SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment." In: *SemEval@ COLING*, pp. 1–8.

McCune, W (2010). "Prover9 and Mace4". URL: `http://www.cs.unm.edu/~mccune/prover9`.

McKeon, Richard et al. (2009). *The basic works of Aristotle*. Modern Library.

Medsker, Larry and Lakhmi C Jain (1999). *Recurrent neural networks: design and applications*. CRC press.

Mendelson, Elliott (1987). *Introduction to Mathematical Logic*. Belmont: Wadsworth & Brooks.

Mikolov, Tomas, Kai Chen, et al. (2013). *Efficient estimation of word representations in vector space*. ICLR Workshop.

Mikolov, Tomas, Ilya Sutskever, et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "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*, pp. 746–751.

Mitchell, Jeff and Mirella Lapata (2008). "Vector-based models of semantic composition". In: *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pp. 236–244.

Mohammad, Saif M et al. (2013). "Computing lexical contrast". In: *Computational Linguistics* 39.3, pp. 555–590.

Montague, Richard (1970). "Universal grammar". In: *Theoria* 36.3, pp. 373–398.

Mostowski, Marcin (1995). "Quantifiers definable by second order means". In: *Quantifiers: logics, models and computation*. Springer, pp. 181–214.

Mozer, Michael C (1989). "A focused backpropagation algorithm for temporal pattern recognition". In: *Complex Systems* 3, pp. 349–381.

Mueller, Jonas and Aditya Thyagarajan (2016a). "Siamese Recurrent Architectures for Learning Sentence Similarity." In: *AAAI*, pp. 2786–2792.

— (2016b). "Siamese Recurrent Architectures for Learning Sentence Similarity". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2786–2792.

Pantel, Patrick (2005). "Inducing ontological co-occurrence vectors". In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 125–132.

Partee, Barbara (1984). "Compositionality". In: *Varieties of formal semantics: Proceedings of the fourth Amsterdam colloquium*. Ed. by F Landman and F Veltman. Dordrecht: Foris Publications, pp. 281–311.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*, pp. 1310–1318.

Paszke, Adam et al. (2017). *Automatic differentiation in PyTorch*. 31st Conference on Neural Information Processing Systems.

Pearson, Karl (1901). "On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Pelletier, Francis Jeffry (1994). "The principle of semantic compositionality". In: *Topoi* 13.1, pp. 11–24.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Reed, Scott and Nando De Freitas (2015). "Neural programmer-interpreters". In: *arXiv preprint arXiv:1511.06279*.

Ren, He and Quan Yang (2013). *Neural Joke Generation*. 31st Conference on Neural Information Processing Systems.

Repplinger, Michael (2017). *Understanding Generalization: Learning Quantifiers and Negation with Neural Tensor Networks*. MSc Thesis, University of Amsterdam.

Rinaldi, Fabio et al. (2003). "Exploiting paraphrases in a question answering system". In: *Proceedings of the Second International Workshop on Paraphrasing*. Association for Computational Linguistics. Saporo, Japan, pp. 25–32.

Robinson, AJ and Frank Fallside (1987). *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering.

Rocktäschel, Tim, Matko Bošnjak, et al. (2014). "Low-dimensional embeddings of logic". In: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pp. 45–49.

Rocktäschel, Tim, Edward Grefenstette, et al. (2015). "Reasoning about entailment with neural attention". In: *arXiv preprint arXiv:1509.06664*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Sadrzadeh, Mehrnoosh and Dimitri Kartsaklis (2016). "Compositional Distributional Models of Meaning". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Tutorial Abstracts*, pp. 1–4.

Sadrzadeh, Mehrnoosh, Dimitri Kartsaklis, and Esma Balkır (2018). "Sentence entailment in compositional distributional semantics". In: *Annals of Mathematics and Artificial Intelligence*, pp. 1–30.

Sánchez Valencia, Victor (1991). "Studies on Natural Logic and Categorial Grammar". PhD thesis. University of Amsterdam.

Schütze, H and J Pedersen (1995). "Information retrieval based on word senses". In: *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*. Vol. 4, pp. 161–175.

Serafini, Luciano and Artur d'Avila Garcez (2016). "Logic tensor networks: Deep learning and logical reasoning from data and knowledge". In: *arXiv preprint arXiv:1606.04422*.

Shapiro, Stewart (1991). *Foundations without foundationalism: A case for second-order logic*. Vol. 17. Clarendon Press.

Sider, Theodore (2010). *Logic for philosophy*. Oxford: Oxford University Press.

Socher, Richard, Brody Huval, et al. (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*. Association for Computational Linguistics, pp. 1201–1211.

Socher, Richard, Andrej Karpathy, et al. (2014). "Grounded compositional semantics for finding and describing images with sentences". In: *Transactions of the Association of Computational Linguistics* 2.1, pp. 207–218.

Sommers, Fred (1982). *The logic of natural language*. Cambridge University Press.

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

Szabó, Zoltán Gendler (2004). "Compositionality". In: *Stanford Encyclopedia of Philosophy*. URL: `https://plato.stanford.edu/entries/compositionality/`.

Tai, Kai Sheng, Richard Socher, and Christopher D Manning (2015). "Improved semantic representations from tree-structured long short-term memory networks". In: *arXiv preprint arXiv:1503.00075*.

Tatu, Marta and Dan Moldovan (2005). "A semantic approach to recognizing textual entailment". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 371–378.

Travis, Charles (1994). "On constraints of generality". In: *Proceedings of the Aristotelian Society*. Vol. 94. JSTOR, pp. 165–188.

Turing, Alan Mathison (1936). "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.42, pp. 230–265.

Veldhoen, Sara (2015). *Semantic adequacy of compositional distributed representations*. MSc Thesis, University of Amsterdam.

Veldhoen, Sara, Dieuwke Hupkes, and Willem Zuidema (2016). "Diagnostic Classifiers Revealing how Neural Networks Process Hierarchical Structure." In: *CoCo@ NIPS*.

Vellido, Alfredo, José David Martín-Guerrero, and Paulo JG Lisboa (2012). "Making machine learning models interpretable." In: *ESANN*. Vol. 12. Citeseer, pp. 163–172.

Vinyals, Oriol et al. (2015). "Grammar as a foreign language". In: *Advances in Neural Information Processing Systems*, pp. 2773–2781.

Weaver, George (2012). *Adding the quantifier 'most' to first-order logic.*

Werbos, Paul J (1988). "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4, pp. 339–356.

Zeiler, Matthew D (2012). "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701*.