Do Neural Models Learn Systematicity of Monotonicity Inference in Natural Language?

Hitomi Yanaka[1], **Koji Mineshima**[2], **Daisuke Bekki**[3], **and Kentaro Inui**[4,1]
[1]RIKEN, [2]Keio University, [3]Ochanomizu University, [4]Tohoku University
hitomi.yanaka@riken.jp, minesima@abelard.flet.keio.ac.jp,
bekki@is.ocha.ac.jp, inui@ecei.tohoku.ac.jp

Abstract

Despite the success of language models using neural networks, it remains unclear to what extent neural models have the generalization ability to perform inferences. In this paper, we introduce a method for evaluating whether neural models can learn *systematicity* of monotonicity inference in natural language, namely, the regularity for performing arbitrary inferences with generalization on composition. We consider four aspects of monotonicity inferences and test whether the models can systematically interpret lexical and logical phenomena on different training/test splits. A series of experiments show that three neural models systematically draw inferences on unseen combinations of lexical and logical phenomena when the syntactic structures of the sentences are similar between the training and test sets. However, the performance of the models significantly decreases when the structures are slightly changed in the test set while retaining all vocabularies and constituents already appearing in the training set. This indicates that the generalization ability of neural models is limited to cases where the syntactic structures are nearly the same as those in the training set.

1 Introduction

Natural language inference (NLI), a task whereby a system judges whether given a set of premises P semantically entails a hypothesis H (Dagan et al., 2013; Bowman et al., 2015), is a fundamental task for natural language understanding. As with other NLP tasks, recent studies have shown a remarkable impact of deep neural networks in NLI (Williams et al., 2018; Wang et al., 2019; Devlin et al., 2019). However, it remains unclear to what extent DNN-based models are capable of learning the compositional generalization underlying NLI from given labeled training instances.

Systematicity of inference (or *inferential systematicity*) (Fodor and Pylyshyn, 1988; Aydede, 1997) in natural language has been intensively studied in the field of formal semantics. From among the various aspects of inferential systematicity, in the context of NLI, we focus on *monotonicity* (van Benthem, 1983; Icard and Moss, 2014) and its *productivity*. Consider the following premise–hypothesis pairs (1)–(3), which have the target label *entailment*:

(1) P: ***Some*** [*puppies*↑] *ran.*
H: ***Some*** *dogs ran.*

(2) P: ***No*** [*cats*↓] *ran.*
H: ***No*** *small cats ran.*

(3) P: ***Some*** [*puppies which chased* ***no*** [*cats*↓]] *ran.*
H: ***Some*** *dogs which chased* ***no*** *small cats ran.*

As in (1), for example, quantifiers such as *some* exhibit **upward monotone** (shown as [... ↑]), and replacing a phrase in an upward-entailing context in a sentence with a more general phrase (replacing *puppies* in P with *dogs* as in H) yields a sentence inferable from the original sentence. In contrast, as in (2), quantifiers such as *no* exhibit **downward monotone** (shown as [... ↓]), and replacing a phrase in a downward-entailing context with a more specific phrase (replacing *cats* in P with *small cats* as in H) yields a sentence inferable from the original sentence. Such primitive inference patterns combine recursively as in (3). This manner of monotonicity and its productivity produces a potentially infinite number of inferential patterns. Therefore, NLI models must be capable of systematically interpreting such primitive patterns and reasoning over unseen combinations of patterns. Although many studies have addressed this issue by modeling logical reasoning in formal semantics (Abzianidze, 2015; Mineshima et al., 2015; Hu et al., 2019) and testing DNN-based models on monotonicity inference (Yanaka et al., 2019a,b; Richardson et al.,

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6105–6117
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

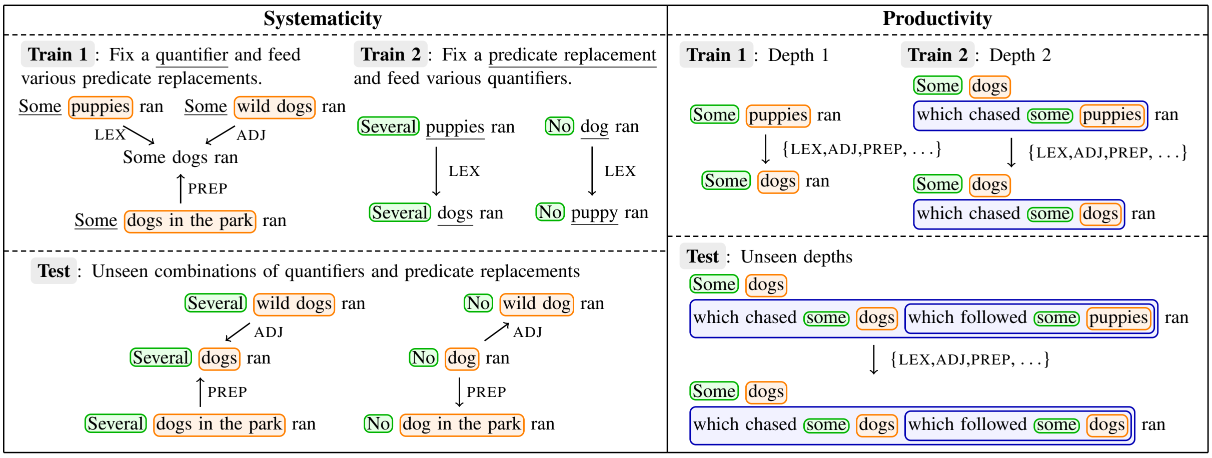
Systematicity
Train 1 : Fix a quantifier and feed various predicate replacements.
Some puppies ran
Some wild dogs ran
LEX
ADJ
Some dogs ran
PREP
Some dogs in the park ran
Train 2 : Fix a predicate replacement and feed various quantifiers.
Several puppies ran
No dog ran
LEX
LEX
Several dogs ran
No puppy ran
Test : Unseen combinations of quantifiers and predicate replacements
Several wild dogs ran
No wild dog ran
ADJ
ADJ
Several dogs ran
No dog ran
PREP
PREP
Several dogs in the park ran
No dog in the park ran
Productivity
Train 1 : Depth 1
Some puppies ran
{LEX,ADJ,PREP, . . .}
Some dogs ran
Train 2 : Depth 2
Some dogs which chased some puppies ran
{LEX,ADJ,PREP, . . .}
Some dogs which chased some dogs ran
Test : Unseen depths
Some dogs which chased some dogs which followed some puppies ran
{LEX,ADJ,PREP, . . .}
Some dogs which chased some dogs which followed some dogs ran

Figure 1: An illustration of the basic idea. For Systematicity and Productivity, we train models on the **Train 1** and **Train 2** sets and test them on the **Test** set. Arrow (⟶) means entailment relation; LEX, ADJ, and PREP mean predicate replacements for lexical relations, adjectives, and prepositional phrases, respectively. In Productivity, we use various quantifiers and predicate replacements in each depth.

2020), the ability of DNN-based models to generalize to unseen combinations of patterns is still underexplored.

Given this background, we investigate the systematic generalization ability of DNN-based models on four aspects of monotonicity: (i) systematicity of predicate replacements (i.e., replacements with a more general or specific phrase), (ii) systematicity of embedding quantifiers, (iii) productivity, and (iv) localism (see Section 2.2). To this aim, we introduce a new evaluation protocol where we (i) synthesize training instances from sampled sentences and (ii) systematically control which patterns are shown to the models in the training phase and which are left unseen. The rationale behind this protocol is twofold. First, patterns of monotonicity inference are highly systematic, so we can create training data with arbitrary combinations of patterns, as in examples (1)–(3). Second, evaluating the performance of the models trained with well-known NLI datasets such as MultiNLI (Williams et al., 2018) might severely underestimate the ability of the models because such datasets tend to contain only a limited number of training instances that exhibit the inferential patterns of interest. Furthermore, using such datasets would prevent us from identifying which combinations of patterns the models can infer from which patterns in the training data.

This paper makes two primary contributions. First, we introduce an evaluation protocol[1] using the systematic control of the training/test split under various combinations of semantic properties to evaluate whether models learn inferential systematicity in natural language. Second, we apply our evaluation protocol to three NLI models and present evidence suggesting that, while all models generalize to unseen combinations of lexical and logical phenomena, their generalization ability is limited to cases where sentence structures are nearly the same as those in the training set.

2 Method

2.1 Basic idea

Figure 1 illustrates the basic idea of our evaluation protocol on monotonicity inference. We use synthesized monotonicity inference datasets, where NLI models should capture both (i) monotonicity directions (upward/downward) of various quantifiers and (ii) the types of various predicate replacements in their arguments. To build such datasets, we first generate a set of premises $G^{\mathbf{Q}}_d$ by a context-free grammar G with depth d (i.e., the maximum number of applications of recursive rules), given a set of quantifiers $\mathbf{Q}$. Then, by applying $G^{\mathbf{Q}}_d$ to elements of a set of functions for predicate replacements (or *replacement functions* for short) $\mathbf{R}$ that rephrase a constituent in the input premise and return a hypothesis, we obtain a set $\mathbf{D}^{\mathbf{Q},\mathbf{R}}_d$ of premise–hypothesis pairs defined as

$$\mathbf{D}^{\mathbf{Q},\mathbf{R}}_d = \{(P, H) \mid P \in G^{\mathbf{Q}}_d, \exists r \in \mathbf{R}\ (r(P) = H)\}.$$

For example, the premise *Some puppies ran* is generated from the quantifier *some* in $\mathbf{Q}$ and

[1] The evaluation code will be publicly available at https://github.com/verypluming/systematicity.

the production rule $S \rightarrow Q, N, IV$, and thus it is an element of $G_1^{\mathbf{Q}}$. By applying this premise to a replacement function that replaces the word in the premise with its hypernym (e.g., *puppy* $\sqsubseteq$ *dog*), we provide the premise–hypothesis pair ***Some*** *puppies ran* ⇒ ***Some*** *dogs ran* in Fig. 1.

We can control which patterns are shown to the models during training and which are left unseen by systematically splitting $\mathbf{D}_d^{\mathbf{Q},\mathbf{R}}$ into training and test sets. As shown on the left side of Figure 1, we consider how to test the systematic capacity of models with unseen combinations of quantifiers and predicate replacements. To expose models to primitive patterns regarding $\mathbf{Q}$ and $\mathbf{R}$, we fix an arbitrary element q from $\mathbf{Q}$ and feed various predicate replacements into the models from the training set of inferences $\mathbf{D}_d^{\{q\},\mathbf{R}}$ generated from combinations of the fixed quantifier and all predicate replacements. Also, we select an arbitrary element r from $\mathbf{R}$ and feed various quantifiers into the models from the training set of inferences $\mathbf{D}_d^{\mathbf{Q},\{r\}}$ generated from combinations of all quantifiers and the fixed predicate replacement.

We then test the models on the set of inferences generated from unseen combinations of quantifiers and predicate replacements. That is, we test them on the set of inferences $\mathbf{D}_d^{\overline{\{q\}},\overline{\{r\}}}$ generated from the complements $\overline{\{q\}}, \overline{\{r\}}$ of $\{q\}, \{r\}$. If models capture inferential systematicity in combinations of quantifiers and predicate replacements, they can correctly perform all inferences in $\mathbf{D}_d^{\overline{\{q\}},\overline{\{r\}}}$ on an arbitrary split based on q, r.

Similarly, as shown on the right side of Figure 1, we can test the productive capacity of models with unseen depths by changing the training/test split based on d. For example, by training models on $\mathbf{D}_d^{\mathbf{Q},\mathbf{R}}$ and testing them on $\mathbf{D}_{d+1}^{\mathbf{Q},\mathbf{R}}$, we can evaluate whether models generalize to one deeper depth. By testing models with an arbitrary training/test split of $\mathbf{D}_d^{\mathbf{Q},\mathbf{R}}$ based on semantic properties of monotonicity inference (i.e., quantifiers, predicate replacements, and depths), we can evaluate whether models systematically interpret them.

2.2 Evaluation protocol

To test NLI models from multiple perspectives of inferential systematicity in monotonicity inferences, we focus on four aspects: (i) systematicity of predicate replacements, (ii) systematicity of embedding quantifiers, (iii) productivity, and (iv) localism. For each aspect, we use a set $\mathbf{D}_d^{\mathbf{Q},\mathbf{R}}$ of premise–hypothesis pairs. Let $\mathbf{Q} = \mathbf{Q}^{\uparrow} \cup \mathbf{Q}^{\downarrow}$ be the union of a set of selected upward quantifiers $\mathbf{Q}^{\uparrow}$ and a set of selected downward quantifiers $\mathbf{Q}^{\downarrow}$ such that $|\mathbf{Q}^{\uparrow}| = |\mathbf{Q}^{\downarrow}| = n$. Let $\mathbf{R}$ be a set of replacement functions $\{r_1, \ldots, r_m\}$, and d be the embedding depth, with $1 \leq d \leq s$. (4) is an example of an element of $\mathbf{D}_1^{\mathbf{Q},\mathbf{R}}$, containing the quantifier *some* in the subject position and the predicate replacement using the hypernym relation *dogs* $\sqsubseteq$ *animals* in its upward-entailing context without embedding.

(4) P: ***Some*** *dogs ran* ⇒ H: ***Some*** *animals ran*

I. Systematicity of predicate replacements The following describes how we test the extent to which models generalize to unseen combinations of quantifiers and predicate replacements. Here, we expose models to all primitive patterns of predicate replacements like (4) and (5) and all primitive patterns of quantifiers like (6) and (7). We then test whether the models can systematically capture the difference between upward quantifiers (e.g., *several*) and downward quantifiers (e.g., *no*) as well as the different types of predicate replacements (e.g., the lexical relation *dogs* $\sqsubseteq$ *animals* and the adjective deletion *small dogs* $\sqsubseteq$ *dogs*) and correctly interpret unseen combinations of quantifiers and predicate replacements like (8) and (9).

(5) P: ***Some*** *small dogs ran* ⇒ H: ***Some*** *dogs ran*
(6) P: ***Several*** *dogs ran* ⇒ H: ***Several*** *animals ran*
(7) P: ***No*** *animals ran* ⇒ H: ***No*** *dogs ran*
(8) P: ***Several*** *small dogs ran* ⇒ H: ***Several*** *dogs ran*
(9) P: ***No*** *dogs ran* ⇒ H: ***No*** *small dogs ran*

Here, we consider a set of inferences $\mathbf{D}_1^{\mathbf{Q},\mathbf{R}}$ whose depth is 1. We move from harder to easier tasks by gradually changing the training/test split according to combinations of quantifiers and predicate replacements. First, we expose models to primitive patterns of $\mathbf{Q}$ and $\mathbf{R}$ with the minimum training set. Thus, we define the initial training set $\mathbf{S}_1$ and test set $\mathbf{T}_1$ as follows:

$$(\mathbf{S}_1, \mathbf{T}_1) = (\mathbf{D}_1^{\{q\},\mathbf{R}} \cup \mathbf{D}_1^{\mathbf{Q},\{r\}},\ \mathbf{D}_1^{\overline{\{q\}},\overline{\{r\}}})$$

where q is arbitrarily selected from $\mathbf{Q}$, and r is arbitrarily selected from $\mathbf{R}$.

Next, we gradually add the set of inferences generated from combinations of an upward–downward quantifier pair and all predicate replacements to the training set. In the examples above, we add (8) and (9) to the training set to simplify the task. We assume a set $\mathbf{Q}'$ of a pair of upward/downward quantifiers, namely, $\{(q^{\uparrow}, q^{\downarrow}) \mid (q^{\uparrow}, q^{\downarrow}) \subseteq \mathbf{Q}^{\uparrow} \times \mathbf{Q}^{\downarrow},\ q^{\uparrow}, q^{\downarrow} \neq q\}$. We consider a set $\mathsf{perm}(\mathbf{Q}')$ consisting of permutations of $\mathbf{Q}'$. For each $p \in \mathsf{perm}(\mathbf{Q}')$, we gradually add a set of inferences generated from $p(i)$ to the training set $\mathbf{S}_i$ with $1 < i \leq n-1$. Then, we provide a test set $\mathbf{T}_i$ generated from the complement $\overline{\mathbf{Q}_i}$ of $\mathbf{Q}_i = \{x \mid \exists y(x,y) \in \mathbf{Q}'_i$ or $\exists y(y,x) \in \mathbf{Q}'_i\}$ and $\overline{\{r\}}$ where $\mathbf{Q}'_i = \{p(1), \dots, p(i)\}$. This protocol is summarized as

$$\mathbf{S}_{i+1} = \mathbf{S}_i \cup \mathbf{D}_1^{\{q_i^\uparrow, q_i^\downarrow\},\mathbf{R}},$$
$$\mathbf{T}_i = \mathbf{D}_1^{\overline{\mathbf{Q}_i},\overline{\{r\}}} \quad \text{with } 1 < i \leq n-1$$

where $(q_i^\uparrow, q_i^\downarrow) = p(i)$.

To evaluate the extent to which the generalization ability of models is robust for different syntactic structures, we use an additional test set $\mathbf{T}'_i = \mathbf{D}_1^{\overline{\mathbf{Q}_i},\overline{\{r\}}}$ generated using three production rules. The first is the case where one adverb is added at the beginning of the sentence, as in example (10).

(10) P_{adv}: *Slowly,* ***several*** *small dogs ran*
H_{adv}: *Slowly,* ***several*** *dogs ran*

The second is the case where a three-word prepositional phrase is added at the beginning of the sentence, as in example (11).

(11) P_{prep}: *Near the shore,* ***several*** *small dogs ran*
H_{prep}: *Near the shore,* ***several*** *dogs ran*

The third is the case where the replacement is performed in the object position, as in example (12).

(12) P_{obj}: *Some tiger touched* ***several*** *small dogs*
H_{obj}: *Some tiger touched* ***several*** *dogs*

We train and test models $|\mathsf{perm}(\mathbf{Q}')|$ times, then take the average accuracy as the final evaluation result.

II. Systematicity of embedding quantifiers To properly interpret embedding monotonicity, models should detect both (i) the monotonicity direction of each quantifier and (ii) the type of predicate replacements in the embedded argument. The following describes how we test whether models generalize to unseen combinations of embedding quantifiers. We expose models to all primitive combination patterns of quantifiers and predicate replacements like (4)–(9) with a set of non-embedding monotonicity inferences $\mathbf{D}_1^{\mathbf{Q},\mathbf{R}}$ and some embedding patterns like (13), where $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are chosen from a selected set of upward or downward quantifiers such as *some* or *no*. We then test the models with an inference with an unseen quantifier *several* in (14) to evaluate whether models can systematically interpret embedding quantifiers.

(13) P: $\boldsymbol{Q}_1$ *animals that chased* $\boldsymbol{Q}_2$ *dogs ran*
H: $\boldsymbol{Q}_1$ *animals that chased* $\boldsymbol{Q}_2$ *animals ran*

(14) P: ***Several*** *animals that chased* ***several*** *dogs ran*
H: ***Several*** *animals that chased* ***several*** *animals ran*

We move from harder to easier tasks of learning embedding quantifiers by gradually changing the training/test split of a set of inferences $\mathbf{D}_2^{\mathbf{Q},\mathbf{R}}$ whose depth is 2, i.e., inferences involving one embedded clause.

We assume a set $\mathbf{Q}'$ of a pair of upward and downward quantifiers as $\mathbf{Q}' \equiv \{(q^\uparrow, q^\downarrow) \mid (q^\uparrow, q^\downarrow) \subseteq \mathbf{Q}^\uparrow \times \mathbf{Q}^\downarrow\}$, and consider a set $\mathsf{perm}(\mathbf{Q}')$ consisting of permutations of $\mathbf{Q}'$. For each $p \in \mathsf{perm}(\mathbf{Q}')$, we gradually add a set of inferences $\mathbf{D}_2$ generated from $p(i)$ to the training set $\mathbf{S}_i$ with $1 \leq i \leq n-1$.

We test models trained with $\mathbf{S}_i$ on a test set $\mathbf{T}_i$ generated from the complement $\overline{\mathbf{Q}_i}$ of $\mathbf{Q}_i = \{x \mid \exists y(x,y) \in \mathbf{Q}'_i$ or $\exists y(y,x) \in \mathbf{Q}'_i\}$ where $\mathbf{Q}'_i = \{p(1), \dots, p(i)\}$, summarized as

$$\mathbf{S}_0 = \mathbf{D}_1^{\mathbf{Q},\mathbf{R}},$$
$$\mathbf{S}_i = \mathbf{S}_{i-1} \cup \mathbf{D}_2^{\{q_i^\uparrow, q_i^\downarrow\},\mathbf{R}},$$
$$\mathbf{T}_i = \mathbf{D}_2^{\overline{\mathbf{Q}_i},\mathbf{R}} \quad \text{with } 1 \leq i \leq n-1$$

where $(q_i^\uparrow, q_i^\downarrow) = p(i)$. We train and test models $|\mathsf{perm}(\mathbf{Q}')|$ times, then take the average accuracy as the final evaluation result.

III. Productivity Productivity (or *recursiveness*) is a concept related to systematicity, which refers to the capacity to grasp an indefinite number of natural language sentences or thoughts with generalization on composition. The following describes how we test whether models generalize to unseen deeper depths in embedding monotonicity (see also the right side of Figure 1). For example, we expose models to all primitive non-embedding/single-embedding patterns like (15) and (16) and then test them with deeper embedding patterns like (17).

(15) P: ***Some*** *dogs ran*
H: ***Some*** *animals ran*

(16) P: ***Some*** *animals which chased* ***some*** *dogs ran*
H: ***Some*** *animals which chased* ***some*** *animals ran*

Depth	Pred.	Monotone	Arg.	**Example** (premise, hypothesis, label)	Avg. Len.
1	CONJ	DOWNWARD	SECOND	***Less than three*** *lions left.* ***Less than three*** *lions left and cried.* ENTAILMENT	4.6
2	PP	UPWARD	FIRST	***Few*** *lions that hurt* ***at most three*** *small dogs walked.* ***Few*** *lions that hurt* ***at most three*** *dogs walked.* ENTAILMENT	9.0
3	AdJ	DOWNWARD	FIRST	***Some*** *elephant* ***no*** *rabbit which touched* ***a few*** *dogs hit rushed.* ***Some*** *elephant* ***no*** *rabbit which touched* ***a few*** *small dogs hit rushed.* ENTAILMENT	12.3
4	RC	UPWARD	FIRST	***Less than three*** *tigers which accepted* ***several*** *rabbits that loved* ***several*** *foxes* ***more than three*** *monkeys cleaned dawdled.* ***Less than three*** *tigers which accepted* ***several*** *rabbits that loved* ***several*** *foxes* ***more than three*** *monkeys which ate dinner cleaned dawdled.* ENTAILMENT	16.6

Table 1: Examples of generated premise–hypothesis pairs. Depth: depth of embedding; Pred.: type of predicate replacements; Monotone: direction of monotonicity; Arg.: argument where the predicate replacement is performed; Avg. Len.: average sentence length.

(17) P: ***Some*** *animals which chased* ***some*** *cats which followed* ***some*** *dogs ran*
H: ***Some*** *animals which chased* ***some*** *cats which followed* ***some*** *animals ran*

To evaluate models on the set of inferences involving embedded clauses with depths exceeding those in the training set, we train models with $\bigcup_{d\in\{1,\ldots,i+1\}} \mathbf{D}_d$, where we refer to $\mathbf{D}_d^{\mathbf{Q},\mathbf{R}}$ as $\mathbf{D}_d$ for short, and test the models on $\bigcup_{d\in\{i+2,\ldots,s\}} \mathbf{D}_d$ with $1 \leq i \leq s-2$.

IV. Localism According to the principle of compositionality, the meaning of a complex expression derives from the meanings of its constituents and how they are combined. One important concern is how local the composition operations should be (Pagin and Westerståhl, 2010). We therefore test whether models trained with inferences involving embedded monotonicity locally perform inferences composed of smaller constituents. Specifically, we train models with examples like (17) and then test the models with examples like (15) and (16). We train models with $\mathbf{D}_d$ and test the models on $\bigcup_{k\in\{1,\ldots,d\}} \mathbf{D}_k$ with $3 \leq d \leq s$.

3 Experimental Setting

3.1 Data creation

To prepare the datasets shown in Table 1, we first generate premise sentences involving quantifiers from a set of context-free grammar (CFG) rules and lexical entries, shown in Table 6 in the Appendix. We select 10 words from among nouns, intransitive verbs, and transitive verbs as lexical entries. A set of quantifiers $\mathbf{Q}$ consists of eight elements; we use a set of four downward quantifiers $\mathbf{Q}^{\downarrow}$ ={*no, at most three, less than three, few*} and a set of four upward quantifiers $\mathbf{Q}^{\uparrow}$ ={*some, at least three, more than three, a few*}, which have the same monotonicity directions in the first and second arguments. We thus consider $n = |\mathbf{Q}^{\uparrow}| = |\mathbf{Q}^{\downarrow}| = 4$ in the protocol in Section 2.2. The ratio of each monotonicity direction (upward/downward) of generated sentences is set to $1:1$. We then generate hypothesis sentences by applying replacement functions to premise sentences according to the polarities of constituents. The set of replacement functions $\mathbf{R}$ is composed of the seven types of lexical replacements and phrasal additions in Table 2. We remove unnatural premise–hypothesis pairs in which the same words or phrases appear more than once.

Function	**Example**
r_1: hyponym	*dogs* $\sqsubseteq$ *animals*
r_2: adjective	*small dogs* $\sqsubseteq$ *dogs*
r_3: preposition	*dogs in the park* $\sqsubseteq$ *dogs*
r_4: relative clause	*dogs which ate dinner* $\sqsubseteq$ *dogs*
r_5: adverb	*ran quickly* $\sqsubseteq$ *ran*
r_6: disjunction	*ran* $\sqsubseteq$ *ran or walked*
r_7: conjunction	*ran and barked* $\sqsubseteq$ *ran*

Table 2: Examples of replacement functions.

For embedding monotonicity, we consider inferences involving four types of replacement functions in the first argument of the quantifier in Table 2: hyponyms, adjectives, prepositions, and relative clauses. We generate sentences up to the depth $d = 5$. There are various types of embedding monotonicity, including relative clauses, conditionals, and negated clauses. In this paper, we consider three types of embedded clauses: peripheral-embedding clauses and two kinds of center-embedding clauses, shown in Table 6 in the Appendix.

The number of generated sentences exponentially increases with the depth of embedded clauses. Thus, we limit the number of inference examples to 320,000, split into 300,000 examples for the training set and 20,000 examples for the test set. We guarantee that all combinations of quantifiers are included in the set of inference examples for each depth. Gold labels for generated premise–hypothesis pairs are automatically determined according to the polarity of the argument position (upward/downward) and the type of predicate replacements (with more general/specific phrases). The ratio of each gold label (entailment/non-entailment) in the training and test sets is set to $1:1$.

To double-check the gold label, we translate each premise–hypothesis pair into a logical formula (see the Appendix for more details). The logical formulas are obtained by combining lambda terms in accordance with meaning composition rules specified in the CFG rules in the standard way (Blackburn and Bos, 2005). We prove the entailment relation using the theorem prover Vampire[2], checking whether a proof is found in time for each entailment pair. For all pairs, the output of the prover matched with the entailment relation automatically determined by monotonicity calculus.

3.2 Models

We consider three DNN-based NLI models. The first architecture employs long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). We set the number of layers to three with no attention. Each premise and hypothesis is processed as a sequence of words using a recurrent neural network with LSTM cells, and the final hidden state of each serves as its representation.

The second architecture employs multiplicative tree-structured LSTM (TreeLSTM) networks (Tran and Cheng, 2018), which are expected to be more sensitive to hierarchical syntactic structures. Each premise and hypothesis is processed as a tree structure by bottom-up combinations of constituent nodes using the same shared compositional function, input word information, and between-word relational information. We parse all premise–hypothesis pairs with the dependency parser using the spaCy library[3] and obtain tree structures. For each experimental setting, we randomly sample 100 tree structures and check their correctness. In LSTM and TreeLSTM, the dimension of hidden units is 200, and we initialize the word embeddings with 300-dimensional GloVe vectors (Pennington et al., 2014). Both models are optimized with Adam (Kingma and Ba, 2015), and no dropout is applied.

The third architecture is a Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019). We used the base-uncased model pre-trained on Wikipedia and BookCorpus from the pytorch-pretrained-bert library[4], fine-tuned for the NLI task using our dataset. In fine-tuning BERT, no dropout is applied, and we choose hyperparameters that are commonly used for MultiNLI. We train all models over 25 epochs or until convergence, and select the best-performing model based on its performance on the validation set. We perform five runs per model and report the average and standard deviation of their scores.

4 Experiments and Discussion

I. Systematicity of predicate replacements Figure 2 shows the performance on unseen combinations of quantifiers and predicate replacements. In the minimal training set $\mathbf{S}_1$, the accuracy of LSTM and TreeLSTM was almost the same as chance, but that of BERT was around 75%, suggesting that only BERT generalized to unseen combinations of quantifiers and predicate replacements. When we train BERT with the training set $\mathbf{S}_2$, which contains inference examples generated from combinations of one pair of upward/downward quantifiers and all predicate replacements, the accuracy was 100%. This indicates that by being taught two kinds of quantifiers in the training data, BERT could distinguish between upward and downward for the other quantifiers. The accuracy of LSTM and TreeLSTM increased with increasing the training set size, but did not reach 100%. This indicates that LSTM and TreeLSTM also generalize to inferences involving similar quantifiers to some extent, but their generalization ability is imperfect.

When testing models with inferences where adverbs or prepositional phrases are added to the be-

[2] https://github.com/vprover/vampire

[3] https://spacy.io/

[4] https://github.com/huggingface/pytorch-pretrained-bert

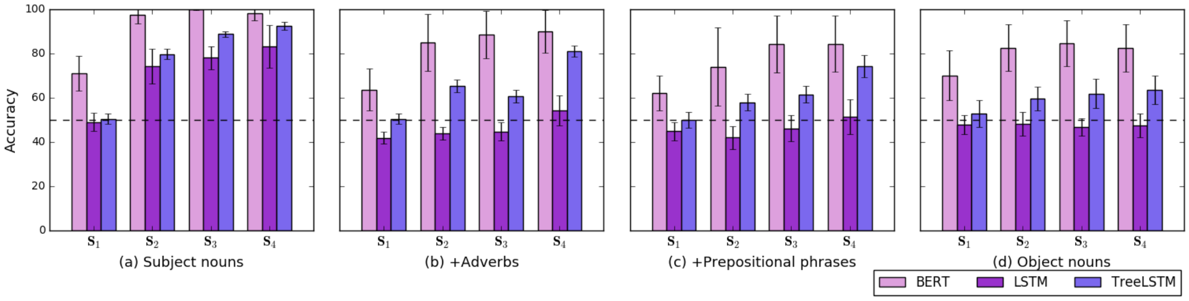

Figure 2: Results for systematicity of predicate replacements. Accuracy on test sets where (a) the replacement is performed in the subject position, (b) one adverb is added at the beginning of the sentence, (c) one three-word prepositional phrase is added at the beginning of the sentence, and (d) the replacement is in the object position. $\mathbf{S}_n$ indicates the experimental setting where the training set $\mathbf{S}_n$ is used.

ginning of the sentence, the accuracy of all models significantly decreased. This decrease becomes larger as the syntactic structures of the sentences in the test set become increasingly different from those in the training set. Contrary to our expectations, the models fail to maintain accuracy on test sets whose difference from the training set is the structure with the adverb at the beginning of a sentence. Of course, we could augment datasets involving that structure, but doing so would require feeding all combinations of inference pairs into the models. These results indicate that the models tend to estimate the entailment label from the beginning of a premise–hypothesis sentence pair, and that inferential systematicity to draw inferences involving quantifiers and predicate replacements is not completely generalized at the level of arbitrary constituents.

II. Systematicity of embedding quantifiers Figure 3 shows the performance of all models on unseen combinations of embedding quantifiers. Even when adding the training set of inferences involving one embedded clause and two quantifiers step-by-step, no model showed improved performance. The accuracy of BERT slightly exceeded chance, but the accuracy of LSTM and TreeLSTM was nearly the same as or lower than chance. These results suggest that all the models fail to generalize to unseen combinations of embedding quantifiers even when they involve similar upward/downward quantifiers.

III. Productivity Table 3 shows the performance on unseen depths of embedded clauses. The accuracy on $\mathbf{D}_1$ and $\mathbf{D}_2$ was nearly 100%, indicating that all models almost completely generalize to inferences containing previously seen depths. When

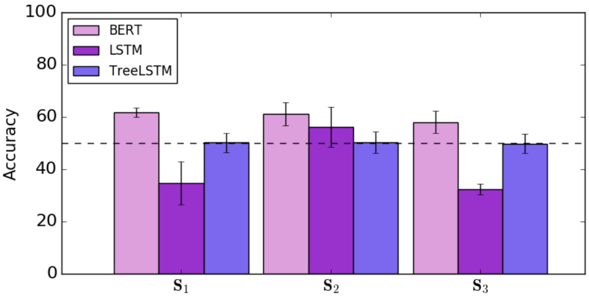

Figure 3: Results for systematicity of embedding quantifiers. $\mathbf{S}_n$ indicates the experimental setting where the training set $\mathbf{S}_n$ is used.

$\mathbf{D}_1 + \mathbf{D}_2$ were used as the training set, the accuracy of all models on $\mathbf{D}_3$ exceeded chance. Similarly, when $\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$ were used as the training set, the accuracy of all models on $\mathbf{D}_4$ exceeded chance. This indicates that all models partially generalize to inferences containing embedded clauses one level deeper than the training set.

However, standard deviations of BERT and LSTM were around 10, suggesting that these models did not consistently generalize to inferences containing embedded clauses one level deeper than the training set. While the distribution of monotonicity directions (upward/downward) in the training and test sets was uniform, the accuracy of LSTM and BERT tended to be smaller for downward inferences than for upward inferences. This also indicates that these models fail to properly compute monotonicity directions of constituents from syntactic structures. The standard deviation of TreeLSTM was smaller, indicating that TreeLSTM robustly learns inference patterns containing embedded clauses one level deeper than the training set.

Train	Dev/Test	BERT	LSTM	TreeLSTM
$\mathbf{D}_1 + \mathbf{D}_2$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.1
	$\mathbf{D}_2$	100.0±0.0	99.8±0.2	99.5±0.1
	$\mathbf{D}_3$	75.2±10.0	75.4±10.8	86.4±4.1
	$\mathbf{D}_4$	55.0±3.7	57.7±8.7	58.6±7.8
	$\mathbf{D}_5$	49.9±4.4	45.8±4.0	48.4±3.7
	$\mathbf{D}_3$ (down)	71.2±4.0	70.4±4.0	86.4±4.1
	$\mathbf{D}_3$ (up)	80.5±7.5	84.7±4.9	86.4±4.1
$\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.0
	$\mathbf{D}_2$	100.0±0.0	95.1±7.8	99.6±0.0
	$\mathbf{D}_3$	100.0±0.0	85.2±8.9	97.7±1.1
	$\mathbf{D}_4$	77.9±10.8	59.7±10.8	68.0±5.6
	$\mathbf{D}_5$	53.5±19.6	55.1±8.2	49.6±4.3
	$\mathbf{D}_4$ (down)	85.8±10.5	76.9±6.6	68.0±5.6
	$\mathbf{D}_4$ (up)	86.8±1.8	81.1±5.6	68.0±5.6

Table 3: Results for productivity. $\mathbf{D}_d$ indicates the set of inferences where the embedding depth is d.

Train	Dev/Test	BERT	LSTM	TreeLSTM
$\mathbf{D}_3$	$\mathbf{D}_1$	49.6±0.5	48.8±13.2	49.8±4.1
	$\mathbf{D}_2$	49.8±0.6	47.3±12.1	51.8±1.1
	$\mathbf{D}_3$	100.0±0.0	100.0±0.0	100.0±0.2
$\mathbf{D}_4$	$\mathbf{D}_1$	50.3±1.0	46.8±6.5	49.0±0.4
	$\mathbf{D}_2$	49.6±0.8	45.4±1.8	49.7±0.3
	$\mathbf{D}_3$	50.2±0.7	45.1±0.6	50.5±0.7
	$\mathbf{D}_4$	100.0±0.0	100.0±0.0	100.0±0.1
$\mathbf{D}_5$	$\mathbf{D}_1$	49.9±0.7	43.7±4.4	49.1±1.1
	$\mathbf{D}_2$	49.1±0.3	43.4±3.9	51.4±0.6
	$\mathbf{D}_3$	50.6±0.2	44.3±2.7	50.5±0.3
	$\mathbf{D}_4$	50.9±0.8	44.4±3.4	50.3±0.4
	$\mathbf{D}_5$	100.0±0.0	100.0±0.0	100.0±0.1

Table 4: Results for localism.

However, the performance of all models trained with $\mathbf{D}_1 + \mathbf{D}_2$ on $\mathbf{D}_4$ and $\mathbf{D}_5$ significantly decreased. Also, performance decreased for all models trained with $\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$ on $\mathbf{D}_5$. Specifically, there was significantly decreased performance of all models, including TreeLSTM, on inferences containing embedded clauses two or more levels deeper than those in the training set. These results indicate that all models fail to develop productivity on inferences involving embedding monotonicity.

IV. Localism Table 4 shows the performance of all models on localism of embedding monotonicity. When the models were trained with $\mathbf{D}_3$, $\mathbf{D}_4$ or $\mathbf{D}_5$, all performed at around chance on the test set of non-embedding inferences $\mathbf{D}_1$ and the test set of inferences involving one embedded clause $\mathbf{D}_2$. These results indicate that even if models are trained with a set of inferences containing complex syntactic structures, the models fail to locally interpret their constituents.

Train	Dev/Test	BERT	LSTM	TreeLSTM
MNLI	$\mathbf{D}_1$	46.9±0.4	47.2±1.1	43.4±0.3
	$\mathbf{D}_2$	46.2±0.6	48.3±1.0	49.5±0.4
	$\mathbf{D}_3$	46.8±0.8	48.9±0.7	41.0±0.4
	$\mathbf{D}_4$	48.5±0.8	50.6±0.5	48.5±0.2
	$\mathbf{D}_5$	48.9±0.6	49.3±0.7	48.8±0.5
	MNLI-test	84.6±0.2	64.7±0.3	70.4±0.1
$\mathbf{D}_1 + \mathbf{D}_2$ +MNLI	$\mathbf{D}_1$	100.0±0.0	100.0±0.1	100.0±0.1
	$\mathbf{D}_2$	100.0±0.0	89.3±9.0	99.8±0.1
	$\mathbf{D}_3$	67.8±12.5	66.7±13.5	76.3±4.1
	$\mathbf{D}_4$	46.8±3.7	47.1±14.6	50.7±7.8
	$\mathbf{D}_5$	41.2±4.3	46.7±11.2	47.5±3.7
	MNLI-test	84.4±0.2	39.7±0.5	63.0±0.2
$\mathbf{D}_1 + \mathbf{D}_2 + \mathbf{D}_3$ +MNLI	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.0
	$\mathbf{D}_2$	100.0±0.0	97.1±5.0	99.8±0.0
	$\mathbf{D}_3$	100.0±0.0	89.2±5.1	98.3±1.1
	$\mathbf{D}_4$	70.9±7.9	73.4±10.9	76.1±5.6
	$\mathbf{D}_5$	42.4±4.2	47.8±3.9	57.0±4.3
	MNLI-test	84.0±0.1	39.7±0.4	62.8±0.2

Table 5: Results for productivity where models were trained with our synthesized dataset mixed with MultiNLI (MNLI).

Performance of data augmentation Prior studies (Yanaka et al., 2019b; Richardson et al., 2020) have shown that given BERT initially trained with MultiNLI, further training with synthesized instances of logical inference improves performance on the same types of logical inference while maintaining the initial performance on MultiNLI. To investigate whether the results of our study are transferable to current work on MultiNLI, we trained models with our synthesized dataset mixed with MultiNLI, and checked (i) whether our synthesized dataset degrades the original performance of models on MultiNLI[5] and (ii) whether MultiNLI degrades the ability to generalize to unseen depths of embedded clauses.

Table 5 shows that training BERT on our synthetic data $\mathbf{D}_1 + \mathbf{D}_2$ and MultiNLI increases the accuracy on our test sets $\mathbf{D}_1$ (46.9 to 100.0), $\mathbf{D}_2$ (46.2 to 100.0), and $\mathbf{D}_3$ (46.8 to 67.8) while preserving accuracy on MultiNLI (84.6 to 84.4). This indicates that training BERT with our synthetic data does not degrade performance on commonly used corpora like MultiNLI while improving the performance on monotonicity, which suggests that our data-synthesis approach can be combined with naturalistic datasets. For TreeLSTM and LSTM, however, adding our synthetic dataset decreases accuracy on MultiNLI. One possible reason for this is that a pre-training based model like BERT can mitigate catastrophic forgetting in various types of datasets.

Regarding the ability to generalize to unseen depths of embedded clauses, the accuracy of all

[5] Following the previous work (Richardson et al., 2020), we used the MultiNLI mismatched development set for MNLI-test.

models on our synthetic test set containing embedded clauses one level deeper than the training set exceeds chance, but the improvement becomes smaller with the addition of MultiNLI. In particular, with the addition of MultiNLI, the models tend to change wrong predictions in cases where a hypothesis contains a phrase not occurring in a premise but the premise entails the hypothesis. Such inference patterns are contrary to the heuristics in MultiNLI (McCoy et al., 2019). This indicates that there may be some trade-offs in terms of performance between inference patterns in the training set and those in the test set.

5 Related Work

The question of whether neural networks are capable of processing compositionality has been widely discussed (Fodor and Pylyshyn, 1988; Marcus, 2003). Recent empirical studies illustrate the importance and difficulty of evaluating the capability of neural models. Generation tasks using artificial datasets have been proposed for testing whether models compositionally interpret training data from the underlying grammar of the data (Lake and Baroni, 2017; Hupkes et al., 2018; Saxton et al., 2019; Loula et al., 2018; Hupkes et al., 2019; Bernardy, 2018). However, these conclusions are controversial, and it remains unclear whether the failure of models on these tasks stems from their inability to deal with compositionality.

Previous studies using logical inference tasks have also reported both positive and negative results. Assessment results on propositional logic (Evans et al., 2018), first-order logic (Mul and Zuidema, 2019), and natural logic (Bowman et al., 2015) show that neural networks can generalize to unseen words and lengths. In contrast, Geiger et al. (2019) obtained negative results by testing models under fair conditions of natural logic. Our study suggests that these conflicting results come from an absence of perspective on combinations of semantic properties.

Regarding assessment of the behavior of modern language models, Linzen et al. (2016), Tran et al. (2018), and Goldberg (2019) investigated their syntactic capabilities by testing such models on subject–verb agreement tasks. Many studies of NLI tasks (Liu et al., 2019; Glockner et al., 2018; Poliak et al., 2018; Tsuchiya, 2018; McCoy et al., 2019; Rozen et al., 2019; Ross and Pavlick, 2019) have provided evaluation methodologies and found that current NLI models often fail on particular inference types, or that they learn undesired heuristics from the training set. In particular, recent works (Yanaka et al., 2019a,b; Richardson et al., 2020) have evaluated models on monotonicity, but did not focus on the ability to generalize to unseen combinations of patterns. Monotonicity covers various systematic inferential patterns, and thus is an adequate semantic phenomenon for assessing inferential systematicity in natural language. Another benefit of focusing on monotonicity is that it provides hard problem settings against heuristics (McCoy et al., 2019), which fail to perform downward-entailing inferences where the hypothesis is longer than the premise.

6 Conclusion

We introduced a method for evaluating whether DNN-based models can learn systematicity of monotonicity inference under four aspects. A series of experiments showed that the capability of three models to capture systematicity of predicate replacements was limited to cases where the positions of the constituents were similar between the training and test sets. For embedding monotonicity, no models consistently drew inferences involving embedded clauses whose depths were two levels deeper than those in the training set. This suggests that models fail to capture inferential systematicity of monotonicity and its productivity.

We also found that BERT trained with our synthetic dataset mixed with MultiNLI maintained performance on MultiNLI while improving the performance on monotonicity. This indicates that though current DNN-based models do not systematically interpret monotonicity inference, some models might have sufficient ability to memorize different types of reasoning. We hope that our work will be useful in future research for realizing more advanced models that are capable of appropriately performing arbitrary inferences.

Acknowledgement

We thank the three anonymous reviewers for their helpful comments and suggestions. We are also grateful to Benjamin Heinzerling and Sosuke Kobayashi for helpful discussions. This work was partially supported by JSPS KAKENHI Grant Numbers JP20K19868 and JP18H03284, Japan.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*, pages 2492–2502.

Murat Aydede. 1997. Language of thought: The connectionist contribution. *Minds and Machines*, 7(1):57–101.

Johan van Benthem. 1983. Determiners and logic. *Linguistics and Philosophy*, 6(4):447–478.

Jean-Philippe Bernardy. 2018. Can recurrent neural networks learn nested recursion. *Linguistic Issues in Language Technology*, 16.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Center for the Study of Language and Information.

Samuel R. Bowman, 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*, pages 12–21.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan & Claypool Publishers.

Jacob Devlin, Chang Ming-Wei, Lee Kenton, and Toutanova Kristina. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2019)*, pages 4171–4186.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? In *International Conference on Learning Representations (ICLR-2018)*.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2019. Posing fair generalization tasks for natural language inference. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP-2019)*, pages 4484–4494.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL-2018)*, pages 650–655.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

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

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S. Moss, and Sandra Kübler. 2019. Monalog: a lightweight system for natural language inference based on monotonicity. *CoRR*, abs/1910.08772.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2019. Compositionality decomposed: how do neural networks generalise? *CoRR*, abs/1908.08351.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Thomas Icard and Lawrence Moss. 2014. Recent progress in monotonicity. *Linguistic Issues in Language Technology*, 9(7):167–194.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR-2015)*.

Brenden M. Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML-2017)*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2019)*, pages 2171–2179.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114.

Gary Marcus. 2003. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL-2019), pages 3428–3448.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*, pages 2055–2061.

Mathijs Mul and Willem Zuidema. 2019. Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization. *CoRR*, abs/1908.08351.

Peter Pagin and Dag Westerståhl. 2010. Compositionality I: Definitions and variants. *Philosophy Compass*, 5(3):250–264.

Jeffrey Pennington, 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-2014)*, pages 1532–1543.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM-2018)*, pages 180–191.

Kyle Richardson, Hai Hu, Lawrence S. Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-2020)*.

Alexis Ross and Ellie Pavlick. 2019. How well do NLI models capture verb veridicality? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP-2019)*, pages 2230–2240.

Ohad Rozen, Vered Shwartz, Roee Aharoni, and Ido Dagan. 2019. Diversify your datasets: Analyzing generalization via controlled variance in adversarial datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL-2019)*, pages 196–205.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations (ICLR-2019)*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP-2018)*, pages 4731–4736.

Nam Khanh Tran and Weiwei Cheng. 2018. Multiplicative tree-structured long short-term memory networks for semantic representations. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM-2018)*, pages 276–286.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC-2018)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR-2019)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2018)*, pages 1112–1122.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019a. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019b. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 250–255.

A Appendix

A.1 Lexical entries and replacement examples

Table 6 shows a context-free grammar and a set of predicate replacements used to generate inference examples. Regarding the context-free grammar, we consider premise–hypothesis pairs containing the quantifier Q in the subject position, and the predicate replacement is performed in both the first and second arguments of the quantifier. When generating premise–hypothesis pairs involving embedding monotonicity, we consider inferences involving four types of predicate replacements (hyponyms N_{hypn}, adjectives Adj, prepositions PP, and relative clauses $RelC$) in the

Context-free grammar for premise sentences		
S	$\rightarrow$	$NP\ IV_1$
NP	$\rightarrow$	$Q\ N \mid Q\ N\ \overline{S}$
$\overline{S}$	$\rightarrow$	$WhNP\ TV\ NP \mid WhNP\ NP\ TV \mid NP\ TV$
Lexicon		
Q	$\rightarrow$	{*no, at most three, less than three, few, some, at least three, more than three, a few*}
N	$\rightarrow$	{*dog, rabbit, lion, cat, bear, tiger, elephant, fox, monkey, wolf*}
IV_1	$\rightarrow$	{*ran, walked, came, waltzed, swam, rushed, danced, dawdled, escaped, left*}
IV_2	$\rightarrow$	{*laughed, groaned, roared, screamed, cried*}
TV	$\rightarrow$	{*kissed, kicked, hit, cleaned, touched, loved, accepted, hurt, licked, followed*}
$WhNP$	$\rightarrow$	{*that, which*}
N_{hypn}	$\rightarrow$	{*animal, creature, mammal, beast*}
Adj	$\rightarrow$	{*small, large, crazy, polite, wild*}
PP	$\rightarrow$	{*in the area, on the ground, at the park, near the shore, around the island*}
$RelC$	$\rightarrow$	{*which ate dinner, that liked flowers, which hated the sun, that stayed up late*}
Adv	$\rightarrow$	{*slowly, quickly, seriously, suddenly, lazily*}
Predicate replacements for hypothesis sentences		
N	to	$N_{hypn} \mid Adj\ N \mid N\ PP \mid N\ RelC$
IV_1	to	$IV_1\ Adv \mid IV_1\ PP \mid IV_1$ or $IV_2 \mid IV_1$ and IV_2

Table 6: A context-free grammar and a set of predicate replacements used to generate inference examples. Predicate replacement is applied to N or IV_1, replacing it with a corresponding phrase.

first argument of the quantifier. To generate natural sentences consistently, we use the past tense for verbs; for lexical entries and predicate replacements, we select those that do not violate selectional restriction.

To check the gold labels for the generated premise–hypothesis pairs, we translate each sentence to a first-order logic (FOL) formula and test if the entailment relation holds by theorem proving. The FOL formulas are compositionally derived by combining lambda terms assigned to each lexical item in accordance with meaning composition rules specified in the CFG rules in the standard way (Blackburn and Bos, 2005). Since our purpose is to check the polarity of monotonicity marking, vague quantifiers such as *few* are represented according to their polarity. For example, we map the quantifier *few* onto the lambda-term $\lambda P \lambda Q \neg \exists x(\textbf{few}(x) \wedge P(x) \wedge Q(x))$.

A.2 Results on embedding monotonicity

Table 7 shows all results on embedding monotonicity. This indicates that all models partially generalize to inferences containing embedded clauses one level deeper than the training set, but fail to generalize to inferences containing embedded clauses two or more levels deeper.

Train	**Test**	**BERT**	**LSTM**	**TreeLSTM**
$\mathbf{D}_1$	$\mathbf{D}_1$	100.0±0.0	91.1±5.4	100.0±0.0
	$\mathbf{D}_2$	44.1±6.4	34.1±3.8	48.1±1.2
	$\mathbf{D}_3$	47.6±3.2	45.1±5.1	48.5±1.8
	$\mathbf{D}_4$	49.6±1.0	44.4±6.5	50.1±2.1
	$\mathbf{D}_5$	49.9±1.1	44.1±5.3	50.3±1.1
$\mathbf{D}_1 \cup \mathbf{D}_2$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.1
	$\mathbf{D}_2$	100.0±0.0	99.8±0.2	99.5±0.1
	$\mathbf{D}_3$	75.2±10.0	75.4±10.8	86.4±4.1
	$\mathbf{D}_4$	55.0±3.7	57.7±8.7	58.6±7.8
	$\mathbf{D}_5$	49.9±4.4	45.8±4.0	48.4±3.7
$\mathbf{D}_1 \cup \mathbf{D}_2 \cup \mathbf{D}_3$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.0
	$\mathbf{D}_2$	100.0±0.0	95.1±7.8	99.6±0.0
	$\mathbf{D}_3$	100.0±0.0	85.2±8.9	97.7±1.1
	$\mathbf{D}_4$	77.9±10.8	59.7±10.8	68±5.6
	$\mathbf{D}_5$	53.5±19.6	55.1±8.2	49.6±4.3
$\mathbf{D}_1 \cup \mathbf{D}_2 \cup \mathbf{D}_3 \cup \mathbf{D}_4$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.1
	$\mathbf{D}_2$	100.0±0.0	99.4±1.1	99.7±0.2
	$\mathbf{D}_3$	100.0±0.0	91.5±4.0	98.9±1.1
	$\mathbf{D}_4$	100.0±0.0	74.1±4.2	94.0±2.3
	$\mathbf{D}_5$	89.1±5.4	64.2±4.7	69.5±4.1
$\mathbf{D}_1 \cup \mathbf{D}_2 \cup \mathbf{D}_3 \cup \mathbf{D}_4 \cup \mathbf{D}_5$	$\mathbf{D}_1$	100.0±0.0	100.0±0.0	100.0±0.1
	$\mathbf{D}_2$	100.0±0.0	95.8±7.3	99.8±0.1
	$\mathbf{D}_3$	100.0±0.0	90.5±13.1	99.1±0.2
	$\mathbf{D}_4$	100.0±0.0	90.2±6.0	94.8±0.1
	$\mathbf{D}_5$	100.0±0.0	93.6±3.1	83.2±12.1
$\mathbf{D}_2$	$\mathbf{D}_1$	36.4±14.4	25.3±9.3	44.9±4.1
	$\mathbf{D}_2$	100.0±0.0	100.0±0.0	100.0±0.2
	$\mathbf{D}_3$	47.6±10.3	43.9±17.5	51.8±1.1
	$\mathbf{D}_4$	61.7±7.8	57.9±14.7	51.7±0.6
	$\mathbf{D}_5$	42.6±5.1	47.2±2.9	50.9±0.4
$\mathbf{D}_3$	$\mathbf{D}_1$	49.6±0.5	48.8±13.2	49.8±4.1
	$\mathbf{D}_2$	49.8±0.6	47.3±12.1	51.8±1.1
	$\mathbf{D}_3$	100.0±0.0	100.0±0.0	100.0±0.2
	$\mathbf{D}_4$	49.7±1.0	42.0±0.6	51.3±0.7
	$\mathbf{D}_5$	50.0±0.4	38.4±9.6	49.8±0.3
$\mathbf{D}_4$	$\mathbf{D}_1$	50.3±1.0	46.8±6.5	49.0±0.4
	$\mathbf{D}_2$	49.6±0.8	45.4±1.8	49.7±0.3
	$\mathbf{D}_3$	50.2±0.7	45.1±0.6	50.5±0.7
	$\mathbf{D}_4$	100.0±0.0	100.0±0.0	100.0±0.1
	$\mathbf{D}_5$	49.7±0.5	45.1±0.9	50.5±1.1
$\mathbf{D}_5$	$\mathbf{D}_1$	49.9±0.7	43.7±4.4	49.1±1.1
	$\mathbf{D}_2$	49.1±0.3	43.4±3.9	51.4±0.6
	$\mathbf{D}_3$	50.6±0.2	44.3±2.7	50.5±0.3
	$\mathbf{D}_4$	50.9±0.8	44.4±3.4	50.3±0.4
	$\mathbf{D}_5$	100.0±0.0	100.0±0.0	100.0±0.1

Table 7: All results on embedding monotonicity.