Latent Compositional Representations for English Function Word Comprehension

Megan Barnes

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Reading Committee:
Shane Steinert-Threlkeld, Chair
Fei Xia

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Latent Compositional Representations for English Function Word Comprehension

Megan Barnes

Chair of the Supervisory Committee:
Assistant Professor Shane Steinert-Threlkeld
Linguistics

This paper investigates whether biasing natural language models toward tree-compositional structure and systematic token representation can improve performance on tasks that require the use of function words. The method used treats tree-structure as latent and thus requires no gold parse labels. Results show that across four function-word-focused NLI probing tasks, tree-compositional models perform as well as LSTMs, but lag behind BERT to varying degrees between tasks. Context-dependent behavior of tree-compositional models highlights a potential weakness of the architecture in the absence of grounding information.

TABLE OF CONTENTS

Page

List of Figures ii

List of Tables iii

Chapter 1: Introduction and Related Work 1
1.1 Related Work 3
1.2 Contribution 7

Chapter 2: Methods 8
2.1 Function Word Probing Tasks 8
2.2 Training Procedure and Evaluation 9
2.3 Model Architecture 10
2.4 Models for Comparison 13

Chapter 3: Results 15

Chapter 4: Discussion 20

Chapter 5: Conclusion 23

Bibliography 24

LIST OF FIGURES

Figure Number Page

2.1 Diagram of the LT sentence encoder, based on a similar figure from [1]. Each gray box is a possible composition for the full span. All compositions, produced by composition function f_h are attended to to produce the final representation for the span, h_{ij}. 12

2.2 Diagram of token-level encoder. The contextualizing module, in gray, creates a representation of each word using an LSTM that takes the full sentence as input. The blue embedding is global and learned during training. Here i and j refer to split points around the word *not*, rather than around a composed phrase. 12

2.3 Comparison illustration of baseline model and latent tree model. Arrows between the LSTM Sentence Embedder and the LT Sentence Embedder show that one is swapped for the other between models. Function word embeddings refer to learned function word embeddings, which are added in addition to the contextual embeddings in the gray box. 13

3.1 Sample Sentence Structures: Premises 17

3.2 Sample Sentence Structures: Hypotheses 18

LIST OF TABLES

Table Number Page

2.1 Example data from NLI-style probing tasks 9

3.1 Accuracy on NLI-Style Probing Tasks. "LT" refers to this work's full proposed model architecture, where "LT (-FWE)" refers to a version of the proposed model architecture that only uses the LT sentence encoder and does not use global function word embeddings. 16

3.2 Accuracy on MNLI Validation Set 16

ACKNOWLEDGMENTS

Many thanks to my adviser, Shane Steinert-Threlkeld, for his tremendous support and guidance during the writing of this thesis. His mentorship has been invaluable and is the reason this work exists. I owe thanks to Fei Xia and the members of the CLMBR lab, for feedback on drafts and earlier stages of this research. I would also like to thank the faculty, staff, my classmates, and my cohort in the Linguistics department, for fostering an environment where supporting each other is paramount.

DEDICATION

to my friends and family

Chapter 1

INTRODUCTION AND RELATED WORK

Human language productions are thought of as largely compositional. This means that larger units of meaning, like phrases and sentences, are functions of the meanings of their components (words, sub-phrases) and the way in which they are combined (grammatic structure) [11]. In support of this idea, concrete examples like the following are often cited: an English speaker knows the meaning of *Jack runs* even if they have never heard the sentence before as long as they know the meaning of *Jack*, *run*, and how to express the subject of an intransitive verb [4]. Language is not an entirely compositional task, as there are myriad examples of composed meanings that are not entirely predictable from the meanings of their components (e.g. idioms, where phrase or sentence meaning cannot be derived from constituent meanings [4]). However, the principle of compositionality is core to the expressive capacity of language in the context of human cognition. The ability to maintain a consistent understanding of what a word means and how a grammar can combine the meaning with that of other words allows us to space-efficiently maximize the number of expressions we can create; we do not have to store the meaning of *run* in the context of every runner.

The property of maintaining consistent meanings for words across contexts is known as systematicity. Systems that can successfully interpret arbitrary combinations of known words or constituents are said to be making systematic generalizations. These stand in contrast to systems that are based upon contextually-conditioned meaning representations, in which sub-units can contribute variably to composed meanings dependent on their context. Clear examples of contextual meaning representations in systems are models like ELMo [19], BERT [3], and their conceptual offspring. These models have dominated natural language processing in recent years for nearly every shared task, sometimes achieving results that

exceed human accuracy. One of the major innovations that these types of models introduced is the capacity to dynamically create contextually-conditioned meaning representations that are capable of representing words' different senses and evolving usage. Even more traditional LSTMs allow information to flow between tokens, producing contextualized representations.

Despite the clear benefit that contextually-conditioned representations provide for many words, it is conceivable that there may be certain classes of words for which variability may be inappropriate. In English one possible example of this is the function word class. Although exhaustive lists of function words are not entirely agreed upon, function words are generally thought of as determiners, quantifiers, prepositions, negation markers, and other types of words that serve grammatical purposes rather than contributing conceptual meaning [5]. They stand in contrast to content words (e.g. nouns and verbs), which portray the concepts at issue. Function words are closed-class, meaning that once a language is formed speakers do not create new ones. This is not true of content words, which speech communities are constantly creating to talk about new concepts as they arise.

It is clear why a flexible representation of a verb like *run* may be useful — a system may find a use for representing it differently in the context of other words like *late* or *hot*. A question that contextualizing systems need to address, however, is how useful variable representations are for words like *all* and *no*. Many function words have logical meanings that are easily formalized and that we expect to be largely context-invariant. The systematicity of function word meaning seems to be at odds with highly contextualizing model architectures.

This suspicion is borne out in the first benchmark work focused explicitly on function word comprehension tasks, proposed by Kim et al. [7]. In baseline tests of contextualizing models like LSTMs (pretrained on a variety of tasks) and BERT, the models performed much more poorly on function-word-focused test sets than on standard held-out sets. Contextualizing models circumvent traditional notions of compositionality by allowing token representations to be conditioned on other tokens and by allowing composition of representations outside immediate adjacency. This work seeks to test whether model architectures with greater inductive bias toward systematicity and compositionality achieve better results on function

word comprehension tasks. Specifically, it uses new function-word probing tasks from [7] to investigate whether a sentence encoder which enforces strict tree-compositional latent structure, from Bogin et al. [1], rather than allowing free global contextualization, can better represent and use function words. The rest of Section 1 will describe the bodies of work on systematic generalization, function words, and compositional neural models, as well as how this work relates to them. Section 2 will focus on the novel model architecture being tested and the tasks used to evaluate its use of function words. Section 3 will describe the results of that evaluation and Section 4 will include discussion of the results and limitations of the approach.

1.1 Related Work

1.1.1 Systematic Generalization

The motivation for this investigation comes from a rich field of work exploring the limitations of neural models' ability to generalize systematically. From a theoretical perspective, studies like McCoy et al. [16] have shown that RNNs trained on string-manipulation tasks that require systematic generalization can create representations that implicitly encode compositional structure. They do this by showing that encodings produced by an RNN can be closely approximated by Tensor Product Representations [20]. TPRs are explicitly compositional, as they are essentially functions dependent only on consistent representations of words and their order. However, they also find that for models trained on naturally-occurring language data in many tasks, approximation using TPRs is achieved using a Bag of Words composition scheme, which completely disregards the structure of an utterance. Soulos et al.[21] confirm this finding, using a more sophisticated neural learned-structure module. Both studies suggest that, despite a capacity to generalize systematically, for reasons possibly related to the nature of shared language tasks, RNNs do not.

Lake and Baroni [12] also find mixed results for the systematic generalization ability of seq2seq RNNs. They propose the dataset, SCAN, which consists of pairs of natural language

instructions like *turn left twice* and their translation into sequences of actions (*LTURN LTURN*, in this case). They find that RNNs can generalize to held out sets that only require that they mix and match components — for example, if the training set contains "*jump opposite right after* turn opposite right" and "jump right twice *after walk around right thrice*", they can properly interpret the new command "*jump opposite right after walk around right thrice*". They cannot, however, systematically generalize at the word level. When presented with training sets in which a basic action is presented without modification (*jump*) and other basic actions are presented composed with other actions or modifiers (e.g. *run*, *run twice*, *walk*, *walk opposite left and run twice* [12]), RNNs are completely incapable of composing the unmodified action with other commands or modifiers.

The findings in [12] point to an inductive bias in RNNs that does not match that of human learners. Speakers tend to have an intuition that the word level is the basic level at which systematicity of meaning and composition applies. In other words, we have an inductive bias toward succeeding at the task at which the models in [12] fail. Humans' assumption that composed meaning is built from systematic meanings of atomic symbols (in English, words) additionally suggests an explanation for the difference in the amount of data required for humans and neural models to learn. Without basic assumptions about systematicity at the word level, neural models require far more data than a human learner to understand the meaning of larger, potentially multi-word atomic units of which the vocabulary space is intractibly large. Hupkes et al. [6] uncover similar patterns in their examination of systematicity in neural networks. They test LSTM seq2seq, convolutional seq2seq, and Transformer models on behavioral tests designed to test systematicity in a similar method to [12]. They find a similar lagging accuracy in the models' interpretation of novel combinations of words. They also find that when longer combinations of the same words a model has seen in training are used for testing, accuracy sharply declines. This leads them to the conclusion that no model architectures they tested are forming general-purpose understandings of words outside of their context. All detailed findings pointing to a general lack of systematicity in widely-used neural models motivate the design of this work's neural architecture, which is

intended to be biased toward systematic generalization.

1.1.2 Function Words

As previously mentioned, neural models forming context-sensitive representations could be an issue in function word usage tasks. Some work examining the capacity for systematic generalization in neural models has focused expressly on function words. Goodwin et al. [5] frame behavioral testing for systematicity in terms of handling quantifiers. Their novel dataset is formed using an artificial language in which quantifiers are combined with open-class words. They form sentence pairs from sentences like *All brown dogs that bark don't run* and label the pairs for their natural logic entailment value (in the style of NLI). They analyze LSTMs', GRUs', and convolutional neural networks' performance on behavioral tests. In one, they perturb the open-class content words in a model's correctly interpreted sentence pairs to see if the model maintains systematic representations of quantifiers (if so, it should be able to succeed after perturbation). In another, they test sentence pairs where content words are identical and the meaning of the quantifiers alone can be used to figure out the entailment value (e.g. *All dogs run* and *Some dogs don't run*). In both cases, all models' accuracies fall short of hold-out test accuracy, demonstrating that all models leverage representations of quantifiers that are highly sensitive to their context, often leading to erroneous generalizations.

Kim et al. [7] propose a suite of behavioral tests aimed at characterizing the accuracy of neural models on function word comprehension. This suite has a wide breadth, consisting of small tasks focusing on prepositions, negation, spatial words, quantifiers, comparatives, and more. They perform initial benchmark tests of LSTMs pretrained on a variety of language objectives, as well as BERT, and compare their performance on all function word challenge tasks. They find that in nearly all cases, LSTMs' accuracy on function-word-focused tests lags significantly behind their accuracy on held-out sets. BERT performs similarly on many tasks in the suite. [7] sets out a useful shared task suite for testing a model's ability to use and represent function words and paints the task as clearly unsolved by even state of the art

neural modeling techniques. The tasks will be described in more detail in Section 2.1.

1.1.3 Compositional Neural Models

In order to address the issue of function word representation and use in neural models, this work proposes that model architectures be biased toward compositionality and away from global contextualization. This type of bias was often explicit in pre-neural models for language tasks, which built in hierarchical structure as they formed representations of language inputs ([24], [14]). This fell out of fashion somewhat with the advance of neural models and preference toward models that are thought to have less inductive bias. Some research has focused on encouraging systematic and compositional inductive bias in neural models, however; most notably, with Tree LSTMs [22]. Tree LSTMs successfully extended LSTM unit chaining to tree-structured (rather than linear) inputs in order to compose final representations of sentences that could be used in downstream tasks. One downside of this type of model is that it relies on ground-truth tree structures for all of its inputs, which are often unavailable for many training sets and at inference. Moreover, it is unclear whether constituent or dependency structures (often designed for the theoretical study of syntax and semantics) are useful composition schemes for a model to achieve good results on a downstream task.

Bogin et al. [1] propose a model that encourages compositional structure using softer assumptions than the Tree LSTM. Addressing the task of visually-grounded question answering, they propose a model that uses attention to marginalize over all possible binary-branching tree structures for the input (ideally learning to heavily weight useful ones). This allows them to impose tree structure onto an input without positing a ground truth structure. Their goal is to eliminate global contextualization (and improve systematicity) in the model by only allowing composition of vector representations that are immediately adjacent within the latent structure. This model is the basis of the modeling solution proposed in this work and will be described in more detail in section 2.3.

1.2 Contribution

This work differs from related work on systematicity of neural models and their comprehension of function words in a number of ways. Unlike all of the studies described in section 1.1.1, this work does not seek to answer the question of whether neural models generalize systematically, but rather takes as an assumption (based on the prior research described) that modern neural architectures fail to do so in a satisfactory manner. Focus will instead be on applying one method of encouraging systematic and compositional bias in neural models and evaluating the value of that method. It differs from related work on function word comprehension in that it does not seek to propose new behavioral tests for models, but rather evaluates a modeling technique on existing tests. This work is closely related to other work on compositional neural models, especially [1], as it applies similar modeling techniques. The focus of that application will be on function word comprehension, however, instead of grounded question answering.

Chapter 2

METHODS

2.1 Function Word Probing Tasks

All data used to evaluate models' function word comprehension comes from [7]. In order to form the tasks in the suite, English sentences were extracted from existing corpora based on the presence of target function word types. The tasks are labeled following one of two formats: grammatical acceptibility judgements or natural language inference (NLI [15]) format. Acceptability judgment tasks are binary classification tasks in which a single sentence is labeled according to whether it is interpretable using English grammar. NLI tasks are ternary classification tasks in which the entailment relationship between two sentences forms the label for the pair (*entailment*, *contradiction*, or *neutral*). In general, all tasks were designed so that the label of each test instance depends on function words within the sentences. This work tests only NLI-format tasks in the suite, as entailment is theoretically more closely related to meaning than grammatical acceptability, which is more related to form[1].

The four tasks used for probing function word understanding in this work are negation, prepositions, quantification and spatial expressions tasks. The negation task tests understanding of *not* as well as lexical negation using antonyms. The prepositions task substitutes certain prepositions for others and tests whether the model understands how the entailment relationship changes. The quantification task tests understanding of entailment relationships between quantifiers like *all* and *some* in the presence of negation. The spatial expressions task probes the understanding of words like *left* and *right* also in the presence of negation. Examples of data from each task are given in Table 2.1 below. All task datasets are relatively

[1]Notable counterexamples to this generalization exist such as Negative Polarity Items, whose grammatical distribution is seemingly licensed by downward entailing semantic environments[2].

Table 2.1: Example data from NLI-style probing tasks

Set	Premise	Hypothesis	Relation
Negation	This is a common problem.	This is **not** an **un**common issue we are facing	Entailment
	This is **not** a common problem.	This is **not** an **un**common issue we are facing	Contradiction
Prep.	**With** a single jerk the man's head tore free.	The man's head tore free **from** a single jerk.	Entailment
	With a single jerk the man's head tore free.	The man's head tore free **without** a single jerk.	Contradiction
Quant.	There's **some**.	There is **none** at all.	Contradiction
	There's **some**.	There is not **none** at all.	Entailment
Spatial	Turn **left** up a small alleyway.	Do not turn **right** up the alleyway.	Entailment
	Turn **left** up a small alleyway.	Turn **right** up the alleyway.	Contradiction

small, ranging from 157 examples (spatial) to 591 examples (negation), with quantification and prepositions at 323 and 358 examples, respectively. They are based upon sentences containing function words drawn from MNLI [23] and targeted modifications to those sentences likely to produce different labels. Thus, all labels are similarly represented in the datasets. Each task example is labeled by three human annotators.

2.2 Training Procedure and Evaluation

In its initial proposal of the function word probing suite, [7] sought to test the impact of different pretraining tasks on function word test performance with a fixed model architecture. This paper instead fixes the pretraining task and tests the impact of different model architectures. MNLI was chosen as the pretraining objective for this study, as it is one of the strongest-performing pretraining objectives from [7] and closely related to the NLI probing tasks. The model architectures tested are trained on MNLI data and then evaluated directly on probing sets without any probe-specific training or fine-tuning.

The training procedure was kept largely the same as that of [7]. A learning rate of 1e-4 was used. Evaluation on the validation set every 1,000 iterations was done and training was stopped if there was no best result after 20 evaluations. The learning rate was multiplied

by 0.5 whenever validation performance failed to improve for more than 4 validation checks. The Adam [8] optimizer was used. The best checkpoints from training across five random seeds were then chosen for evaluation and results were averaged.

2.3 Model Architecture

In order to bias sentence representations toward compositionality, a latent tree (LT) sentence encoder based on the grounded latent tree (GLT) encoder from [1] is used. The LT encoder computes a representation for a sentence while simultaneously performing a bottom-up parse using a softened version of the CKY algorithm. Because the data has no ground-truth structural annotations, the model learns to induce latent tree structure for the sentence using only NLI labels, and therefore learns latent structure useful for the task, rather than structure that approximates syntactic parses. The GLT model is identical to the LT model used, except that it produces an additional vector at each composition step in the tree (taking the composed word representations at that level as input). This *denotation* vector represents the set of objects in an image that the span refers to (where the image constitutes the grounding in a grounded question answering task). Denotation vectors are composed using a more restricted set of primarily logical operations. More detailed description of this grounding mechanism can be found in Chapter 4.

By creating a tree structure for a sentence, the LT encoder enforces compositionality for each span in the sentence. That is, it is able to define the representation of any span (word or multi-word phrase) in the sentence using only the representations of the two sub-spans which constitute the span. If a span over words $\mathbf{w}_i$ to $\mathbf{w}_j$ is split into subspans at point k, we define its representation as $\mathbf{h}_{ij}$ and its subspan representations as $\mathbf{h}_{ik}$ and $\mathbf{h}_{kj}$. Given a composition function $f_h(\cdot)$:

$$\mathbf{h}_{ij}^k = f_h(\mathbf{h}_{ik}, \mathbf{h}_{kj}) \quad [1] \tag{1}$$

Rather than choosing particular split points (as in parsing algorithms like CKY), the model learns a weighted sum over all possible split points. If this weighting is thought of

as the probability of a given split point, this sum is the expected representation over split points:

$$\mathbf{h}_{ij} = \sum_k p_H(k|i,j) \cdot \mathbf{h}_{ij}^k = E_{p_H(k|\cdot)}[\mathbf{h}_{ij}^k] \quad [1] \tag{2}$$

The composition function, $f_h(\cdot)$, is meant to create a combined meaning representation for a span from only a span's two subspan representations. To do this, an attention mechanism is applied to the two subspans (Eq. 3-4) and then a feed-forward layer with a residual connection is applied (Eq. 5). A diagram of the latent tree sentence encoder can be found in Figure 2.1.

$$\alpha_{ij}^k = \text{softmax}\left([\mathbf{a}_L \mathbf{h}_{ik}, \mathbf{a}_R \mathbf{h}_{kj}]\right) \tag{3}$$

$$\hat{\mathbf{h}}_{ij}^k = \alpha_{(1)} W_L \mathbf{h}_{ik} + \alpha_{(2)} W_R \mathbf{h}_{kj} \tag{4}$$

$$f_h(\mathbf{h}_{ik}, \mathbf{h}_{kj}) = \text{FF}_{rep}(\hat{\mathbf{h}}_{ij}^k) + \hat{\mathbf{h}}_{ij}^k \quad [1] \tag{5}$$

Beyond the compositional encoder from [1], this work adds an additional embedding module to represent each word in order to enable both systematicity of function words while leveraging contextual representations of all words. The first embedding module, based on the benchmark model from [7], passes each token through a pre-trained CNN over characters from ELMo [19], then passes the output of that layer through a bidirectional LSTM, which creates representations contextualized based on the other words of the sentence. On top of the BiLSTM is a layer that computes dot-product attention between all pairs of tokens between the sentences; this layer contextualizes each token based on the tokens in its pair sentence. The contextualizing token embedding module can be seen in the gray shaded box in Figure 2.2. Its output is labeled as l_{ij} in the figure. In contrast, the systematic function word representation that is added in this work is a global word embedding (e_{ij} in Figure 2.2). All non-function words in the systematic embedding module are treated as unknown and set to zero. Function words are defined in a list from [18]. The two embeddings, l_{ij} and e_{ij}, are

Figure 2.1: Diagram of the LT sentence encoder, based on a similar figure from [1]. Each gray box is a possible composition for the full span. All compositions, produced by composition function f_h are attended to to produce the final representation for the span, h_{ij}.

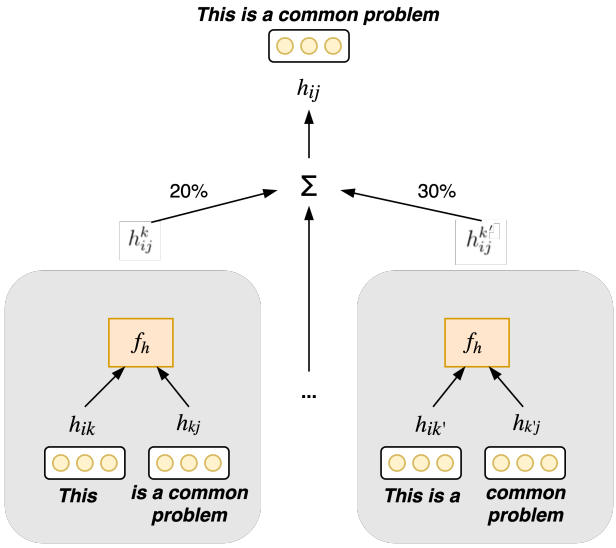


Figure 2.2: Diagram of token-level encoder. The contextualizing module, in gray, creates a representation of each word using an LSTM that takes the full sentence as input. The blue embedding is global and learned during training. Here i and j refer to split points around the word *not*, rather than around a composed phrase.

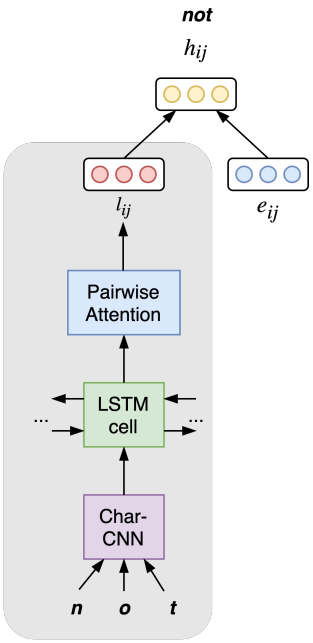
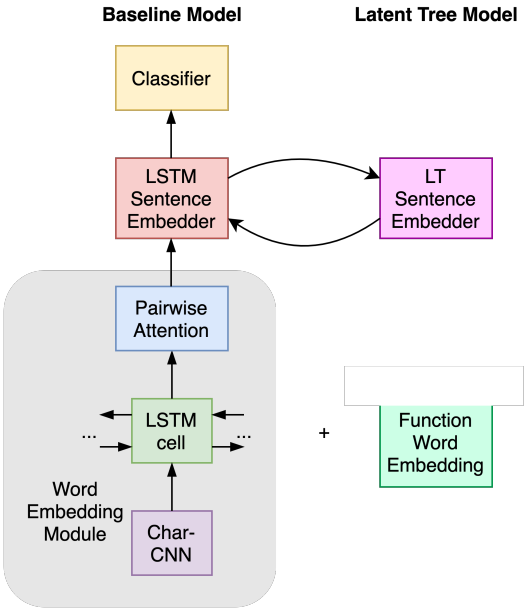


Figure 2.3: Comparison illustration of baseline model and latent tree model. Arrows between the LSTM Sentence Embedder and the LT Sentence Embedder show that one is swapped for the other between models. Function word embeddings refer to learned function word embeddings, which are added in addition to the contextual embeddings in the gray box.



combined by a learned linear weighting, which was empirically chosen based on comparison to other methods of combination, including alternation, addition, and attention.

2.4 Models for Comparison

The baseline model for comparison comes from [7] and varies primarily with respect to its sentence encoder and token encoder (as described in Section 2.3). A diagram of their full structure and differing components can be observed in Figure 2.3. Crucially, the baseline sentence encoder allows each word to be contextualized by every other word in the sentence, as well as every other word in the pair sentence at both word and sentence levels. The LT model we propose as the sentence encoder restricts contextualization to directly adjacent words or spans. At the token level, it introduces an additional strictly systematic embedding mode for function words that the model can learn to balance with contextualized representations.

Both models use the same classification layer which feeds a heuristic matching feature vector [17] of both sentence representations into a multilayer perceptron. Training hyperparameters are kept the same between both baseline and LT models and are described in Section 2.2. The performance of both models is also compared to BERT base uncased [3], fine-tuning on MNLI for 3 epochs with learning rate 2e-5. For comparison, the BERT model also benefits from pretraining, which the other models we evaluate do not. The baseline model contains around 81 million parameters, the LT model 60 million, and BERT 110 million.

Chapter 3

RESULTS

Table 3.1 shows the performance of the LT model on the four function word datasets. It provides a comparison to a naive 'Majority Class' baseline, where only labels of the majority class in the dataset are assigned. Additionally, results of an LSTM model and BERT are reported. The result of removing the fixed function word embeddings is also shown in order to isolate their effect from that of the compositional sentence encoder. Comparing these different models we find that in all cases the full LT model achieves a similar accuracy to the baseline LSTM model from [7] (all accuracies are within quoted standard deviations of models with different random seeds from [7]). Both perform significantly better than the majority class baseline and are outperformed by BERT in all tasks. Removing fixed function word embeddings and using only the compositional sentence encoder degrades accuracy on most tasks by 1-2%. Table 3.2 shows the models' accuracy on the MNLI validation set, in which the LT model outperforms the model from [7] by 2%. Both of these results suggest that the LT model, with much stronger inductive bias toward compositionality and systematicity, is a comparable encoder to LSTM-based models for tasks that prioritize function word understanding. It still does not meet the accuracy achieved by BERT even in tasks like the preposition dataset, however, where BERT only slightly exceeded 50% accuracy.

An artifact of the latent tree model that LSTM-based models do not provide is explicit description of the latent structure that the model uses for composition; it allows us to visualize which span split points the model assigns the highest weighting to. At a high level, we can compare these implicit parses to those generated by off-the-shelf parsers to see if they match theoretical notions of syntax. Comparing latent structures from the LT model to those generated by the Berkeley Parser [9] shows very little similarity. When comparing

Table 3.1: Accuracy on NLI-Style Probing Tasks. "LT" refers to this work's full proposed model architecture, where "LT (-FWE)" refers to a version of the proposed model architecture that only uses the LT sentence encoder and does not use global function word embeddings.

	Majority Class	Kim et al. (Baseline)	**LT**	LT (-FWE)	BERT	Human
Negation	0.40	0.59	0.59	0.59	0.65	0.80
Prep.	0.37	0.50	0.49	0.48	0.52	0.77
Quant.	0.48	0.65	0.66	0.64	0.78	0.87
Spatial	0.46	0.62	0.61	0.60	0.76	0.86

Table 3.2: Accuracy on MNLI Validation Set

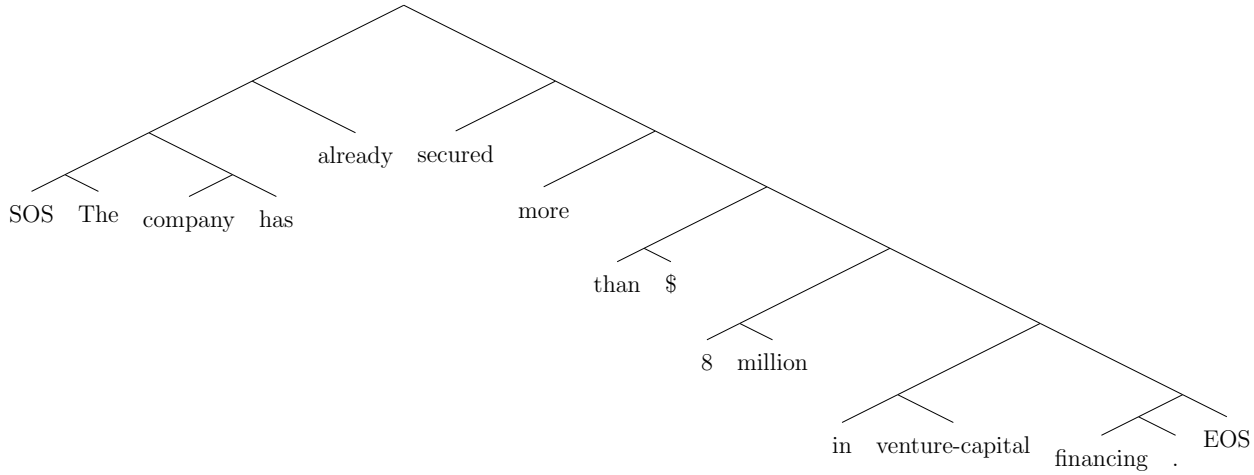
Kim et al.	**LT**
0.74	0.76

the percentage of non-leaf subtrees of all parses that were the same between single models with different random seeds and the Berkeley Parser, percentages ranged from 3.5-6% within the one test set. Further, increased matching subtree percentage does not correlate with increased accuracy on the test set between these models, meaning that models that better capture accepted syntactic structure have no distinct advantage in accomplishing the NLI task.

Since the structures produced by the LT model bear little resemblance to treebank-style syntactic structures, another result to examine is the form the structures take. Some examples are shown in Figures 3.3 and 3.4. The two pairs shown are similar pairs, where the premises are the exact same sentence and the hypotheses differ in that one sentence is negated. The structures produced are not necessarily interpretable from a syntactic or semantic standpoint and do not follow any obvious patterns. One phenomenon that emerges,

Figure 3.1: Sample Sentence Structures: Premises

Premise 1:



Premise 2:

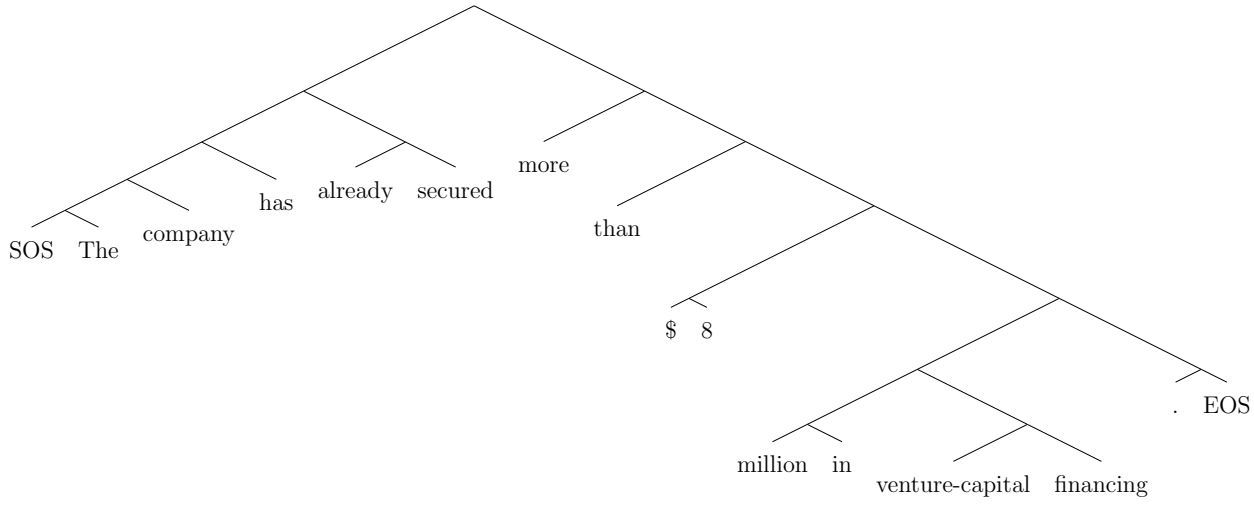
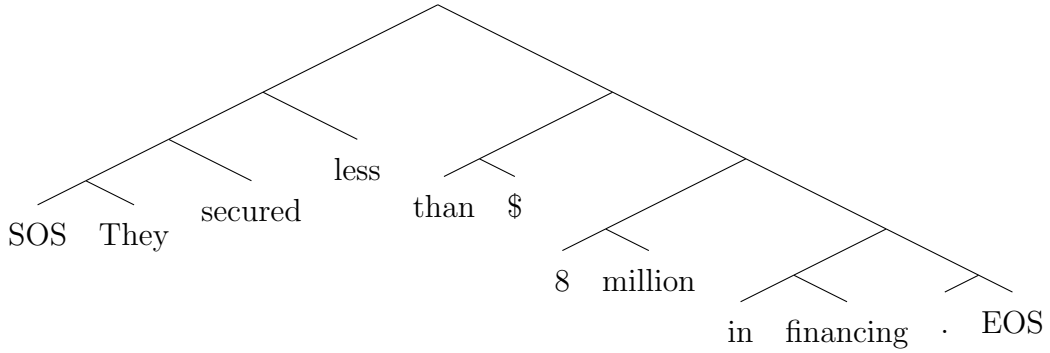
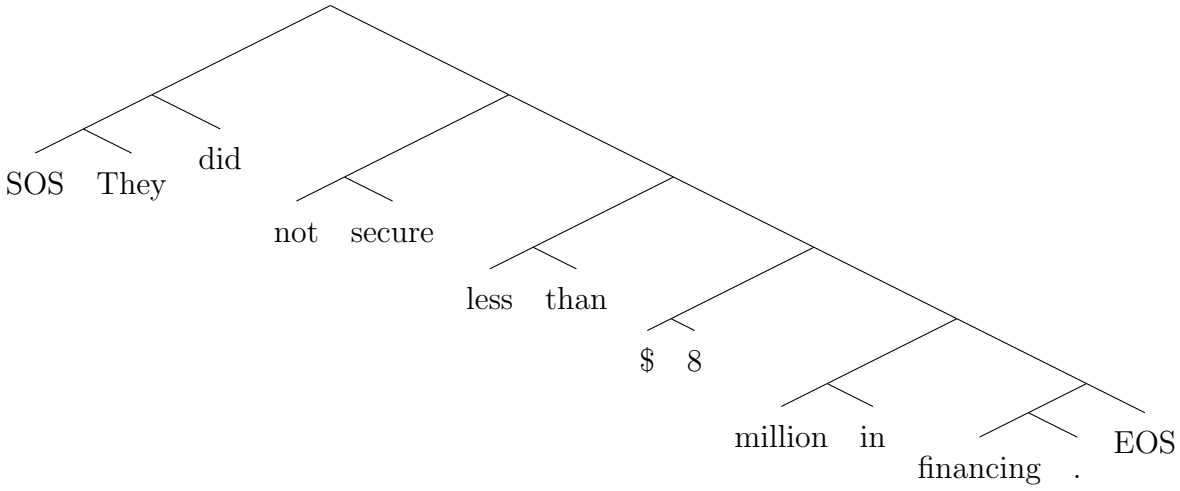


Figure 3.2: Sample Sentence Structures: Hypotheses

Hypothesis 1:



Hypothesis 2:



however, is that the structures of the same sentence are sensitive to changes in their pair sentence. We can see this in the diagrams of each of the premises in Figure 3.1 — despite consisting of the same tokens, the sentences are composed differently by the model due to their differing pair sentences (recall that each token representation in the contextualizing word embedding module is able to influenced by pair sentence tokens). Similarly, the hypothesis sentences shown in Figure 3.2 do not exhibit similar structure, despite differing only by negation. Similar variation is seen between models trained with different random seeds. In general, the LT models trained in this study did not converge on any consistent composition strategies for the same sentence between models or pair contexts.

Chapter 4

DISCUSSION

The accuracies of the LT model on function word tasks, listed in Table 3.1, support the idea that the model architecture proposed in this work is at least as effective at representing and using function words as standard LSTM architectures. Both facets of the model that differ from the baseline LSTM are worth evaluating for their contribution to accuracy on the tasks. The learned function word embedding was introduced in order to give the model access to non-contextual (and therefore systematic) representations of function words, in addition to contextualized representations of all words. The tree-structured sentence encoder was introduced in order to limit contextualization of the word representations to immediately adjacent words and phrases during composition. The results in Table 3.1 show that the latent tree sentence encoder was not effective on its own and required global function word embeddings in order to reach the accuracy of the LSTM baseline. The idea that adding the LT sentence encoder to a baseline model could degrade performance rather than add any incremental value on its own calls into question the utility of the compositional encoder architecture for this type of task.

Another problem with the LT model in the context of function word tasks is that its actual behavior is directly at odds with the linguistic principles it is aimed at encouraging. Although it is successful at enforcing the kind of local compositionality that we expect from language, it is not successful at inducing compositional rules governing its behavior that are consistent or context-independent. The idea that the model may produce different composition weightings for the same tokens in the context of different NLI pair sentences (as illustrated in the example in Figure 3.1) is directly at odds with the aim of this work. By enforcing compositionality in the sentence encoder architecture, we have still achieved a

model behavior that is context-dependent and, therefore, not systematic. This may explain why accuracy improvements on function-word-focused tasks from the LT model are marginal.

The inconsistency of composition that the LT model exhibits can potentially be attributed to the type of task it is trained to do. The latent tree model in this work is adapted from the Grounded Latent Tree model in [1]. Bogin et al. [1] successfully apply the GLT model to a grounded question answering task in which the model has access to information about objects in an image that it is using natural language to answer questions about. Unlike the LT model, at each step in composition the GLT model is required to produce a *denotation*, or a vector representing the set of objects in the image that the span refers to. Although there is no supervision at the span level for these denotations, there is an added inductive bias in the GLT model from this grounding strategy that the LT model does not have: composed word representations at the span level ($\mathbf{h}_{ij}$) are used as input to produce denotations. The denotations are composed through a more constrained logical system that consists of weighted combinations of intersection, union, 'pick one', and 'introduce a new object' over the sets of objects that the denotations represent. Unlike the setup in this work, where composition is only constrained by the structural assumptions of the LT encoder and the loss optimization of the task, the GLT model is constrained by an additional goal. The GLT model's compositional strategy must allow the model to create denotation vectors that can produce answers to questions using simple logical operations. Under these conditions, consistency of composition may become more crucial, as poor compositions can cause loss of information (e.g. removing objects) that may not be recoverable in later layers. The majority of shared NLP tasks, including challenging and carefully designed datasets like MNLI and the function word probing suite used in this study, still do not incorporate grounding outside of text, however, and thus the use of models like GLT to address them is not possible.

Finally, the accuracy of the LT model should be compared to BERT. This work conceptualizes the different sentence encoders at issue as existing on a spectrum, where the LT model gives the most restricted access to global contextual information, the LSTM gives more, and BERT allows token information to flow relatively freely anywhere in a sentence

and is, therefore, the least systematic. While it is true that BERT's accuracy lags significantly behind human accuracy in all the tasks examined, it still far exceeds the accuracy of all other models under comparison. The assumption based on its architecture that BERT is the most freely-contextualizing model, however, may be flawed. In examinations of the actual attention patterns that BERT learns when fine-tuned on various shared tasks, Kovaleva et al. [10] find that many attention heads learn highly consistent patterns of attention in which each token attends chiefly to the tokens immediately surrounding it. This bears out in analysis of attention that BERT exhibits on the function word probing suite, in which around 13% of attention heads exhibit this 'diagonal' behavior. The percentage of attention heads does vary significantly based on the fine-tuning task, however. It seems that at present transformer models consistently learn bias toward compositionality and systematicity from the data, to varying degrees and dictated by the task they are fine-tuned on. The caveat, as mentioned previously, is that they remain data-hungry and computationally intensive to train.

Chapter 5

CONCLUSION

This paper explores the affects of encouraging bias toward systematicity in models on tasks that require use of function words. This is achieved by imposing induced tree-compositional structure on sentence encoders and introducing context-independent token representations of function words. Across four function-word-focused NLI probing tasks, the model proposed achieves parity with LSTM-based architectures, but continues to lag behind BERT. This result suggests that bias toward systematicity in model architecture may hold less promise for improved function word comprehension than enrichment of task type and data.

BIBLIOGRAPHY

[1] Ben Bogin, Sanjay Subramanian, Matt Gardner, and Jonathan Berant. Latent compositional representations improve systematic generalization in grounded question answering. *TACL 2020*, 2020.

[2] Gennaro Chierchia. *Logic in grammar: Polarity, free choice, and intervention*. OUP Oxford, 2013.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

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

[5] Emily Goodwin, Koustuv Sinha, and Timothy J. O'Donnell. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online, July 2020. Association for Computational Linguistics.

[6] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 2020.

[7] Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[10] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*, 2019.

[11] Angelika Kratzer and Irene Heim. *Semantics in generative grammar*. Oxford: Blackwell, 1998.

[12] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

[13] Brenden M Lake, Tal Linzen, and Marco Baroni. Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*, 2019.

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

[15] Bill MacCartney and Christopher D Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, 2008.

[16] R Thomas Mccoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. Rnns implicitly implement tensor product representations. In *ICLR 2019-International Conference on Learning Representations*, 2019.

[17] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany, August 2016. Association for Computational Linguistics.

[18] James O'Shea, Zuhair Bandar, and Keeley Crockett. A machine learning approach to speech act classification using function words. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 82–91. Springer, 2010.

[19] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[20] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

[21] Paul Soulos, R Thomas McCoy, Tal Linzen, and Paul Smolensky. Discovering the compositional structure of vector representations with role learning networks. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 238–254, 2020.

[22] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

[23] Adina Williams, Nikita Nangia, and Samuel Bowman. 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, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[24] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666, 2005.