Structural Guidance for Transformer Language Models

Peng Qian[1] **Tahira Naseem**[2] **Roger Levy**[1] **Ramón Fernandez Astudillo**[2]
[1] Department of Brain and Cognitive Sciences, MIT [2] IBM Research
pqian@mit.edu tnaseem@us.ibm.com
rplevy@mit.edu ramon.astudillo@ibm.com

Abstract

Transformer-based language models pre-trained on large amounts of text data have proven remarkably successful in learning generic transferable linguistic representations. Here we study whether structural guidance leads to more human-like systematic linguistic generalization in Transformer language models without resorting to pre-training on very large amounts of data. We explore two general ideas. The "Generative Parsing" idea jointly models the incremental parse and word sequence as part of the same sequence modeling task. The "Structural Scaffold" idea guides the language model's representation via additional structure loss that separately predicts the incremental constituency parse. We train the proposed models along with a vanilla Transformer language model baseline on a 14 million-token and a 46 million-token subset of the BLLIP dataset, and evaluate models' syntactic generalization performances on SG Test Suites and sized BLiMP. Experiment results across two benchmarks suggest converging evidence that generative structural supervisions can induce more robust and humanlike linguistic generalization in Transformer language models without the need for data intensive pre-training.

1 Introduction

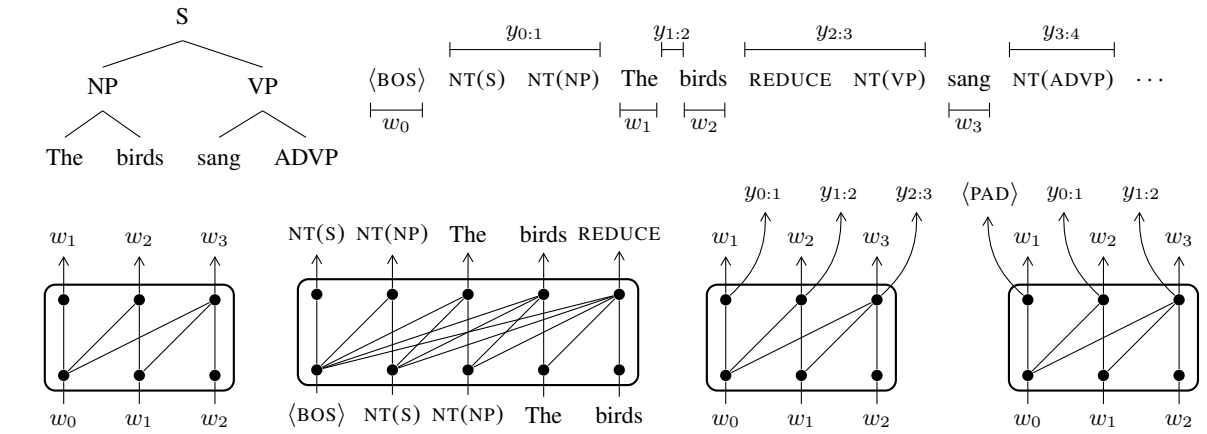
Pre-trained Transformer architectures have led to huge progress in building more human-like language processing systems (Radford et al.; Devlin et al., 2019; Brown et al., 2020, among others). These models achieve impressive perplexity results on language modelling datasets, perform well on grammatical judgments (Warstadt et al., 2020), and provide useful linguistic representations that benefit a wide range of downstream tasks. Probing analyses also suggest that these models learn to implicitly encode syntactic information (Hewitt and Manning, 2019; Clark et al., 2019) that may support better linguistic generalization than recurrent neural network architectures (RNNs).

However, the Transformer architecture (Vaswani et al., 2017) is an interesting subject of study beyond its success in transfer-learning settings. Transformer models lack the inductive biases of RNNs. Rather than maintaining vector-valued state and updating it in a recurrent manner, auto-regressive Transformer models encode all past decisions simultaneously at each inference step, thanks to a self-attention mechanism. The only notion of sequence order is also given by position embeddings summed to content embeddings in both input and auto-regressive signals.

Previous works have shown the advantage of structural supervision in RNNs in learning to maintain syntactic states and non-local dependencies (Dyer et al., 2016; Wilcox et al., 2019; Futrell et al., 2019). It remains an open question whether Transformer language models can similarly benefit from generative structural supervision, and what form of structural supervision would more effectively induce human-like syntactic generalization.

This work hypothesizes that the Transformer language model may benefit from explicit generative structural supervision to systematically generalize syntactic knowledge. Here we explore two major classes of structural guidance for Transformer language models based on joint modeling of language and constituency parses. The "generative parsing as language modeling" approach builds a Transformer-parameterized model to learn to predict actions that incrementally build constituency trees along with terminal words, following prior work on RNNs (Dyer et al., 2016; Choe and Charniak, 2016). The "structural scaffolding" approach follows the general idea of regularizing hidden representation through multi-task learning objective, with prior success in various NLP tasks (Zhang

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 3735–3745
August 1–6, 2021.



(a) Vanilla language model (b) Parsing as Language Modelling (c) Language models with Structural Scaffold

Figure 1: Top: Illustration of a partial constituency tree and corresponding transitions. Bottom: unidirectional transformer language model (a) without explicit structural supervision, (b) for modelling generative action parsing sequence, and (c) with structural scaffold for predicting the local incremental parsing state.

and Weiss, 2016; Søgaard and Goldberg, 2016; Swayamdipta et al., 2018).

We test these two approaches on two subsets of the BLLIP dataset (Charniak et al., 2000) and evaluate models' syntactic generalization performances on SG Test Suites (Hu et al., 2020) and a sampled subset of the BLiMP Benchmark (Warstadt et al., 2020). We show evidence that generative structural supervision indeed induces more robust and human-like linguistic generalization in Transformer language models and explore the different trade-offs involved in the presented methods.

2 Models

Here we explore joint modelling of structures and words parametrized with Transformers by considering both a sentence W and its constituency parse Y and modeling the joint distribution $P(W, Y)$.

2.1 Generative Parsing as Language Modeling

A language model can be described formally as a probability distribution over strings of a language $w_1, \cdots, w_T$, usually left-to-right factored.

$$p(W) = p(w_1, \cdots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{<t}) \quad (1)$$

There are many possible approaches that can combine both language modeling and syntax modeling tasks. As long as both tasks share some of the parameters they can be considered a case of multi-task learning (Caruana, 1997). Of interest here is the model proposed in Recurrent Neural Network Grammars (RNNGs; Dyer et al., 2016) and parsing as language model (LSTM-LM; Choe and Charniak, 2016). Both approaches model the joint distribution of words W and constituency tree components Y as

$$p(Y, W) = p(a_1, \cdots, a_R) = \prod_{t=1}^{R} p(a_t \mid a_{<t}) \quad (2)$$

where a_t are transitions of a state machine that generates both the sentence and the tree. These transitions are similar to the well-established transition sets used for transition-based parsing (Earley, 1970) but adapted to generate both text and parse simultaneously. For the reminder of this work, we will consider each a_t to be integer valued and indexing a dictionary of transitions. A transition a can be a word w or a transition action that generates a component of the constituency tree y. The actions include non-terminal symbols that open and label a new constituent with the label x, indicated as NT(x), or a REDUCE action closing the closest open constituent. An example of a partial parse tree and transitions can be found at the top of Figure 1.

RNNG and LSTM-LM parametrize the same factorization in Equation 2 in different ways. RNNG utilizes stack-LSTMs, which allow it to dynamically create representations for partial tree components by composition. The LSTM-LM, however, uses a flat parametrization treating the transitions as a sequence in a conventional language model learnt with an LSTM (Hochreiter and Schmidhuber, 1997). It should also be noted that the LSTM-

LM is designed as a parser, while RNNG is also used as a language model. In order to derive a language model from a joint model, it is is necessary to marginalize over all possible parse trees

$$p(W) = \sum_{Y \in \mathcal{Y}(W)} p(Y, W) \tag{3}$$

which is an intractable problem since there is an exponentially large number of possible trees. The original RNNG work (Dyer et al., 2016) proposes an approximate solution based on importance sampling. In this work we use the word-synchronous beam search approximation introduced in Stern et al. (2017).

The marginalized likelihood language model in Equation 3 is desirable because it makes no statistical independence assumption between language and syntax and shares all parameters across both tasks, with the exception of action specific embeddings. Particularly relevant for this work is the fact that both word and non-word transitions are predicted as language model output indiscriminately and are available at each prediction step through its history $a_{<t}$.

In this work we propose to parametrize Eq 2 with a Transformer language model (Vaswani et al., 2017). This is equivalent to the flat parametrization of the LSTM-LM but using a Transformer language model instead. Unlike LSTM-LM, which is a parsing model, we derive from it a language model by marginalization as in the RNNG. A Transformer language model can be succinctly described as a neural network of vertically stacked layers where the m-th layer is given by

$$h^m_{<t} = \mathrm{FF}^m\left(O \cdot \begin{bmatrix} \mathrm{A}^m_1(h^{m-1}_{<t}) \\ \mathrm{A}^m_2(h^{m-1}_{<t}) \\ \cdots \\ \mathrm{A}^m_N(h^{m-1}_{<t}) \end{bmatrix}\right). \tag{4}$$

Here $h^{m-1}_{<t} \in \mathbb{R}^{H \times t}$ is the output of the previous decoder layer for all previous predictions of the model at time step t and H is the size of the hidden vector. The input to the first layer i.e. $h^0_{<t}$ are the embeddings of all previous transitions $a_{<t}$ concatenated with a start symbol. Each embedding is the sum of both a content embedding, dictionary vector that is being indexed, and a position embedding that encodes the absolute or relative position of each action in the sequence.

$\mathrm{FF}^m()$ is a feed-forward layer, $A^m_1() \cdots A^M_N()$ are multiple self-attention heads and $O \in \mathbb{R}^{H \times H}$ is a matrix multiplication performed on the concatenated output of the attention heads. Both the feed-forward and the projection of N attention heads through O are wrapped around with residual, dropout and layer normalization operations that are here removed for clarity.

Each attention head comprises a simple inner product attention mechanism

$$\mathrm{A}^m_n(h^{m-1}_{<t}) = V^m_n \cdot h^{m-1}_{<t} \cdot \mathrm{softmax}\big((K^m_n \cdot h^{m-1}_{<t})^T \cdot Q^m_n \cdot h^{m-1}_{<t} + \mathcal{M}\big) \tag{5}$$

where $V^m_n, K^m_n, Q^m_n \in \mathbb{R}^{H/N \times H}$ are value, key and query projection matrices respectively and the softmax operation is normalized over columns to sum to one. The matrix $\mathcal{M} \in \{-\infty, 0\}^{t \times t}$ is used to prevent the model from attending to future states during training, enabling efficient parallelization. It is displayed here due to its relevance for the next section.

Similarly to other models, to derive a distribution over all possible transitions, including words, non-terminal symbols and the REDUCE operation, we can use a softmax together with an inner product

$$p(a_t \mid a_{<t}) = \mathrm{softmax}(E^{W \cup Y} \cdot h^m_{<t})_{a_t} \tag{6}$$

where $E^{W \cup Y}$ are the embeddings for the joint vocabulary of words, non-terminals and REDUCE transitions. Henceforth, we refer to this model as **Parsing as Language Model**, or **PLM** for short.

Unlike LSTMs or the RNNG, the Transformer has direct access to all past decisions through self-attention and relies on position embeddings to encode word order. Thus, in principle, there is no structural bias for the model to favor past decisions that are close in time to inform current prediction. On one hand, this potential ability to use long distance information can enable a less local, more human like processing of language, but on the other hand, it can also result in an additional learning burden, especially if there is not sufficient learning data available. Also worth noting for the experiments proposed here is that the total number of parameters of a typical Transformer greatly exceeds that of an LSTM or a RNNG model.

2.2 Incorporating RNNG-like characteristics

As previously mentioned, unlike any of the other models, the RNNG is able to create partial tree representations by composition using stack-LSTMs.

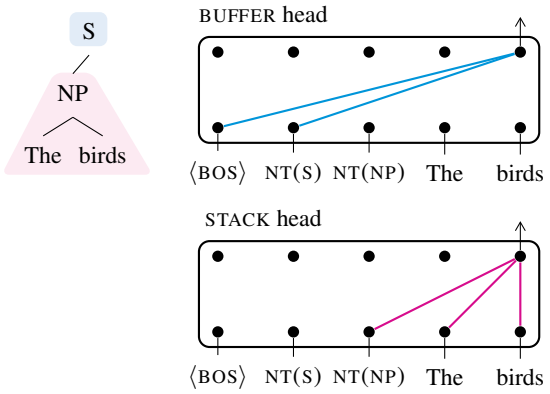


Figure 2: Illustration of how the generated incremental constituency parse is used to inform attention patterns in the structure-guided attention heads.

This changes the RNNG model structure dynamically as a function of the partial parse, a very desirable property to derive syntax-aware representations. Moreover, the fact that Recurrent Neural Networks such as LSTMs summarize all information about previous time steps on two hidden vectors, creates a bottleneck that forces the model to focus on the local state. This is a situation where a syntax-aware representation can provide additional value by enabling the local state to better encompass past structures. We conjecture that a similarly constrained local state might benefit Transformer models in learning linguistic regularities, especially in a limited training data scenario.

In an attempt to capture a similar effect in the Transformer, we explore here the idea of masking some attention heads to reflect the parser state as in the stack-Transformer (Astudillo et al., 2020). In the stack-Transformer, two attention heads are specialized to attend only to the contents of buffer and stack respectively for dependency and semantic parsing tasks. Here we choose to specialize two heads as well for each layer in Equation 4, as depicted in Fig. 2. One attention head attends to the contents of the last open constituent whereas another head attends all other past decisions not involving that constituent. The rest of the heads are left free as in the original Transformer architecture. To constrain the attention heads, we only need to alter the mask $\mathcal{M}$ in Equation 5 to depend on head index n and past actions $\mathcal{M}_n(a_{<t})$, which results in a negligible computation overhead.

This hard masking makes the model structure change dynamically depending on the partial parse and it forces some heads to focus on the local syntactic state. Nevertheless, unlike the RNNG, it does not create new representations of partial parses that can be composed in a recurrent manner at each time step, and some attention heads can still operate unrestricted. We hypothesize that structure-aware attention mechanism may still help the model achieve better generalization. The symbolic representation induces a strong inductive bias to how the model should use the structure that it generates on the fly. We henceforth refer to this model **PLM-mask**.

2.3 Scaffolding by Learning to Predict Local Parse States

Given the strong coupling between the tasks, the marginal likelihood Transformer language model of the previous section can be expected to be strongly influenced by the additional syntax prediction task. This comes however at a big cost. First, sequences combine both words and non-terminal and reduce transitions, yielding longer sentences than those of a normal language model $R > T$. Furthermore the approximated marginalization is computationally intensive and also introduces an approximation error.

One well-established regime that allows joint modeling of tasks at a low complexity is that of the syntactic scaffold (Zhang and Weiss, 2016; Søgaard and Goldberg, 2016; Swayamdipta et al., 2018). Scaffolding adds an additional structure prediction task at one of the layers of the model as a separate layer and only during training. This is a minimally intrusive change since it just branches some hidden vector of the network and computes an additional loss. It also has no influence on test runtime and avoids expensive steps such as marginalization.

However, applying the idea of syntactic scaffolding to our present scenario poses one difficulty. If we use a standard language model predicting words w and predict the non-word symbols y separately, we face the problem that the two sequences have different lengths. To overcome this in a straightforward way, we predict the n-gram of non-word actions $y_{t:t+n(t)}$ corresponding to the partial parse synchronous with step t when we predict word w_t. We use a secondary softmax layer for this action n-gram prediction.

$$p(y_{t:t+n} \mid y_{<t}) = \text{softmax}(E^{Y^*} \cdot h^m_{<t})_{y_{t:t+n}} \quad (7)$$

Here E^{Y^*} is the vocabulary of all transition n-grams excluding words found in the train corpus plus a blank symbol. Note that since Scaffolding

operates only at train time, we do not need to worry about generalization of these n-grams to test time.

The models are thus trained to minimize the loss function $-\log p(Y, W)$ where

$$\begin{aligned} p(Y, W) \quad &= \prod_{t=1}^{T} p(w_t \mid w_{<t}) \\ &+ \prod_{t=1}^{T} p(y_{t:t+n(t)} \mid w_{<t}) \end{aligned} \tag{8}$$

The scaffold can be set so that the synchronous non-word action n-grams $y_{t:t+n(t)}$ are predicted either before (Figure 1c, left) or after (Figure 1c, right) producing w_t. We considered both variants in our experiments to empirically assess their impact on performance. We refer to this model as **Transformer Language Model with Syntactic Scaffold**, or **ScLM** in short, and its two versions **ScLM-past** and **ScLM-next**, for past and next n-gram prediction.

3 Experiments

3.1 Model Training

All models, including the baseline vanilla language models (**LM** in short), the syntactic scaffold models, and the generative parsing models, are based on the same architecture of GPT-2 small (Radford et al.) (117M parameters, 12 layers, $H = 768$) and use the same BPE tokenizer, but with randomly initialized weights. We believe this would give us a fair comparison to pretrained GPT-2 as well, in order to evaluate whether structural guidance helps improve sample efficiency. We implemented all the proposed models using Huggingface's Transformer package (Wolf et al., 2020)[1].

As our goal here is to study whether structural guidance helps models learn robust humanlike generalization of syntactic knowledge, we train our model on the BLLIP dataset (Charniak et al., 2000), an English newswire style corpus used in Hu et al. (2020). This makes the results here more comparable to the results reported in previous work, especially with RNNGs. We train the proposed models and the baseline vanilla Transformer language models on BLLIP-MD, a 14 million-token corpus, and BLLIP-LG, a 46 million-token corpus, both of which are auto-parsed using a state-of-the-art constituency parser (Kitaev and Klein, 2018). We used the parsed sentences to generate oracle parsing action sequence for PLM and PLM-mask. We collected a list of word-synchronous parsing action sequences from the train and development oracle of BLLIP-LG and use it to parametrize the action n-gram vocabulary of ScLMs trained on both BLLIP-MD and BLLIP-LG. There are 3756 action n-gram types from the corpora, including one padding token and one blank token.

All models were trained with learning rate 10^{-5}, AdamW optimizer, and minibatch of size 5. We trained the models with multiple seeds within the capacity of our resources, in order to accommodate potential variance. In total, there are three seeds of LM, four of ScLM-past, four of ScLM-next, three of PLM, and three of PLM-mask for BLLIP-MD, and the same number of seeds of each model type for BLLIP-LG. Models were trained until convergence, as suggested by the loss of the development set during training.

3.2 Targeted Syntactic Evaluation

To assess whether a trained model systematically generalizes its syntactic knowledge, we employ targeted syntactic evaluation paradigm (Marvin and Linzen, 2018). Specifically, we measure models' performance on two held-out test datasets, a collection of syntactic generalization test suites from Hu et al. (2020) and BLiMP Benchmark from Warstadt et al. (2020). These two datasets cover a wide range of English syntactic phenomena.

Tests from Hu et al. (2020), which we refer as **SG Test Suites**, consist of hand-designed test suites for evaluating fine-grained syntactic generalization in incremental processing of a linguistic input. The general method is to compare models' surprisals $p(\text{continuation}|\text{prefix})$ of grammatical and ungrammatical continuations given certain sentence prefixes. We report the accuracy averaged across SG test suites. BLiMP Benchmark features minimal pairs of a grammatical sentence W and an ungrammatical counterpart W^*. To evaluate a model on these minimal pairs, one simply compares the likelihood of W and W^* assigned by the model.

As is implied by the evaluation methods, we need to marginalize out the structure variables for PLM or PLM-mask models in order to estimate the surprisal of a continuation, given a sentence prefix or the likelihood of a complete sentence. We follow similar setup as in Futrell et al. (2019); Wilcox et al. (2019) applying word-synchronous beam search (Stern et al., 2017) to find a list Y_k of k incremental parses given a sentence prefix $w_{<t}$.

[1]Code available at `https://github.com/IBM/transformers-struct-guidance`

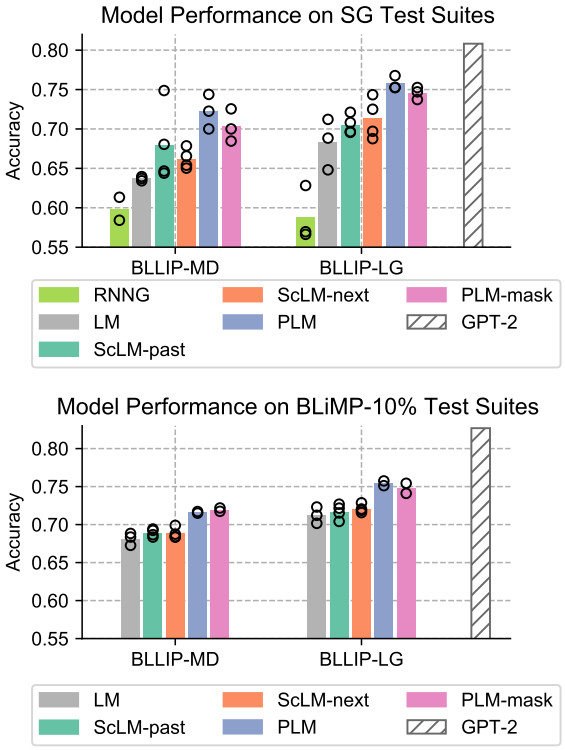


Figure 3: Comparing models' overall accuracy across test suites from SG Test Suites (top) and BLiMP-10% (bottom). RNNG performances are from Hu et al. (2020).

We then sum the joint probability $p(w_{<t}, y_{<t})$ over the list of incremental parses given by the model to approximate the likelihood of $p(w_{<t})$. We set the parse beam size to 100, word-synchronous beam size k as 10, and fast track size of 5. Since the search process can be computationally intensive, the large number of items in BLiMP benchmark poses a computational challenge. We therefore select the first 10% out of the 1000 items in each of the 67 tests of BLiMP Benchmark. We report the accuracy over the 100 items and refer to this down-sized BLiMP Benchmark as **BLiMP-10%**.

We compare models' performance on the SG Test Suites and BLiMP-10% in Figure 3. Each bar shows a model's performance averaged across multiple seeds on a given benchmark, with each dot plotting the accuracy of a specific seed. Overall, syntactic generalization performance improves as the training data size increases from BLLIP-MD (14 million tokens) to BLLIP-LG (42 million tokens). Models with structural guidance achieve higher accuracy than the vanilla Transformer language model trained on the same set of raw text data without explicit structural information. We also include the results for the RNNGs taken from Hu et al. (2020). RNNG lags behind all Transformer models by a large margin in average scores. We also notice that among different forms of structural guidance, generative parsing as language modeling is the most effective in improving syntactic generalization performance against the baseline transformer language models. We didn't observe additional benefits of adding dynamic masking mechanism to PLM. While scaffolding approach slightly improves vanilla Transformer language models, it still falls behind the best performance of the model trained with generative parsing. We hypothesize that our scaffold did not fully exploit the compositional structure in the local parses by modelling each action n-gram as a distinct type, while the generative parsing models only predict actions in a relatively small set of non-terminal action space, which might make it easier for PLM and PLM-mask to learn compositional generalization. We leave it for future work to design new scaffolds that can take advantage of the combinatorial nature of syntactic structure.

For completeness, we also ran the pre-trained GPT-2 model on the syntactic suites. This yielded a score of 0.808 on the SG Test Suites and 0.827 on BLiMP-10% for the small version of pre-trained GPT-2. Among models trained on BLLIP-LG, the average accuracy score on the SG Test Suites is 0.758 for PLMs and 0.683 for LMs. Similar trend is observed on BLiMP-10% as well, where among models trained on BLLIP-LG the average accuracy is 0.754 for PLMs and 0.712 for LMs. The proposed PLM method is able to close the gap between GPT-2 small and the same model trained with BLLIP-LG by more than half, while the improvement for BLiMP is more modest but still significative. It remains an open question whether scaling syntactic supervision to a larger dataset than BLLIP-LG would bring the generalization performance of PLM models closer to that of the pretrained GPT-2 model.

3.2.1 Relationship between Perplexity and Syntactic Generalization Performance

We compare perplexity on the BLLIP held-out test set against syntactic generalization performance in Figure 4. Perplexities of PLM and PLM-mask models are computed setting the parse tree equal to the gold parse in Equation 3 to approximate the likelihood. Note that, unlike Hu et al. (2020), all our models use the same BPE vocabulary and word

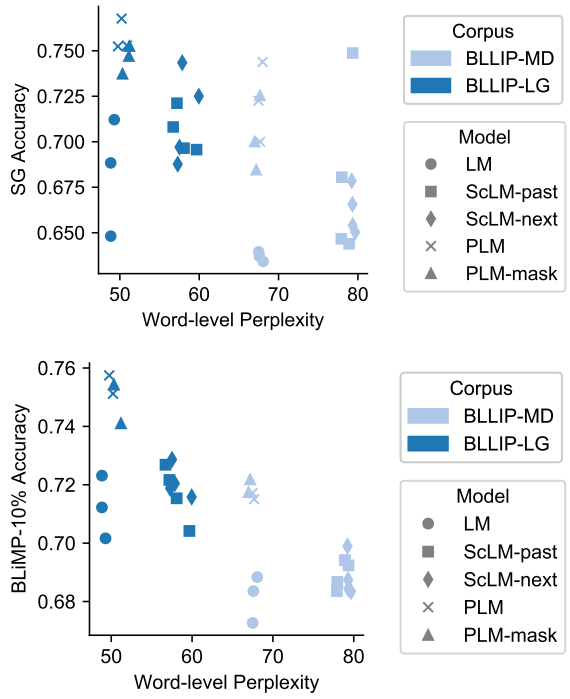


Figure 4: Comparison between model perplexity on BLLIP test data and syntactic generalization performance on SG Test Suites (top) and BLiMP-10% (bottom).

tokenization from GPT-2. The only exception are the additional parsing actions in the vocabulary y.

From Figure 4, both perplexity and syntactic generalization performance improve with dataset size. However, for both training dataset sizes, we see that structural guidance can improve syntactic generalization. PLM models consistently perform better than vanilla models; scaffolded models usually improve SG score, but not necessarily BLiMP performance.

3.2.2 Effect of Structural Guidance on Learning Specific Syntactic Structures

In addition to comparing model's aggregated performances, we also compare their generalization performances in the clustered subsets of tests in SG Test Suites and BLiMP-10%. These subsets consist of several related tests that target specific type of syntactic phenomenon, such as NPI licensing, subject-verb agreement, filler-gap dependencies, etc. We also include the results for the RNNGs taken from Hu et al. (2020).

Results in Figure 5 show converging evidence that structural guidance in the form of generative parsing can robustly improve learning of subject-verb agreement and NPI licensing, and helps the model to better capture incremental processing phenomenon such as garden-path effects, but seems to slightly hurt the performance on gross syntactic state. While overall the RNNG shows a poor performance this is mostly due to its very low scores for licensing suites. Excluding these suites only the RNNG shows a performance close to the PLM model, even outperforming it clearly for the gross syntactic state suites. In this category and binding PLM variants seem inferior to all other models.

4 Related Work

Multitask learning (Caruana, 1997) has been applied to a variety of NLP tasks with traditional modeling approaches (Miller et al., 2000; Sutton and McCallum, 2005; Sutton et al., 2007) as well as more recent neural models (Collobert et al., 2011; Li et al., 2020a). A recurring theme has been the use of structure in the form of syntactic trees to benefit other NLP tasks. Among the early works exploring this direction, Punyakanok et al. (2008) showed that syntactic parses can benefit Semantic Role Labeling (SRL). Poon and Domingos (2009) extended this idea to induce first-order logic representation in a unsupervised fashion, by clustering the dependency structures. In both cases syntax forms part of a pipeline and is not strictly supervision for the end task.

This trend continued with the rise of neural models. Collobert et al. (2011) improved deep convolution neural network for syntactic chunking models with additional POS supervision. Zhang and Weiss (2016); Søgaard and Goldberg (2016) observe the benefits of POS supervision at different depths of a neural network model with impact on dependency parsing, tagging and CCG super tagging performance. He et al. (2019) perform a syntax-based pruning of semantic roles, showing benefits in a multilingual setting. More recently, Sachan et al. (2020) incorporate a syntactic graph recurrent neural network into BERT models for better semantic role labeling. However, their method shows little or no benefit of syntax modeling for Named Entity Recognition and relation linking task. Neural machine translation (Chen et al., 2018) and text generation (Li et al., 2020a) have also been shown to benefit from syntactic modeling. In a recent work, Li et al. (2020b) use syntactic modeling in BERT based transformers to achieve performance gains on several text classification benchmarks. Other works have found that structural supervision in the

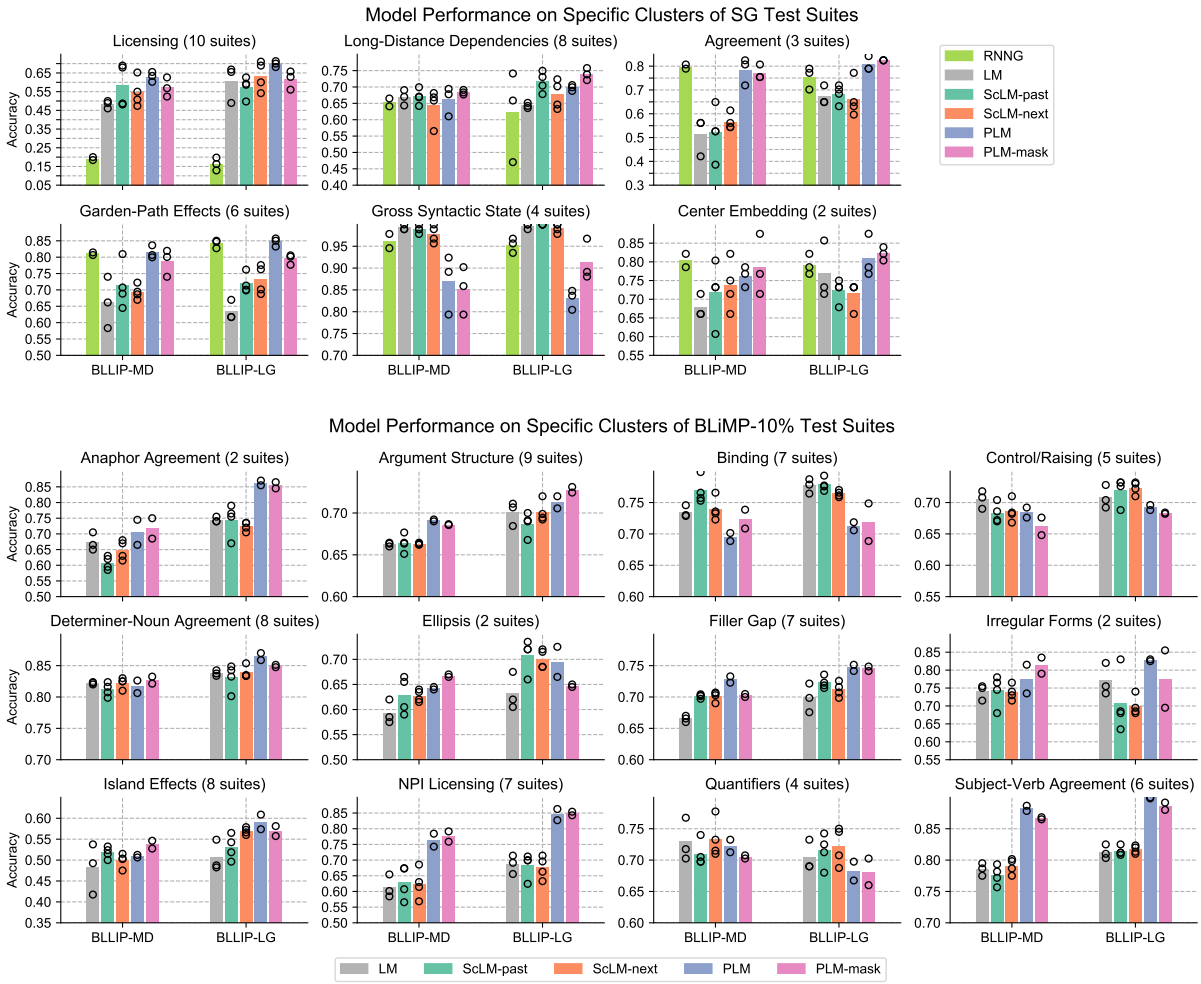


Figure 5: Model performance comparison by specific linguistic phenomena clustered in SG Test Suites (top) and BLiMP-10% (bottom). RNNG performances are from Hu et al. (2020).

form of intermediate fine-tuning (e.g., on CCG super tagging) is not helpful or even harmful (Pruksachatkun et al., 2020; Warstadt et al., 2019).

The focus of our work is on gauging the impact of joint modeling on syntactic generalization performance. In this direction, the work of Swayamdipta et al. (2018) is close to the scaffolding version of our model. They predict multiple labels, extracted from syntactic information, as auxiliary task and show positive effects on shallow semantic parsing and co-reference resolution. We use however a single feature, constituency parsing n-gram, which is closer to prior work relying on Part-of-Speech information. In addition, we explore impact of using preceding structure as feature vs postceding structure, which as shown plays a role in the learning process.

In terms of modeling objective and syntactic representations, our method is closest to the works of Choe and Charniak (2016); Dyer et al. (2016) that jointly model syntax and language. A more recent work from Peng et al. (2019) uses Rational Neural Networks language model that can derive binary unlabeled constituents from attention weights and can supervise the attention to attain a structural inductive bias. The proposed models show lower language modeling perplexity compared to their structure agnostic counterparts. We also extend here the idea of syntax-aware language modeling to transformer-based language models.

Finally, our approach relates to the other works that propose ways of incorporating structural information into Transformer-based models. This includes the use of dependency or tree structure for constraining self-attention patterns (Strubell et al., 2018; Wang et al., 2019; Zhang et al., 2020), guiding cross-attention (Chen et al., 2018; Astudillo et al., 2020), modelling syntactic distance (Du et al., 2020), using syntactic information to guide the computation flow in the model (Shen et al., 2021),

or through knowledge distillation (Kuncoro et al., 2020). Our structured masking in parsing as language modeling approach is close in spirit to the methods that modify attention mechanism according to syntactic connections (Astudillo et al., 2020); This work, however, primarily aims to study the impact of structural guidance on syntactic generalization. Therefore, we resort to simpler methods of incorporating structure to minimize the impact of modeling intricacies.

5 Conclusion

Our work explores two forms of syntactic supervision as structural guidance for Transformer language models. Experiments suggest that generative parsing approach can effectively improve systematic generalization of learned syntactic knowledge in small training data regime, while a naive syntactic scaffold approach does not improve the baseline to the same extent despite reduced computation cost at inference time. Future work may explore alternative structural guidance strategies that combine the best of both approaches.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by the MIT-IBM Watson AI Lab.

References

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. page 1001–1007.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia*, 36.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

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

Wenyu Du, Zhouhan Lin, Yikang Shen, Timothy J. O'Donnell, Yoshua Bengio, and Yue Zhang. 2020. Exploiting syntactic structure for better language modeling: A syntactic distance approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6611–6628, Online. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. 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 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Shexia He, Zuchao Li, and Hai Zhao. 2019. Syntax-aware multilingual semantic role labeling. *arXiv preprint arXiv:1909.00310*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

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

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794.

Yinghao Li, Rui Feng, Isaac Rehg, and Chao Zhang. 2020a. Transformer-based neural text generation with syntactic guidance. *arXiv preprint arXiv:2010.01737*.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2020b. Improving bert with syntax-aware local attention. *arXiv preprint arXiv:2012.15150*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A hybrid parser and language model. 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)*, pages 3644–3651, Hong Kong, China. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1–10.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R Bowman. 2020. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2020. Do syntax trees help pretrained transformers extract information? *arXiv preprint arXiv:2008.09084*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, Siva Reddy, and Aaron Courville. 2021. Explicitly modeling syntax in language models with incremental parsing and a dynamic oracle. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1660–1672, Online. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multitask learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700, Copenhagen, Denmark. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling

and segmenting sequence data. *Journal of Machine Learning Research*, 8(3).

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. 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)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, et al. 2019. Investigating bert's knowledge of language: Five analysis methods with npis. 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)*, pages 2870–2880.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. 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 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. Sg-net: Syntax guided transformer for language representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.