Oracle Linguistic Graphs Complement a Pretrained Transformer Language Model: A Cross-formalism Comparison

Jakob Prange  Nathan Schneider
Georgetown University
{jp1724, nathan.schneider}@georgetown.edu

Lingpeng Kong
The University of Hong Kong
lpk@cs.hku.hk

Abstract

We examine the extent to which, in principle, linguistic graph representations can complement and improve neural language modeling. With an ensemble setup consisting of a pretrained Transformer and ground-truth graphs from one of 7 different formalisms, we find that, overall, semantic constituency structures are most useful to language modeling performance—outpacing syntactic constituency structures as well as syntactic and semantic dependency structures. Further, effects vary greatly depending on part-of-speech class. In sum, our findings point to promising tendencies in neuro-symbolic language modeling and invite future research quantifying the design choices made by different formalisms.

1 Introduction

Modern computational representations of language generally fall into one of two buckets: either dense fixed-length embedding vectors learned by deep neural networks from raw text, or sparse symbolic graph structures hand-designed based on linguistic principles and annotated over tokenized text (symbolic linguistic representation, SLR). In both cases the goal is to encode information about grammar and meaning at one or more levels of linguistic processing (e.g., words, sentences, paragraphs), but with drastically diverging structural foundations (figure 1). Neural representations are in the foreground of most recent work in NLP due to their scalability and high performance on a vast variety of tasks. In particular, Transformer language models (Vaswani et al., 2017) and the pretrain-finetune paradigm (Devlin et al., 2019) have advanced the state of the art in many areas.

There is an ongoing debate about the role of linguistic structure in neural modeling (Tenney et al., 2019a; Bender and Koller, 2020; Trott et al., 2020). However, the broad diversity of existing SLR formalisms—each implementing a nuanced linguistic view on language structure and meaning—remains an understudied problem. Compare, e.g., the treatment of the passive auxiliary ‘were’ in the three graph-structured SLRs shown in figure 1: UD assigns it a distinct graph node, PTG explicitly groups it with the subsequent predicate, and it is unanchored in the EDS graph (by default, we attach it to the preceding unit).

How do differences in representations of linguistic structure play out in neural modeling? This question has not been investigated in depth, perhaps due to the challenge of quantifying such differences between formalisms and the linguistic criteria that
shape their design. To wit, each formalism relies on a distinct set of assumptions as to the range of linguistic phenomena covered (coarsely, syntax vs. semantics), and the types of structural relations linguistic units engage in (trees vs. DAGs, dependencies vs. higher-level structures). This makes it difficult to use a shared neural encoding scheme across different frameworks. Subsequently, state-of-the-art parsers often need to be engineered towards specific frameworks, which adds a further hurdle in comparing parsers across formalisms: if one provides a greater degree of disambiguation, parsing may be inherently more difficult, and thus comparing parsing scores across frameworks (even on the same test set) would be an apples-to-oranges comparison. Instead, one can measure how well different formalisms complement each other for joint parsing or can be merged or converted into one another (Prange et al., 2019a; Hershcovich et al., 2020). Alternatively, frameworks could be compared in an extrinsic application such as machine translation, summarization, natural language generation, or natural language inference (e.g., Hajdik et al., 2019; Wu et al., 2021); but such comparisons are rarely performed and may not generalize to other domains or application settings.

Here we envision a different approach to comparing across frameworks. Rather than comparing parser accuracy, we embed linguistic graphs within a neural language model. We use an incremental LM and compare the effect of linguistic frameworks on application-agnostic measures such as perplexity. To determine which (if any) linguistic formalisms have the potential to reduce the perplexity of a pretrained language model, we explore a best-case scenario with oracle graphs. That is, assuming a perfect linguistic representation is available for the preceding context at test time, we measure whether it can enhance next-word prediction beyond a baseline pretrained transformer (GPT-2; Radford et al., 2019). We devise a framework-general approach to encoding the linguistic graphs as discrete vectors (§3 and §4) and then use this encoding to compare 7 SLR formalisms by virtue of their language modeling capability. This comparison is carried out via a controlled experimental setup (§5), fueled by the jointly annotated dataset released with the recent Meaning Representation Parsing shared tasks (MRP; Oepen et al., 2019, 2020). The results (§6) suggest that linguistic graphs are informative for next-word prediction, complementing what is learned through standard pretraining, and at the same time invite future research quantifying the design choices made by different formalisms (§7).

2 Background: Symbolic Linguistic Representation

Following a long tradition in formal linguistics, graph-structured representations of language qualitatively describe grammatical and meaning-oriented relations among words. The SLR paradigm has seen a relatively recent revival in the form of larger-scale ‘treebanking’ and ‘sem-banking’ for training neural parsers.

Formally, an SLR instance is a directed acyclic graph (DAG) $G = (N, E, \alpha)$, with nodes $N$, labeled edges $E$, and an anchoring function $\alpha : N \rightarrow \mathbf{w}$ that maps each node to a (potentially empty) subset of tokens in the sentence. We broadly distinguish SLR frameworks along two dimensions:

Scope. The goal of syntactic representations is, broadly, to explain distributional patterns in word order; they tend to be rooted trees with often projective anchoring functions. Semantic formalisms are meaning-oriented, aiming to capture the higher-level logic expressed in a sentence; thus, they may have more complex structures, including reentrant edges and discontiguous anchors.

Structure. SLRs can further be subdivided into dependency and constituency structures, the former of which are relatively shallow, while the latter contain abstract nodes not directly anchored in any one word token.

3 Overview: Language Modeling with Linguistic Graphs

Our main goal is to quantify the predictive power of different SLR formalisms by combining them with a pretrained language model and evaluating how this affects performance in the next-token generation task.

Language modeling is a powerful and widespread tool in many NLP workbenches. A language model (LM) assigns probabilities to sentences and can be used to both process existing sentences and generate new ones. As is standard practice, we treat sentences as length-$(n+1)$ sequences of word tokens, $\mathbf{w} = \langle w_0, w_1, \ldots, w_n \rangle$. An incremental LM factorizes the joint probability

---

1 See Abend and Rappoport (2017); Koller et al. (2019); Prange et al. (2019b) for more detailed taxonomies.
of the sentence in terms of the probability of each word \( w_i \) conditioned on previous tokens \( (w_{<i}) \). This can be approximated with a Markov assumption, where the history is limited by window size \( m \):

\[
P_{LM}(w) = \prod_{i=0}^{n} P_{LM}(w_i | w_{<i}) \approx \prod_{i=0}^{n} P_{LM}(w_i | w_{i-m}, ..., w_{i-1})
\]  

(1)

Here we describe at a high level how we process SLR graphs for use in this language modeling scenario, i.e., to obtain context-conditional distributions over tokens from them. In contrast to traditional sequential LMs, contexts are now graph-structured, and which context tokens to select as well as in what way they are related to the target token is determined by the underlying SLR graph \( G \).

\[
P_{SLR}(w) := P(w|G)
\]

(2)

This general idea is closely related to syntactic language modeling (Pauls and Klein, 2012; Gubbins and Vlachos, 2013, inter alia). We extend this line of work to arbitrarily complex syntactic and semantic DAG structures and, in doing so, take particular care to restrict conditioning contexts from accessing not only future words but also future subgraphs, so effectively top-down and left-to-right.

Our high-level procedure is as follows: First, we select for each token position \( i \) to be predicted a subgraph \( G_i \) that is both relevant to the token prediction according to some criteria and admissible in the language modeling setting, i.e., it does not violate the left-to-right generation order (§4.1). The relevance criteria identify a region of the graph close to the token being predicted; here we consider criteria based on graph relationships generally, without relying on formalism-specific labels.

We call these subgraphs slices (small subgraphs for each token in figure 1; e.g., the EDS-slice for the target ‘reported’ starts at node 3, and extends to the ARG2-child 2, ARG1-coparent 1, and BV-coparent 0, which are anchored, respectively, in the spans ‘injuries’, ‘Numerous’, and ‘Numerous injuries’). Recall from §2 that context words \( w_{<i} \) are contained in \( G_i \), to the extent that they are anchored in a node reachable from \( w_i \). Slicing SLR graphs thus allows us to factorize \( P(w|G) \) as

\[
P(w|G) := \prod_{i=0}^{n} P(w_i|G_i).
\]  

(3)

The intuition for linguistic graph slicing is similar to that of Markov assumptions of independence in generative modeling and Markov blankets in causal Bayesian networks: To determine likely values of any given node in the graph it is hardly necessary to consider the entire graph. The further one travels on a path starting from a target node \( n \), the less directly relevant the information from encountered nodes becomes to \( n \). At the same time, only considering a node’s direct neighbors would be too narrow an independence assumption.

Next, we encode each graph slice into a fixed-sized vector. We use a computationally inexpensive method for statically and deterministically projecting slices into a sparse, high-dimensional space (§4.2). Finally, we compute output distributions \( P(W|G_i) \) from the vector representations (§4.3).

### 4 Modeling Details

#### 4.1 Slicing Graphs

A slice \( G_i \) is a subgraph of \( G \) that captures \( w_i \)’s linguistically structured context, masking \( w_i \) itself (or else estimating \( P(w_i|G_i) \) would be trivial). Figure 1 shows a few examples. \( G_i \) always minimally consists of \( w_i \)’s direct anchor node \( a_i = \text{Select}({n : w_i \in \alpha(n)}) \). Starting from \( a_i \), we traverse the graph and add nodes and edges that are connected to \( a_i \) via paths of a few select relative types. Here we settle on the following inventory of relative types \( REL = (P, B, O, T, C, R) \), namely, parents \( P_i \), siblings \( B_{i,p} \), grandparents \( O_{i,p} \), aunts \( T_{i,p} \) (all indexed by parent \( p \)), children \( C_i \), and coparents \( R_{i,c} \) (indexed by child \( c \)).

Precise definitions are given in table 1.

#### Table 1: Relative types and capacities

| rel | Name | Definition | \( \gamma \) |
|-----|------|-----------|-----------|
| \( P \) | parent | \( \{n : (n, a_i) \in E\} \) | 2 |
| \( B_{i,p} \) | sibling | \( \{n : (p, n) \in E\} \forall p \in P_i \) | 2 |
| \( O_{i,p} \) | grandparent | \( \{n : (n, p) \in E\} \forall p \in P_i \) | 1 |
| \( T_{i,p} \) | aunt | \( \{n : (n, o) \in E \wedge o \in O_{i,p}\} \) | 2 |
| \( C_i \) | child | \( \{n : (o, n) \in E\} \) | 1 |
| \( R_{i,c} \) | coparent | \( \{n : (n, c) \in E\} \forall c \in C_i \) | 1 |

In case there are multiple anchoring options (see, e.g., EDS nodes 0 vs. 1 for first token in figure 1), we use the following tie-breaker heuristics: Choose the anchor node with the most parent and child nodes; if still a tie, choose the node with the highest node ID (tends to be hierarchically lower, i.e., vertically closer to the token anchor).

3Selected based on general linguistic intuition and preliminary experiments, but without ‘tuning’ for specific formalisms.
We allocate capacities $\Gamma = \{ a_i \} \cup P_i \cup \bigcup_{p \in P_i} B_{i,p} \cup O_{i,p} \cup T_{i,p} \cup C_i$ and $G_i = \langle N_i, E_i, \alpha \rangle$. (4)

To prevent information leaks from future tokens, we discard from $G_i$ all nodes $\{n : \alpha(n) = w_j, j > i\}$ (and associated edges) which are only anchored in tokens following $w_j$. E.g., in figure 1, the UD-slice for the token ‘were’ does not contain the parent node 3 because that is anchored only in the token ‘reported’ which follows ‘were’ (and thus the sibling 1 cannot be accessed either). If a node’s anchors contain or overlap with $w_i$ (i.e., neither only precede nor only follow $w_i$), we retain the node and its edges, but remove its token anchors.

Note that two distinct slices $G_i, G_j, i \neq j$ may overlap in the original graph, but can always be distinguished by their structural identity with respect to the anchor.

4.2 Vectorizing Graph Slices

We allocate capacities $\Gamma = \{ \gamma_{rel} : rel \in \text{REL} \}$ for each relative type (right-most column in table 1). Up to $\gamma - 1$, relative nodes are added “with high resolution”, maintaining their identity and order; beyond the capacity, the capacities are aggregated “with low resolution”. Within each relative type, precedence $k$ is given to relatives whose token anchors are sequentially closer to $w_i$.

$$\text{HiRes}_{i,rel} = \{ \text{rel}_k : k < \gamma_{rel} \}$$

$$\text{LoRes}_{i,rel} = \{ \text{rel}_k : k \geq \gamma_{rel} \}$$

As a side note, this grouping specifies a hypergraph $G_i$ over $G_i$:

$$\hat{N}_i = \bigcup_{\text{rel}} \{ n : n \in \text{HiRes}_{i,rel} \} \cup \{ \text{LoRes}_{i,rel} \}$$

$$\hat{G}_i = \langle \hat{N}_i, E_i, \alpha \rangle$$

(6)

Next we look up the hypergraph’s edge label and word vector encodings $\hat{l}_k$ and $\hat{w}_k$ and collate them into a single vector $\tilde{s}_{i,rel}$ per relative type. High-resolution vectors are concatenated $\bigoplus$ and low-resolution vectors are averaged. $\tilde{s}_{i,rel}$ is zero-padded whenever $|\.rel| < \gamma_{rel}$.

$$\tilde{s}_{\text{HiRes}_{i,rel}} = \bigoplus_{k \in \text{HiRes}_{i,rel}} \hat{l}_k; \hat{w}_k$$

$$\tilde{s}_{\text{LoRes}_{i,rel}} = \sum_{k \in \text{LoRes}_{i,rel}} \hat{l}_k; \hat{w}_k$$

(7)

Finally, we concatenate all of these relative-vectors to obtain the final vector representation of the whole slice, $\tilde{s}_i$.

$$\tilde{s}_i = \bigoplus_{\text{rel} \in \text{REL}} \tilde{s}_{i,rel}$$

(8)

This vector essentially specifies a linguistically principled, deterministic, typed, hard self-attention over the token history.

An advantage of this method over recursive (Socher et al., 2011a,b, inter alia) or convolutional (Kipf and Welling, 2017, inter alia) parametrized graph encoders is that slice vectors are interpretable (e.g., an interesting avenue for future work could be to analyze their backprop gradients) and do not change during training, so they can be cached in preprocessing. There might be further qualitative benefits to static transparent encodings of structure (as, e.g., in constructive CCG supertagging; Prange et al., 2021), but we do not test that here.

4.3 Predicting Emission Distributions

We compute model posteriors for next-token predictions as

$$P_{\mu}(w_i = \nu_{i} | \text{context}_{i,\mu}) = \text{SoftMax}(\text{logits}_{i,\mu}[k])$$

where $\mu$ stands for either a pure SLR model or LM, or an ensemble of the two.

SLR only. As described above, we define context$\text{SLR}$ as $G_i$, which is encoded as $\tilde{s}_i$. We obtain $P_{\text{SLR}}$ by letting the slice-vectors serve as inputs to a $d$-multilayer perceptron (MLP) with a final softmax layer over the vocabulary, which yields the estimated token emission distributions.

$$\text{logits}_{\text{SLR}} = \text{MLP}(\tilde{s}_i)$$

$$\text{MLP}_d(x) = H^{(d)}(\ldots H^{(1)}(x)) \text{Emb}^T,$$

where $\text{Emb}$ is an embedding matrix.
LM + SLR. Since we want to measure whether and how much the information contained in the SLR can contribute to state-of-the-art language models, our primary experimental condition is a combined setup $P_{\text{Ensemble}}$, where logits obtained from slice-encodings are added to a base neural LM’s logits before taking the softmax:

$$\text{logits}_{i,\text{Ensemble}} = \text{logits}_{i,\text{SLR}} + \text{logits}_{i,\text{LM}},$$

where

$$\text{logits}_{i,\text{LM}} = \text{LM(w)}[i].$$

LM only. $P_{\text{LM}}$, i.e., the bare LM without any exposure to SLR graphs, serves as our baseline.

5 Experimental Setup

All models are implemented in PyTorch and experiments are run on 1 NVIDIA Tesla T4 GPU.

5.1 Data

Our data set consists of the intersection of Wallstreet Journal (WSJ; English financial news) sentences that have been annotated with syntactic trees in the Penn Treebank (PTB; Marcus et al., 1993; Hovy et al., 2006)\footnote{See appendix A for details.} as well as a range of semantic representation formalisms for the MRP 2019 & 2020 shared tasks (Oepen et al., 2019, 2020). Quasi-gold UD 2.0 trees are obtained by running the UD converter released with the Java Stanford Parser, version 4.2.0,\footnote{https://catalog.ldc.upenn.edu/LDC2013T19} on the PTB trees.

We split the corpus into training and development\footnote{We evaluate on the data split that was used as evaluation data in MRP 2019 and as development data in 2020, as only for this data gold annotations for all formalisms have been released. We do not perform empirical hyperparameter tuning. In early development, a small subset of the data was used.} data following the MRP task setup; summary statistics are shown in table 2.

|          | Train | Dev |
|----------|-------|-----|
| Sentences| 26,325| 921 |
| Tokens   | 658,475| 22,596|
| Toks/sent| 25.0 | 24.5|
| Vocabulary| 27,344 | 5,364|

Table 2: Data statistics.

5.2 SLR Formalisms

In table 3, and below, we give an overview of the 7 (versions of) linguistic representation frameworks examined in this study.

| Formalism | Sem/syn | Structure  |
|-----------|---------|------------|
| UD        | syntax  | dependency |
| PTB-phrase| syntax  | constituency|
| PTB-func  | syntax  | constituency|
| PSD       | semantics | dependency |
| PTG       | semantics | constituency|
| DM        | semantics | dependency |
| EDS       | semantics | constituency|

Table 3: Properties of the chosen linguistic representation formalisms.

5.3 Language Model and Data Encoding

The base language model we use in all our experiments is GPT-2 (Radford et al., 2019, as distributed in the huggingface-transformers PyTorch library). GPT-2 is a Transformer model (Vaswani et al., 2017) pretrained on a diverse collection of web texts, notably excluding Wikipedia articles.

In contrast to other widely-used Transformers like BERT (Devlin et al., 2019), which optimize bidirectional masked language modeling (MLM), GPT-2 is incremental, i.e., the decision over the next word only takes into account the preceding context.

Tokenization. We follow the sentence segmentation of the Penn Treebank corpus. Within sentences, we obtain token boundaries from GPT-2’s
pretrained byte-level byte-pair encoding (BBPE) tokenizer. The BBPE tokens are then aligned with the formalism-dependent SLR node anchors via raw-text character offsets. Tokens that are continuations of multiword anchors in the graph (‘_reported’ in PTG, figure 1); subword tokens of a single graph anchor (‘N-umerous’); or are unanchored in the graph (‘_were’ in EDS), are treated as unanalyzable, i.e., their slice consists of a copy of the preceding token’s slice, plus the preceding within-anchor tokens.

Representing tokens and labels. We use GPT-2’s pretrained global embeddings (from the lowest layer, before any local contextualization) both to obtain embeddings for relative token anchors in the slice-vector and again in the MLP to project the last hidden state into the vocabulary. This ensures that there is no surface-lexical information gap between GPT-2 and the SLR models, which would likely be the case if we were training embeddings from scratch on the limited graph-annotated training data alone. Thus, the main difference between the models is a structural and abstract one, namely, which observed token anchors relate to the target anchor and how.

SLR edge labels are encoded as one-hot vectors since label sets are relatively small and we are more interested in analyzing the ‘hard’ boundaries between different linguistic design philosophies than in creating more model parameters to optimize.

5.4 Training

We train all models for 10 epochs with the AdamW optimizer (Loshchilov and Hutter, 2019), minimizing cross-entropy between the model posterior and the ground-truth at each token position.

5.5 Evaluation

We compute model perplexity (PPL) as the most standard language modeling evaluation measure, as well as accuracy (Acc) and confidence (Conf) of a model’s top-ranked guess, mean reciprocal rank (MRR), and entropy of the model’s posterior (in contrast, PPL is the exponent of the correct answer’s probability). Means ± stdev over individual formalisms; # of formalisms per condition in parentheses.

6 Findings

6.1 Main Results

The most striking observation in terms of overall model performance (table 4) is that ground truth linguistic graphs of all investigated linguistic formalisms improve vanilla GPT-2 by a large margin, in all metrics. This improvement holds up when compared to a version of GPT-2 that is exposed to the raw WSJ text without the graphs; with this condition we control for mere domain differences between our evaluation data and the data GPT-2 was trained on originally (‘+Domain’ in table 4). The large performance gap suggests that at least a subset of the oracle knowledge about linguistic structure may not yet be encoded in the base language model, which learns from only raw text. Contrasting overall accuracy with overall confidence, we note that both additional in-domain
We show perplexities of the baseline model (domain-which is relatively consistent in all metrics, is indi-
ated by order of rows, with UD having the smallest (though still respectable) improvement over the baseline, and PTB and EDS the largest.

Interestingly, there are two marked separations: a primary one between dependency and constituency formalisms, and a secondary one between syntactic (i.e., more surface-oriented) and semantic (i.e., more abstract) formalisms. This is summarized in Table 5. A limiting factor for dependency representations in the incremental LM setting is that relations between the target token and subsequent tokens are entirely ignored, whereas constituency graphs can back off to higher-level structures. Semantic constituency representations, with their abstract and meaning-oriented labeling and structure schemes, jump out as being especially predictive of the underlying text, as compared to both syntax and shallow semantics.

We note a few additional details regarding the three pairs of closely related formalisms: First, the function-enhanced PTB label set has a slight advantage over the basic phrase-structure labels. Second, we observe mild correlations in the entropy, accuracy, and confidence ranking between DM and ESD on one hand, and PSD and PTG on the other, which each are dependency and constituency versions converted from the same underlying grammars. PPL and MRR do not follow this same ranking, but are themselves correlated.

Since we use oracle linguistic graphs in this study, it is not guaranteed that we compare the different frameworks on an entirely level playing ground. For instance, it could be that the extremely helpful abstract semantic information contributed by EDS and PTG is also much harder to predict automatically at test time than UD or other dependency formalisms. Our results are thus to be understood as estimated upper bounds.

6.3 Linguistic Phenomena

To better understand where particular strengths and weaknesses of the baseline LM and linguistically enhanced models lie, we analyze subsets of tokens by part-of-speech tag (Table 6). We highlight a few interesting patterns:

Across all models there is a clear and expected separation between rather predictable function words (auxiliaries, adpositions, particles, subordinating conjunctions, coordinating conjunctions, determiners, pronouns), more perplexing content words (nouns, verbs, modifiers), as well as numbers, punctuation, and miscellaneous tokens somewhere in the middle.

Average perplexity of the tested SLR models is better than baseline GPT-2 in all POS classes but one. The one exception is the noun class, where both the SLR macro-average and, e.g., UD in particular do not raise performance. Only EDS and DM show perplexity improvements on nouns; PTB even has a noticeable negative impact (see appendix B for more details). We conjecture that this may have to do with relatively deep NP nesting in PTB (compared to the other formalisms), such that the current slicing hyperparameters (relative types and capacities) are too strict and hide informative signals like modifiers and verb attachment.

Some formalisms seem to be particularly well-suited for the prediction of certain parts-of-speech:

| POS   | Dev Toks | Train Vocab | GPT-2 | UD   | EDS | SLR Avg |
|-------|----------|-------------|-------|------|-----|---------|
| All   | 22,596   | 27,344      | 45.8  | 35.2 | 26.6| 32.3 ± 3.1 |
| noun  | 5,865    | 18,435      | 176.0 | 180.1| 136.0| 177.8 ± 20.9 |
| verb  | 2,404    | 7,100       | 153.5 | 113.8| 119.8| 6.5 ± 6.5  |
| mod   | 1,884    | 6,292       | 265.7 | 193.3| 106.4| 143.6 ± 29.5 |
| aux   | 569      | 95          | 18.0  | 10.8 | 5.9  | 9.1 ± 2.3  |
| adp   | 1,928    | 232         | 9.7   | 6.8  | 5.5  | 5.1 ± 1.6  |
| part  | 814      | 27          | 8.7   | 5.4  | 4.0  | 5.0 ± 0.7  |
| sconj | 263      | 96          | 14.7  | 11.5 | 6.8  | 6.6 ± 4.0  |
| cconj | 535      | 35          | 12.8  | 7.1  | 1.9  | 4.3 ± 2.1  |
| det   | 1,696    | 91          | 9.3   | 7.4  | 4.3  | 5.8 ± 1.3  |
| pron  | 782      | 149         | 21.0  | 16.2 | 4.8  | 10.3 ± 3.8 |
| num   | 472      | 1,059       | 47.6  | 38.2 | 31.2 | 36.1 ± 4.0 |
| punct | 5,112    | 68          | 17.9  | 11.8 | 12.3 | 13.4 ± 1.1 |
| misc  | 272      | 183         | 18.8  | 15.4 | 18.5 | 18.5 ± 1.7 |

Table 6: Breakdown by Universal POS (UPOS). We show perplexities of the baseline model (domain-trained GPT-2), two representative SLR-combined models, and the macro-average ± stdev over all SLR-combined models, as well as development token counts and observed training vocabulary size of each word class for reference. Stdev over 5 random seeds for each model is small: <1 for function words (aux, adp, part, sconj, cconj, det, pron) and punctuation, <2 for number words, and otherwise <3 with 3 exceptions: nouns in PTG (3.25) and PTB-phase (3.66), and verbs in DM (5.64). noun = {NOUN, PROPN}, mod = {ADJ, ADV}, misc = {INTJ, SYM, X}. Training and exposure to SLRs seems to make the models increasingly overconfident (up to 8–12%, compared to 4% with the vanilla model).
E.g., UD for verbs; PTB and PTG for adpositions and subordinating conjunctions; EDS for pronouns, determiners, and numbers; as well as PTG, PSD, and EDS for coordinating conjunctions. The advantage of EDS and DM on nouns, pronouns, determiners, and numbers can likely be attributed to its explicit representation of variable binding/quantification. Similarly, PTG and PSD have detailed categories for coordination, distinguishing, e.g., conjunction and disjunction.

Finally, for nouns and modifiers, the spread (standard deviation) across formalisms is particularly wide, which suggests that symbolic representations of these types of words are especially diverse (e.g., whether adjectives and certain nouns can count as predicates or not) and that this diversity has a strong effect on utility for language modeling.

6.4 Model Ablations

Before jumping to conclusions, we shall rule out several factors that could potentially confound these results. The linguistically enriched models consist of a substantial number of newly learned parameters—around 50–60M each, an additional ~50% the size of vanilla GPT-2. Although model size does not seem to be correlated with performance among the SLR-enriched models, it could still be that the additional capacity allows the models to store more information about the words’ distributions than the baseline GPT-2 model, without ever truly using the concrete linguistic structures.

We check this by randomly shuffling (∧) two core properties of the graphs: (i) the assignment of edge labels and (ii) the anchoring mapping between nodes and word tokens in each graph. If the models are largely independent of the actual correct label and structure assignments in predicting next words, these changes should have a very small effect on overall performance (Dubossarsky et al., 2018; Hewitt and Liang, 2019).

But on the contrary, we find that performance worsens considerably in the ablated settings compared to the full combined models of each formalism (Table 7, see appendix B for more details). This confirms that the models really do acquire—and are quite sensitive to—the graph-encoded linguistic signals, relying to a large part on this new information in making their predictions.

Shuffling only edge labels while leaving the rest of the graphs unchanged has a smaller effect than changing how tokens are anchored in the graph structure. This suggests that the attention-like character of the linguistic graphs, i.e., their selection of relevant context tokens, plays a crucial role in the SLR-combined models—more so than knowing the type(s) of grammatical slot(s) the target token fills.

If a model has learned to rely on correct labels and structure during training, then perturbing these properties at test time has a highly adverse effect, confusing the model and leading to a drastic decrease in performance—even worse than not consulting SLR graphs at all! Given previous findings that syntactic structure is to some extent already encoded in the pretrained LM (Linzen et al., 2016; Tenney et al., 2019b), we conjecture that this representational capacity gets offloaded to the structures at training time, thus the permuted test-time graphs fool the PTB model to a much greater extent than DM.

As expected, exposing models to shuffled graphs at training time renders the additional model parameters practically neutral, resulting in similar perplexity as the base LM. In this case, it also does not matter whether test-time graphs are correct or random (‘training’ vs. ‘both’ in column 2)—either way, in training the model should learn to mostly disregard the random structure as noise.

7 Discussion

7.1 Related Work

Researchers have long been interested in scaffolding sequential language modeling with linguistically-motivated structured inductive biases.

| Ablation | Applied in | DM | PTB | SLR Avg |
|----------|------------|----|-----|---------|
| None     | 34.2       | 33.5| 32.3| ± 3.1   |
| × Labels | testing    | +11.4| +144.4| +59.5| ±59.6 |
| × Anchors| testing    | +59.4| +245.6| +161.6| ±94.1 |
| × Both   | testing    | +54.1| +260.2| +151.7| ±98.1 |
| × Labels | training   | +2.8 | +9.9| +5.8| ± 4.0 |
| × Anchors| training   | +12.0| +21.7| +17.8| ± 5.7 |
| × Both   | training   | +13.0| +25.6| +19.7| ± 7.1 |
| × Labels | both       | +2.6| +11.1| +6.3| ± 4.4 |
| × Anchors| both       | +10.3| +21.4| +17.7| ± 6.2 |
| × Both   | both       | +10.9| +22.5| +18.7| ± 6.8 |
| – SLR    | both       | +11.6| +12.3| +13.5| ± 3.1 |
| – LM     | both       | +233.1| +136.8| +196.7| ±75.7 |

Table 7: Ablations measured in ∆PPL for two representative SLR-combined models and the macro-average ± stdev over all SLR-combined models. ‘None’ and ‘−SLR’ correspond, respectively, to table 4’s rows 4 (DM) / 6 (PTB-phrase) and row 2 (GPT-2 +Domain). In the ‘−LM’ setting we evaluate $P_{SLR}$ in isolation, without combining it with GPT-2.
Syntactic language modeling dates back to the pre-neural era, when Pauls and Klein (2012) and Gubbins and Vlachos (2013) began generalizing Markov assumptions from word n-grams to syntactic subtrees or ‘treelets’.

These ideas have since been adapted to recurrent neural network (RNN) LMs (Mirowski and Vlachos, 2015) and expanded on, resulting in models like recurrent neural network grammars (RNNG; Dyer et al., 2016), parsing-reading-predict networks (PRPN; Shen et al., 2018), and ordered neurons (ON; Shen et al., 2019), among many others. Ek et al. (2019) condition RNN-LMs on syntactic and semantic tags and compare the models’ acceptability judgements with those of humans.

In the era of attention-based neural modeling of language dominated by pretrained Transformers, there have been two main directions:

One group of approaches has in common that the entire input text is first parsed, and then the estimated SLR instance is used to guide the model, for example, by directly optimizing Transformers’ attention weights to reflect linguistic graph structures (Strubell et al., 2018; Bai et al., 2021; Slobodkin et al., 2021). Rather than controlling the existing sequential attention itself, Wu et al. (2021) extend a pretrained Transformer with an additional graph encoder, optimizing for GLUE. Notably, Wu et al. (2021) and Slobodkin et al. (2021) experiment with a few different semantic and syntactic SLRs, while all other studies we have looked at are limited to either syntax or very shallow semantics.

In contrast, some recent work more closely continues the old syntactic language modeling tradition by incrementally generating words and SLRs with either a joint model (Peng et al., 2019; Qian et al., 2021) or an iteratively-coupled LM and parser (Choshen and Abend, 2021).

Since these structurally enriched neural language models are often optimized (finetuned) for and evaluated on a variety of specific NLP tasks—like semantic role labeling, machine translation, natural language inference, and the GLUE benchmark (Wang et al., 2019)—rather than language modeling in its own right, it is difficult to compare them directly to our findings.

Another relevant line of work employs probing tasks in investigating to what extent grammar and meaning are already encoded in neural language models trained predominantly on raw text with little to no linguistic supervision (Linzen et al., 2016; Tenney et al., 2019a,b; Hewitt and Manning, 2019; Liu et al., 2019; Kim et al., 2019; Wu et al., 2020; Geiger et al., 2021, inter alia).

7.2 Limitations and Future Work

For practical reasons, we only include formalisms in our study that are anchored in linguistic units roughly corresponding to subword tokens used in language modeling. This prevents us from applying the same paradigm to some other widely-used unanchored formalisms like AMR (Banarescu et al., 2013) without some changes to the setup.

While the use of oracle graphs has theoretical (measuring an upper bound without needing to account for potential errors or uncertainties) and practical (saving the computational overhead from training and running a parser) advantages, ground-truth SLR graphs are a very limited resource and thus generally assumed to only be available at training time. In future work, we plan to explore beyond this assumption and obtain graph slice estimates from an incremental parser.

An important aspect of measuring the quality of all computational linguistic models and symbolic formalisms alike is how well they generalize to the diversity of languages and within-language variation. In the present study our main focus is on creating a controlled testing environment, meaning that we deliberately selected a text corpus with as many overlapping annotations from different representation frameworks as possible. This has the unfortunate side effect that the analyzed text sample is very small in the grand scheme of things and only comprises English text from a very specific genre not representative of the English language as a whole.

7.3 Broader Impact

Although our findings are not (yet) directly transferrable to practical language modeling ‘in the wild’ due to the oracle setting, there are several important take-aways to be drawn from this study.

Our experiments provide supporting evidence for the thesis of Bender and Koller (2020), Trott et al. (2020), and others that meaning goes beyond form in language. Computational models of language that exclusively learn from even very large amounts of raw text are thus generally expected
to hit a ceiling which can only be overcome with access to higher-level structures and mechanisms of understanding.

It further seems to matter in which manner and shape linguistic graph structure is drawn. Assuming a perfect incremental parser, deeper structure and semantic categorization seems to be particularly beneficial for integration with a standard language model. This is in line with previous findings by, e.g., Tenney et al. (2019b) that while pretrained LMs tend to encode shallow syntactic structure, abstract relations are more difficult to probe for.

We thus see a promising research direction in moving towards linguistic scaffolding of language models with representations that are more complex than tags or dependencies and that capture meaningful relations beyond surface structure.

8 Conclusion

We have presented evidence that symbolic linguistic representations of varying frameworks have the potential to aid a pretrained incremental Transformer language model in task-neutral next-word predictions. Our approach involves a framework-agnostic neural encoding scheme for linguistic graphs, which we run on an English dataset jointly annotated with 7 different formalisms (Hovy et al., 2006; Oepen et al., 2019, 2020). The results highlight the importance of appreciating the complexity of linguistic structure and handling its computational representation with nuance.

References

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In Proc. of ACL, pages 77–89, Vancouver, Canada.

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving pre-trained transformers with syntax trees. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 3011–3020, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pages 178–186, Sofia, Bulgaria.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5185–5198, Online. Association for Computational Linguistics.

Alena Böhmová, Jan Hajicí, Eva Hajiciová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, Treebanks: Building and Using Parsed Corpora, Text, Speech and Language Technology, pages 103–127. Springer Netherlands, Dordrecht.

Leshem Choshen and Omri Abend. 2021. Transition based graph decoder for neural machine translation. ArXiv:2101.12640.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. Research on language and computation, 3(2):281–332.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proc. of NAACL-HLT, pages 4171–4186.

Haim Dubossarsky, Eitan Grossman, and Daphna Weintrall. 2018. Coming to your senses: on controls and evaluation sets in polysemy research. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1732–1740, Brussels, Belgium. 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.

Adam Ek, Jean-Philippe Bernardy, and Shalom Lappin. 2019. Language modeling with syntactic and semantic representation for sentence acceptability predictions. In Proceedings of the 22nd Nordic Conference on Computational Linguistics, pages 76–85, Turku, Finland. Linköping University Electronic Press.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. Natural Language Engineering, 6(1):15–28.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In Advances in Neural Information Processing Systems.

Joseph Gubbins and Andreas Vlachos. 2013. Dependency language models for sentence completion. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages
2020 Shared Task: Cross-Framework Meaning Representation Parsing, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershovich, Marco Kuhlmann, Tim O’Gorman, Nianwen Xue, Jayeel Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-framework meaning representation parsing. In Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning, pages 1–27, Hong Kong. Association for Computational Linguistics.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC’06), Genoa, Italy. European Language Resources Association (ELRA).

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 959–968, Jeju Island, Korea. 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.

Jakob Prange, Nathan Schneider, and Omri Abend. 2019a. Made for each other: Broad-coverage semantic structures meet preposition supersenses. In Proc. of CoNLL, pages 174–185, Hong Kong, China. Association for Computational Linguistics.

Jakob Prange, Nathan Schneider, and Omri Abend. 2019b. Semantically constrained multilayer annotation: the case of coreference. In Proc. of DMR, pages 164–176, Florence, Italy.

Jakob Prange, Nathan Schneider, and Vivek Srikumar. 2021. Supertagging the long tail with tree-structured decoding of complex categories. Transactions of the Association for Computational Linguistics, 9:243–260.

Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. 2021. Structural guidance for transformer language models. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3735–3745, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In ICLR, pages 1–22.

Petr Sgall, Eva Hajicová, and Jarmila Panevová. 1986. The meaning of the sentence and its semantic and pragmatic aspects. academia.

Yikang Shen, Zhouran Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In International Conference on Learning Representations.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In Proc. of ICLR.

Aviv Slobodkin, Leshem Choshen, and Omri Abend. 2021. Semantics-aware attention improves neural machine translation. ArXiv:2110.06920.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In Proc. of NIPS, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 151–161, Edinburgh, Scotland, UK. 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, Brussels, Belgium. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In International Conference on Learning Representations.

Sean Trott, Tiago Timponi Torrent, Nancy Chang, and Nathan Schneider. 2020. (Re)constructing Meaning in NLP. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5170–5184, Online. Association for Computational Linguistics.
you need. In Proc. of NeurIPS, pages 5998–6008, Long Beach, CA, USA.

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

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021. Infusing Finetuning with Semantic Dependencies. Transactions of the Association for Computational Linguistics, 9:226–242.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4166–4176, Online. Association for Computational Linguistics.
A Sentence Filtering

To establish a common ground for comparison, we take the intersection of sentences occurring in the annotated datasets of *all* linguistic formalisms.

In a first step, we discard two sentences whose linguistic graph in at least one formalism is empty.\(^9\) We then select only those 35,513 train / 1,401 dev sentences that appear in both the MRP 2019 and 2020 datasets (the 2019 corpus contains 143/1,958 more in train/dev). Next, we take the intersection of these sentences and OntoNotes 5.0, which contains the gold PTB syntax annotations. 26,719/929 sentences remain in the training/development set.

The MRP graph format operates on raw-text character offsets, while PTB and UD trees operate on word tokens. We are able reconstruct offset-based text anchors for PTB and UD from the raw text strings used in the MRP data for all but 394 train / 8 dev sentences, which leaves us with the final 26,325 training and 921 development sentences.

B Detailed Results

Below we report detailed results per POS-class table 8 as well as ablation experiments table 9 for all SLR formalisms.

\(^9\)The sentence “It is.” in DM and a ‘sentence’ consisting of the @-symbol in PTG.
| POS     | Toks   | Vocab | Toks   | Vocab | UD  | DM  | PSD  | PTB-phrase | PTB-fxn | PTG  | EDS  |
|---------|--------|-------|--------|-------|-----|-----|------|------------|---------|------|------|
| All     | 22,596 | 5,364 | 658,475| 27,344| 35.2| 34.2| 34.1 | 33.5       | 32.4    | 29.6 | 26.6|
| noun    | 5,865  | 2,555 | 223,267| 18,435| 180.1| 2.3 | 172.9| 1.1       | 179.6   | 2.7  | 200.2| 3.66 |
| verb    | 2,404  | 1,099 | 74,218 | 7,100 | 18.7 | 2.0 | 127.2| 1.6       | 125.4   | 2.0  | 140.4| 1.52 |
| mod     | 1,884  | 815   | 65,900 | 6,292 | 193.3| 1.9 | 166.8| 1.3       | 147.7   | 1.7  | 142.2| 1.21 |
| aux     | 569    | 43    | 18,303 | 95    | 10.8 | 1.8 | 6.4  | 0.2       | 8.9     | 0.9  | 11.4 | 0.42 |
| part    | 814    | 144   | 17,263 | 27    | 5.4  | 1.0 | 5.5  | 0.1       | 6.5     | 0.3  | 9.4  | 0.19 |
| sconj   | 263    | 41    | 9,053  | 96    | 11.5 | 1.1 | 7.4  | 0.8       | 11.9    | 0.5  | 3.2  | 0.03 |
| cconj    | 535    | 8     | 14,485 | 35    | 7.1  | 1.0 | 5.5  | 0.6       | 5.7     | 0.1  | 6.5  | 0.01 |
| det     | 1,696  | 34    | 51,762 | 91    | 7.4  | 1.0 | 6.4  | 0.3       | 6.5     | 0.3  | 9.4  | 0.29 |
| pron    | 782    | 64    | 23,473 | 149   | 16.2 | 1.0 | 13.8 | 0.9       | 10.9    | 0.3  | 9.1  | 0.27 |
| num     | 472    | 115   | 26,161 | 1,059 | 38.2 | 1.2 | 35.4 | 0.7       | 34.6    | 0.7  | 40.9 | 1.19 |
| punct   | 5,112  | 1,646 | 69,358 | 68    | 11.8 | 1.1 | 13.9 | 0.2       | 15.1    | 0.3  | 13.9 | 0.18 |
| misc    | 272    | 94    | 5,614  | 183   | 18.8 | 0.9 | 18.5 | 0.3       | 19.4    | 0.5  | 18.2 | 0.54 |

Table 8: Detailed breakdown by POS.

| Ablation | Applied in | UD  | DM  | PSD  | PTB-phrase | PTB-func | PTG  | EDS  |
|----------|------------|-----|-----|------|------------|---------|------|------|
| None     | 35.2       | 34.2| 34.1| 33.5 | 32.4       | 29.6    | 26.6|
| - Labels | +21.1      | +11.4|+12.5|+144.4|+145.6     | +37.1   | +44.4|+18  |
| - Anchor | +174.8     | +59.4|+42.3|+245.6|+295.5     | +197.3  | +116.3|+71  |
| - Both   | +142.7     | +51.4|+35.7|+269.2|+291.2     | +154.5  | +114.3|+53  |
| - Labels | +2.8       | +2.8 | +2.8 | +9.9  | +11.7      | +8.2    | +2.4 | +0.4|
| - Anchor | +16.0      | +12.0|+8.8 | +21.7 | +23.4      | +22.7   | +20.0|+0.5|
| - Both   | +17.0      | +13.0|+9.0 | +25.6 | +27.3      | +26.0   | +20.1|+0.4|
| - Labels | +2.7       | +2.7 | +2.7 | +3.6  | +11.1      | +8.6    | +2.8 |+0.3|
| - Anchor | +17.1      | +10.3|+8.8 | +21.4 | +22.7      | +25.3   | +18.3|+0.8|
| - Both   | +18.1      | +10.9|+9.0 | +22.5 | +24.8      | +27.1   | +18.3|+0.9|
| - SLR    | +10.6      | +11.6|+11.7|+12.3 | +13.4      | +16.2   | +19.2|
| - LM     | +328.9     | +233.1|+246.6|+136.8|+121.8     | +246.6  | +141.2|+23|

Table 9: Detailed ablations measured in ΔPPL.