Linguistic Frameworks Go Toe-to-Toe at Neuro-Symbolic Language Modeling

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, different syntactic and semantic graph representations can complement and improve neural language modeling. Specifically, by conditioning on a subgraph encapsulating the locally relevant sentence history, can a model make better next-word predictions than a pretrained sequential language model alone? With an ensemble setup consisting of GPT-2 and ground-truth graphs from one of 7 different formalisms, we find that the graph information indeed improves perplexity and other metrics. Moreover, this architecture provides a new way to compare different frameworks of linguistic representation. In our oracle graph setup, training and evaluating on English WSJ, *semantic constituency* structures prove most useful to language modeling performance—outpacing syntactic constituency structures as well as syntactic and semantic dependency structures.

1 Introduction

Linguistic theories posit that humans can take advantage of hierarchical structure related to some notion of *compositionality* to produce and comprehend utterances with complex meanings. Yet explicit representations of this kind of structure are harder to come by than raw text, and large-scale pretrained neural language models (e.g., Devlin et al., 2019; Radford et al., 2019) have managed to perform strikingly well at contextually encoding and predicting words from distributional evidence alone. At the same time, there are good reasons to doubt that these models can be said to *understand* language in any meaningful way (Trott et al., 2020; Bender and Koller, 2020; Merrill et al., 2021). To address this conundrum, people have started to explore probing pretrained models (Liu et al., 2019; Tenney et al., 2019a, *inter alia*) and supplementing training data with linguistic structure guidance (Strubell et al., 2018; Swayamdipta et al., 2018; Peng et al., 2019; Wu et al., 2021, *inter alia*).

A question that has received less attention is *which kind* of symbolic linguistic representation (SLR) is most conducive to guiding neural language models (LMs). Numerous domain-general candidates exist (Abend and Rappoport, 2017; Oepen et al., 2019, 2020; Žabokrtský et al., 2020; Müller, 2020): some are focused on syntactic structure, others on semantics (§2; big grey example graphs in the left panels of figure 1). Frameworks vary along several dimensions, with different label inventories and treatments of specific constructions. Formal differences include the type of structure (dependency or constituency, one or multiple parents, projectivity) and its relation to the input string. In general, different design choices may aim to capture different kinds of generalizations or facilitate different kinds of processing, and may make parsing raw text easier or harder. It is often not obvious which framework should be chosen for best results on an external task—or indeed, how to even perform a controlled comparison across frameworks.

In this paper we investigate whether structurally guided language modeling can serve as a benchmark task for directly comparing linguistic representations. Specifically, we evaluate on next-word prediction—a relatively neutral task in that it does not rely on any artificial test suite, nor does it target a specific downstream application where one linguistic framework may have an advantage.[1]

We devise a method for selecting and encoding partial views of linguistic graphs over the preceding context relevant to predicting the next token (§3 and §4).[2] We call these views *slices* (small per-token graphs and dashed lines in figure 1). Our neuro-symbolic encoder statically allocates distinct vector dimensions for different structural

[1] Our findings are limited to a particular *language* (English) and *domain* (financial news) in which gold graphs from multiple frameworks are available for the same sentences, but such annotations could be obtained for other samples in the future.

[2] Our code is available to the research community at https://github.com/jakpra/LinguisticStructureLM.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4375 - 4391

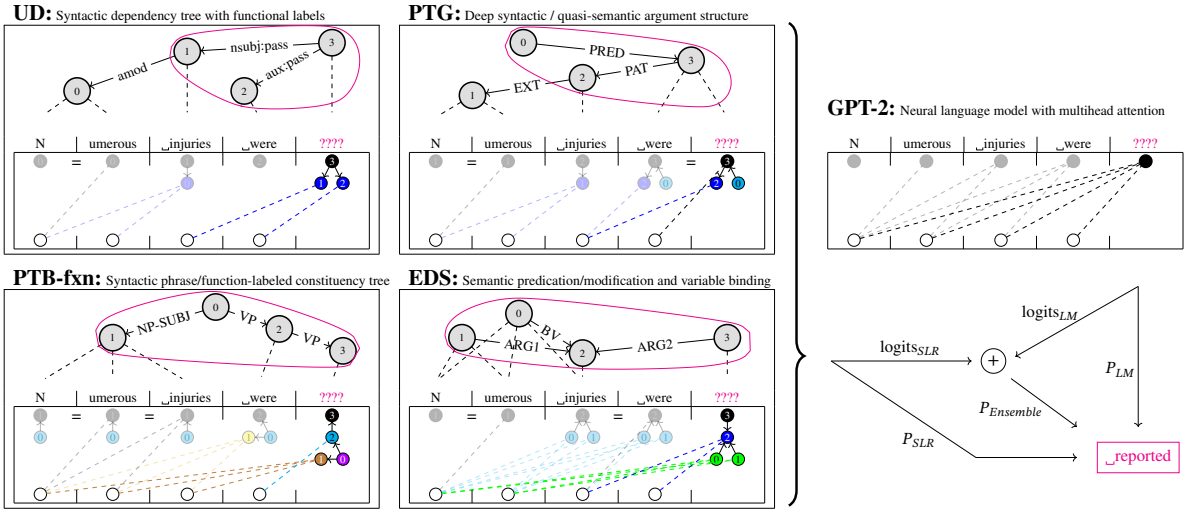

Figure 1: Contrasting GPT-2's incremental attention mechanism (top right) with incremental context *slices* obtained from linguistic graphs (left four panels) of four different formalisms (§5.2). As shared tokenization we use GPT-2's byte-pair encoding. Slice nodes are color-coded by local relation type (black: target, cyan: parent, blue: child, green: coparent, yellow: sibling, purple: grandparent, brown: aunt). Dashed lines indicate the token anchoring of original (big grey) graph nodes and, correspondingly, which previous tokens (empty circles) are accessible for each next-token prediction. In the bottom right we visualize how different models arrive at their prediction (§3 and §4.3).

relations within each slice, which in the incremental setting is much faster and more flexible than computation-intensive deep graph encoders. Using this encoding, we compare 7 SLR formalisms by virtue of their incremental language modeling capability in an controlled experimental setup (§5) on jointly annotated ground-truth data (Oepen et al., 2019, 2020). The results (§6) suggest that linguistic graphs are indeed informative for next-word prediction, complementing what is learned in pretraining. This invites future research quantifying different formalisms' design choices (§7).

2 Background: Symbolic Linguistic Representation

Following a long tradition in formal linguistics, graph-structured representations of language qualitatively describe grammatical and logical relations among words. The SLR paradigm has recently seen a revival in the form of larger-scale *treebanking* and *sembanking* for training neural parsers.

Formally, an SLR instance is a directed acyclic graph (DAG) $G = \langle V, E, \alpha \rangle$, with vertices V, labeled edges E, and an anchoring function $\alpha : V \rightarrow \mathbf{w}$ that maps each vertex to a (potentially empty) subset of tokens in the sentence. We broadly distinguish SLR frameworks along two dimensions:[3]

Scope. A main goal of *syntactic* representations is 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 are relatively shallow, while the latter contain abstract nodes with no or multiple word anchors.

3 Overview: Language Modeling with Linguistic Graphs

Our main goal is to quantify the predictive power of different SLRs by combining them with a pretrained language model and measuring how this affects next-token generation performance. 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 sequences of word tokens, $\mathbf{w} = \langle w_0, w_1, \ldots, w_{n-1} \rangle$. An *incremental* LM factorizes the joint probability of the sentence in terms of the probability of each word w_i conditioned on previous tokens $\mathbf{w}_{<i}$; eq. (1).

Here we describe at a high level how we process (oracle) SLR graphs for use in this language modeling scenario, i.e., to obtain context-conditional

[3] See Abend and Rappoport (2017); Koller et al. (2019); Prange et al. (2019b) for more detailed taxonomies.

vocabulary distributions from them. In contrast to 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; eq. (2).

$$P_{LM}(\mathbf{w}) = \prod_{i=0}^{n-1} P_{LM}(w_i|\mathbf{w}_{<i}) \quad (1)$$

$$P_{SLR}(\mathbf{w}) := P(\mathbf{w}|G) \quad (2)$$

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 procedure is as follows:

First, we select for each token position i to be predicted a subgraph G_i, called the token's **slice**. Slices are both *admissible* in the language modeling setting, i.e., they do not violate the left-to-right conditioning order, and *relevant* to the token prediction according to some criteria—here we consider criteria based on structural relationships generally, without relying on formalism-specific labels (§4.1). Consider the small colored subgraphs for each token in figure 1: the EDS-slice for the target 'reported', for example, 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 $\mathbf{w}_{<i}$ are contained in G_i, to the extent that they are anchored in a node reachable from w_i. Inspired by Markov assumptions of independence in generative modeling and Markov blankets in causal networks, SLR graph slicing thus allows us to factorize $P(\mathbf{w}|G)$ as

$$P(\mathbf{w}|G) := \prod_{i=0}^{n-1} P(w_i|G_i). \quad (3)$$

Next, we **encode** each graph slice as a fixed-sized vector. Prior approaches to encoding linguistic graphs for neural modeling have involved serialization, e.g., as parser transition sequences (Qian et al., 2021, *inter alia*), recursive auto-encoders (Tai et al., 2015; Roth and Lapata, 2016), and graph-convolutional networks (GCNs; Yang and Deng, 2020; Wu et al., 2021). However, transition sequences for non-tree graphs are subject to spurious ambiguity; and we find that graph-structured neural networks are impractical in the incremental setting (§6.5). Instead, we propose a computationally inexpensive method for statically and deterministically projecting slices into a high-dimensional space by vector concatenation (§4.2).

Finally, we compute output **distributions** $P(W_i|G_i)$ from the vector representations (§4.3).

4 Modeling Details

4.1 Slicing Graphs

A slice G_i is a connected 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). G_i always minimally consists of w_i's *direct anchor* node $a_i = \text{Select}(\{v : w_i \in \alpha(v)\})$. Starting from a_i, we traverse the graph and add vertices and edges that are connected to a_i via paths of a few specific relative types, REL. Here we settle on 6 types: parents, siblings, grandparents, parents' siblings, children, and coparents. The vertices V_i and edges E_i for slice $G_i = \langle V_i, E_i, \alpha \rangle$ consist then of the union of these sets.[4]

To prevent information leakage from future tokens, we discard from G_i all nodes $\{v : \alpha(v) = w_j, j > i\}$ which are *only* anchored in tokens *following* w_i. 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 following token 'reported' (and thus the sibling 1 cannot be accessed either). If a node's anchors *contain or overlap with* w_i (i.e., the node is a non-terminal above w_i), we retain the node and its edges but remove its token anchors.

4.2 Vectorizing Graph Slices

Because slices can be large, we partition each slice's nodes by structural *relative type*, in order to aggregate them into a fixed-length summary vector. Specifically, we allocate capacities for each relative type: $\gamma_{\text{rel}} = 2$ for parents, siblings, aunts, and children, and 1 for grandparents and coparents. Up to $\gamma - 1$, relative nodes V_{rel} are added 'with high resolution', maintaining their identity and order; beyond the capacity, relatives are aggregated 'with low resolution'; eq. (4). Within each relative type, precedence k is given to relatives whose token anchors are sequentially closer to w_i.

[4] See appendices A.1 and A.2 for details.

$$\begin{aligned}\text{HiRes}_{i,\text{rel}} &= \langle v_{\text{rel},k} : k < \gamma_{\text{rel}} \rangle \\ \text{LoRes}_{i,\text{rel}} &= \{v_{\text{rel},k} : k \geq \gamma_{\text{rel}}\}\end{aligned} \tag{4}$$

Next we look up the relatives' edge label and word vector encodings[5] $\vec{l}_k$ and $\vec{w}_k$ and collate them into a single vector $\vec{s}_{i,\text{rel}}$ per relative type. High-resolution vectors are concatenated $\bigoplus$ and low-resolution vectors are averaged; eq. (5). Finally, we concatenate all of these (zero-padded) relative-vectors to obtain the final vector representation of the whole slice, $\vec{s}_i$; eq. (6). At a high level, this vector essentially specifies a deterministic, structured, typed, discrete self-attention over the token history.

$$\begin{aligned}\vec{s}^{\,\text{HiRes}}_{i,\text{rel}} &= \bigoplus_{k \in \text{HiRes}_{i,\text{rel}}} \left[\vec{l}_k; \vec{w}_k\right] \\ \vec{s}^{\,\text{LoRes}}_{i,\text{rel}} &= \sum_{k \in \text{LoRes}_{i,\text{rel}}} \frac{\left[\vec{l}_k; \vec{w}_k\right]}{|\text{LoRes}_{i,\text{rel}}|_+}\end{aligned} \tag{5}$$

$$\vec{s}_i = \bigoplus_{\text{rel} \in \text{REL}} \left[\vec{s}^{\,\text{HiRes}}_{i,\text{rel}}; \vec{s}^{\,\text{LoRes}}_{i,\text{rel}}\right] \tag{6}$$

4.3 Predicting Emission Distributions

We compute model posteriors for next-token predictions as

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

where μ is either a pure SLR model or LM, or an ensemble of the two (bottom right of figure 1).

SLR only. As described above, we define $\text{context}_{i,SLR}$ as G_i, which is encoded as $\vec{s}_i$. We obtain P_{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.

$$\begin{aligned}\text{logits}_{i,SLR} &= \text{MLP}_d(\vec{s}_i) \\ \text{MLP}_d(x) &= H^{(d)}\left(\ldots H^{(1)}(x)\right)\text{Emb}^\top,\end{aligned}$$

where 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 $P_{Ensemble}$, where logits obtained from slice-encodings are added to a base neural LM's logits before taking the softmax:

$$\begin{aligned}\text{logits}_{i,Ensemble} &= \text{logits}_{i,SLR} + \text{logits}_{i,LM}, \\ \text{with} \quad \text{logits}_{i,LM} &= \text{LM}(\mathbf{w}_{<i}).\end{aligned}$$

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

[5] See appendix A.3 for details.

	Sentences	Tokens	Vocabulary
Train	26,325	658,475	27,344
Train (EarlyStop)	23,692	591,829	26,422
Dev (EarlyStop)	2,633	66,646	10,073
Eval	921	22,596	5,364

Table 1: Data statistics.

5 Experimental Setup

All models are implemented in PyTorch and experiments are run on 1 NVIDIA Tesla T4 GPU. Model hyperparameters are reported in appendix A.5.

5.1 Data

Our dataset consists of the intersection of Wall Street 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)[6] as well as a range of semantic representation formalisms for the MRP 2019 & 2020 shared tasks (Oepen et al., 2019, 2020). Summary statistics are shown in table 1. Our preprocessing steps are described in appendix B.

5.2 SLR Formalisms

The 7 (versions of) linguistic representation frameworks examined in this study are listed in table 2, along with their classifications along the scope and structure dimensions. We draw the structural dependencies vs. constituencies distinction (described at a high level in §2) based on specific properties of the MRP shared task data: a framework is considered a dependency framework if all edges are only between pairs of individual word anchors at a time; if there are any unanchored[7] nodes or nodes anchored in more than one linguistic word token, it is considered a constituency framework.[8] Below we give a brief description of each framework.

PTB trees specify hierarchically nested syntactic constituents. We consider two labeling variants: basic phrase structure (**-phr**) and phrase types refined with functional specifications (**-fxn**).

[6] https://catalog.ldc.upenn.edu/LDC2013T19

[7] Not including "ROOT" nodes in UD.

[8] See appendix A.4 for details.

Universal Dependencies (**UD**; Nivre et al., 2016, 2020; de Marneffe et al., 2021) is a syntactic dependency representation with coarse, cross-linguistically applicable edge labels.

DELPH-IN MRS Bi-Lexical Dependencies (**DM**; Ivanova et al., 2012) and Elementary Dependency Structures (**EDS**; Oepen and Lønning, 2006) are derived from underspecified logical forms computed by the English Resource Grammar (Flickinger, 2000; Copestake et al., 2005).

Prague Semantic Dependencies (**PSD**; Hajič et al., 2012) and Prague Tectogrammatical Graphs (**PTG**) are syntactico-semantic predicate–argument structures converted from the Prague Functional Generative Description (Sgall et al., 1986; Böhmová et al., 2003; Hajič et al., 2012).

5.3 Language Model

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

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.

We perform early stopping with the last 10% of the original training corpus set aside for development scoring after each epoch.[9] We keep the model state that achieves the best perplexity on the dev set. Peak development performance is reached after ≈3 epochs for SLR models, whereas finetuning GPT-2 by itself takes between 7 and 9 epochs.

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 of the correct answer (MRR), and entropy of the model's token prediction posterior (H). All metrics are reported as microaverages over the evaluation data at the BPE token level.[10]

6 Findings

6.1 Main Results

The most striking observation in terms of overall model performance (table 2) 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 2). The large performance gap suggests that at least a subset of the oracle knowledge about linguistic structure is **not yet encoded** in the base language model, which learns from only raw text.

We observed that if we keep training for the entirety of 10 epochs, rather than early stopping based on development performance, we somewhat overfit to the training set. While accuracy itself is not affected very much by this, the models become increasingly overconfident ($\frac{\text{overall confidence}}{\text{overall accuracy}}$, which gets up to 8–12%, compared to ≈4% with the vanilla GPT-2 model and in most cases even slightly less than that with the early-stopped SLR models). This leads to overall worse perplexity.

6.2 Differences between Formalisms

Comparing across rows in table 2, we find a considerable performance spread. The general trend, which is relatively consistent in all metrics,[11] is indicated by the order of rows, with UD having the smallest (though still respectable) improvement over the baseline, and PTG 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 (more abstract) formalisms. This is summarized

[9] The R-GCN baseline (table 6) is always trained for the full 10 epochs, but to ensure fairness, it is also only compared to concatenation-based encoders that have been trained for the full 10 epochs, too.

[10] We compute average PPL over all sentences j by exponentiating last: $\exp\left(\frac{1}{\sum_j |\mathbf{w}_j|}\sum_j\sum_{i=0}^{|\mathbf{w}_j|-1} -\log P_\mu\left(w_{ji} = v^*_{ji}|\text{context}_{ji,\mu}\right)\right)$

[11] The multitude of metrics might thus seem redundant. But since each measurement emphasizes different properties of model performance, we consider it a a very interesting result (and, potentially, a success of our modeling technique and experimental setup) to achieve this broad consistency.

				Training Efficiency		Language Model Quality				
Model	**Scope/Struct**		**#Labels**	**Speed** ↑	**Size** ↓	**PPL** ↓	**H** [nats] ↓	**Acc** [%] ↑	**Conf** [%] ↑	**MRR** ↑
GPT-2				–	124.4M	59.3	4.09	30.0	31.2	.403
+ Domain				15		45.9 ±.09	3.64 ±.008	33.3 ±.02	34.9 ±.07	.435 ± .3e-3
						***		***		
+ UD	syn	dep	39	14	+54.1M	32.7 ±.18	3.30 ±.013	39.1 ±.15	**40.1** ±.14	.486 ±1.2e-3
						***		–		
+ DM	sem	dep	59	15	+54.4M	31.4 ±.08	3.24 ±.026	38.9 ±.10	40.2 ±.37	.491 ± .6e-3
						***		–		
+ PSD	sem	dep	90	16	+54.9M	30.7 ±.09	3.21 ±.014	39.1 ±.11	40.9 ±.11	.491 ± .5e-3
						**		***		
+ PTB-phr	syn	const	38	14	+54.1M	29.8 ±.18	3.14 ±.029	41.2 ±.19	42.8 ±.42	.507 ±1.3e-3
						***		*		
+ PTB-fxn	syn	const	537	14	+62.7M	29.0 ±.28	3.07 ±.049	42.0 ±.30	43.8 ±.60	.514 ±1.8e-3
						***		*		
+ PTG	sem	const	72	15	+54.6M	26.8 ±.26	3.03 ±.041	**43.1** ±.12	44.6 ±.51	.522 ± .9e-3
						***		–		
+ EDS	sem	const	10	15	+53.6M	**24.7** ±.28	**2.92** ±.048	**43.1** ±.17	45.0 ±.55	**.527** ±1.3e-3

Table 2: Main results: performance of language models combined with 7 SLR formalisms of different scope, structure, and label set (each corresponding to a $P_{Ensemble}$ in §4.3), compared to vanilla GPT-2 and a version of GPT-2 that has been domain-finetuned on the raw text of the SLR training corpus (P_{LM}). We report each quality metric as mean ± stdev over 3 random seeds. We also report model size in #parameters (all non-baseline models as absolute difference to baseline) and training speed in sentences per second as measures of efficiency. Statistical significance of the PPL and Acc differences to the next-best model (always adjacent rows) is reported as $***p < .0001$ / $**p < .001$ / $*p < .005$ / –not significant (approximate randomization test as described in Riezler and Maxwell (2005), with R =10,000 shuffles). We only consider a difference significant if $p < \alpha$ for all three random model initialization seeds. Best results in each column are **bolded**. For confidence, 'best' means best-calibrated, i.e., the smallest relative difference to accuracy.

	Dep			**Const**	**Avg**
Syn	32.7 (1)	***		29.4 ±0.6 (2)	30.5 ±2.0 (3)
	***			***	–
Sem	31.0 ±0.5 (2)	***		**25.7** ±1.5 (2)	28.4 ±3.2 (4)
Avg	31.6 ±1.0 (3)	**		27.6 ±2.3 (4)	29.3 ±2.8 (7)

Table 3: Model perplexity (lower is better) summarized in terms of two SLR dimensions: Scope (syntax vs. semantics) and structure (dependency vs. constituency). $\mu_{\pm\sigma}(n)$ over frameworks per condition. Statistical significance of the difference between the two closest SLRs of each pair of conditions is reported as $***p < .0001$ / $**p < .001$ / $*p < .005$ / –not significant (approximate randomization test with R =10,000 shuffles).

in table 3. 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. Further, the syntactic graphs we use are always trees, so they never populate the coparent capacity in the slices. *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 that the function-enhanced PTB label set has a slight advantage over the basic phrase-structure labels; and that, among the two closely related pairs of formalisms (DM/EDS and PSD/PTG, which each are dependency and constituency versions converted from the same underlying grammars), the constituency versions always work better than the dependency versions in our setting. There is, however, no consistent ranking between DM/EDS on one hand and PSD/PTG on the other. In terms of perplexity, EDS works better than PTG, and PSD better than DM, but these differences are not significant for accuracy.

6.3 Differences between Word Classes

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 (POS) tag (table 4, see appendix C for more details). Across all models there is a clear and expected separation between rather predictable function words, more perplexing content words, and 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 UD in particular do not raise performance. Only EDS and DM show perplexity improvements on nouns; PTB even has a noticeable negative impact. 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 POS: UD for verbs; PTB and PTG for adpositions and subordinating conjunctions; EDS for pronouns, determiners, and numbers; PTG, PSD, and EDS for coordinating conjunctions. The advantage of EDS and DM on nouns, pronouns, determiners, and numbers can likely be attributed to their explicit representation of variable binding/quantification. Similarly, PTG and PSD have detailed categories for coordination, distinguishing, e.g., con- and disjunction.

For nouns and modifiers, the spread across formalisms is particularly wide, which suggests that SLRs diverge quite a bit on these types of words (e.g., whether adjectives and certain nouns can count as predicates) and that this diversity has a strong effect on utility for language modeling.

6.4 Model Ablations

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 graph properties: (i) the assignment of edge *labels*, and (ii) the *anchoring* mapping between graph nodes and word tokens in each graph. If the models are largely independent of the correct label and structure assignments, these changes should have a very small effect on 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 5, see appendix C 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 linguistic graphs' entire structural arrangement of labels and attention-like selection of context words play a crucial role—more so than knowing the type of each individual (correctly attached) grammatical relations. Note that the ⤮ Anchors setting, too, changes which edge labels are used in the predic-

[12] https://universaldependencies.org/u/pos/

	POS	Eval Toks	Train Vocab	Perplexity↓ GPT-2	UD	EDS	SLR Avg
	All	22,596	27,344	45.9	32.7	**24.7**	29.3 ± 2.8
content	noun	7,731	18,435	142.5	122.0	**98.0**	122.6 ±13.9
content	verb	2,639	7,100	128.8	**80.4**	85.9	84.9 ± 4.5
content	mod	2,235	6,292	228.7	158.8	**98.6**	124.4 ±22.6
function	aux	582	95	17.6	11.1	**5.9**	9.1 ± 2.1
function	adp	1,957	232	10.1	7.3	5.5	**5.3** ± 1.6
function	part	645	27	3.7	2.0	**1.6**	1.9 ± 0.3
function	sconj	268	96	15.4	12.3	6.8	**6.8** ± 3.9
function	cconj	548	35	13.0	7.4	**1.9**	4.1 ± 2.1
function	det	1,726	91	9.4	7.8	**4.4**	6.0 ± 1.3
function	pron	868	149	22.9	17.5	**5.4**	11.0 ± 4.0
	num	719	1,059	72.6	57.1	**47.5**	54.1 ± 4.6
	punct	2,527	68	4.9	**2.3**	2.7	2.6 ± 0.3
	misc	151	183	7.0	4.6	**4.0**	4.5 ± 0.8

Table 4: Breakdown by Universal POS,[12] in terms of PPL of domain-trained GPT-2, two exemplary SLR-combined models, and the macro-average ± stdev over all SLR-combined models. Best results (within the variance) in each row are **bolded**. We show token counts and observed vocabulary size for reference.

Ablation	Applied in	DM	PTB	SLR Avg
Full		31.4	29.0	29.3 ± 2.8
⤮ Labels	testing	+4.7	+73.9	+28.3 ±28.3
⤮ Anchors	testing	+34.8	+223.1	+106.0 ±73.1
⤮ Both	testing	+33.4	+207.4	+95.9 ±68.9
⤮ Labels	training	+1.4	+9.0	+4.2 ± 3.3
⤮ Anchors	training	+8.5	+17.5	+13.3 ± 4.6
⤮ Both	training	+7.8	+18.3	+13.4 ± 5.0
⤮ Labels	both	+1.3	+9.3	+4.3 ± 3.4
⤮ Anchors	both	+7.9	+17.5	+13.5 ± 5.0
⤮ Both	both	+7.3	+18.1	+13.6 ± 5.2
− SLR	both	+14.5	+16.9	+16.6 ± 2.8

Table 5: Ablations measured in ΔPPL for two exemplary SLR-combined models and the macro-average ± stdev over all SLR-combined models. *Full* and −SLR correspond, respectively, to table 2's rows 4 (DM) / 7 (PTB-fxn) and row 2 (GPT-2 +Domain).

	Training Efficiency		LM Quality	
Model	**δSpeed ↑**	**ΔSize ↓**	**ΔPPL ↓**	**ΔAcc ↑**
UD	−50%	−1.9M	+2.9 ±.08	−0.4 ±.02
DM	−47%	+2.5M	+1.6 ±.35	+0.1 ±.14
PSD	−56%	+9.1M	+3.6 ±.23	−0.9 ±.15
PTB-phr	−43%	−1.9M	+6.8 ±.39	−1.7 ±.11
PTB-fxn	−86%	+107.7M	+10.5 ±.21	−2.7 ±.15
PTG	−53%	+5.9M	+6.2 ±.22	−3.5 ±.03
EDS	−47%	−8.5M	−0.2 ±.07	+0.1 ±.03

Table 6: Performance differences between R-GCN slice encoder baseline and our concatenation-based encoder (table 2). Relative differences (δ) for speed in sentences per second; absolute differences (Δ) otherwise. Means ± stdev over 2 runs without early stopping.

tion of a given token, resulting in a smaller difference between ⤮ Anchors and ⤮ Both.

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 learned in pretraining (Linzen et al., 2016; Tenney et al., 2019b), we conjecture that this representational capacity gets offloaded to the graphs at training time, and thus test-time permutations 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, the model learns to mostly disregard the random structure as noise.

6.5 Comparison with R-GCN Encoding

As an additional strong baseline, we compare our concatenation-based slice vector encoding to a graph neural network from the literature. We choose relational graph-convolutional networks (R-GCN; Schlichtkrull et al., 2018; Kipf and Welling, 2017) as a suitable representative of this type of model, which has been used successfully by Wu et al. (2021) to encode DM graphs.

Results are shown in table 6. Contrasting with table 2, there is a big difference in training speed: our simple encoder is on average roughly twice as fast as the computation-heavy alternative, whose time and space complexity is dominated by the number of labels.[13]

We observe at best similar LM quality as with our concatenation method (EDS and DM), but for most formalisms performance degrades. We follow Schlichtkrull et al. and Wu et al. in using 2 R-GCN layers with basis matrix regularization. Possible disadvantages of this for encoding linguistic graphs are the fixed path length (2 layers exclude parent's siblings; but 3 layers would include a lot of irrelevant information) and that many of the trained parameters are shared between different relations. In contrast, our concatenation encoding forces the MLP input layer to learn distinct parameters for each structural relative type and edge label.

7 Discussion

7.1 Related Work

Researchers have long been interested in scaffolding sequential language models with linguistic-structure-based inductive biases. *Syntactic language modeling* dates back to the pre-neural era, when Pauls and Klein (2012) and Gubbins and Vlachos (2013) generalized Markov assumptions from word n-grams to syntactic subtrees. These ideas have since been adapted to recurrent neural network (RNN) LMs (Mirowski and Vlachos, 2015) and expanded on (Dyer et al., 2016; Choe and Charniak, 2016; Shen et al., 2018, 2019). Ek et al. (2019) condition RNN-LMs on predicted syntactic and semantic (unstructured) tags, interestingly finding less or sometimes no benefit, especially on the semantic side. They hypothesize this might be due to tagging errors—an issue our oracle setup avoids.

In the era of attention-based neural modeling of language dominated by pretrained Transformers, models are often finetuned for and evaluated on specific NLP tasks—like semantic role labeling, machine translation, natural language inference, graph-to-text generation, or the GLUE benchmark (Wang et al., 2019)—rather than language modeling in its own right, which makes it difficult to compare them directly to our findings. There have been two main directions: One group of approaches continues the old syntactic language modeling tradition by incrementally generating words and SLRs with either joint (Peng et al., 2019; Qian et al., 2021; Sartran et al., 2022) or iteratively-coupled LM and parser models (Choshen and Abend, 2021). The second group assumes parsed input sentences,

[13] And this is a very optimistic estimate of R-GCN training speed in practice; see appendix A.6.

which are then used to guide the model, e.g. 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, Hajdik et al. (2019) process serialized graphs directly with a sequence-to-sequence model, and Wu et al. (2021) extend a pretrained Transformer with an additional graph encoder. 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.

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*). Among the probing literature, the works of Kuznetsov and Gurevych (2020) and Kulmizev et al. (2020) are noteworthy in that they investigate subtle differences between different (versions of) frameworks roughly covering the same representational scope, namely, semantic roles and syntactic dependencies, respectively.

Orthogonal approaches to *comparing SLR designs* have involved measuring how well different frameworks complement each other for joint parsing or can be merged or converted into one another (Prange et al., 2019a; Hershcovich et al., 2020).

7.2 Limitations and Future Work

While the use of oracle graphs has both theoretical advantages (measuring an upper bound without needing to account for potential errors or uncertainties) and practical ones (saving the computational overhead from training and running a parser), ground-truth SLR graphs are a very limited resource and generally assumed to only be available at training time. There is no guarantee our results translate to the non-oracle setting. For instance, it could be that the most helpful abstract semantic information is also the hardest to predict. And despite segmenting the existing sentence-level graph into token-level slices, the human annotator who created the graph in the first place has seen and analyzed the whole sentence, thus already resolving crucial ambiguities and simplifying the task based on knowledge 'from the future'. In subsequent work, we plan to *parse* graph slices incrementally, which will both relax the *conditional* modeling assumption into a more broadly interpretable *joint* model and enable test-time use of the full system on datasets without linguistic annotations.

We also only test formalisms that are explicitly anchored in linguistic units, roughly corresponding to LM (sub-)word tokens. 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.

7.3 Broader Impact

Our experiments yield evidence which—at least in the case of encoding contexts for next-word prediction—supports the thesis of Bender and Koller (2020), Trott et al. (2020), and others that linguistic *meaning* goes beyond *form*. Computational models of language that exclusively learn from even very large amounts of raw text are thus generally expected to hit a ceiling[14] 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 various frameworks have the potential to aid a pretrained incremental Transformer in task-neutral next-word prediction. To this end, we have proposed a framework-agnostic neural encoding scheme for linguistic graphs and applied it to an English dataset jointly annotated with 7 different formalisms. The results highlight the importance of appreciating complex linguistic structure and handling its computational representation with nuance.

[14] See also Merrill et al. (2021) for formal proofs.

Acknowledgements

We would like to thank Katrin Erk and Chris Dyer; members of the Georgetown NERT/GUCL and HKU NLP labs; the organizers, reviewers, and audience of MASC-SLL 2022; as well as the anonymous ARR reviewers for their extremely insightful feedback and suggestions.

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 *Proc. of EACL*, 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 LAW-ID*, 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 Hajič, Eva Hajičová, 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.

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.

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.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

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 Weinshall. 2018. Coming to your senses: on controls and evaluation sets in polysemy research. In *Proc. of EMNLP*, 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 *Proc. of NAACL-HLT*, pages 199–209, San Diego, CA, USA.

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 effcient 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 *Proc. of NeurIPS*.

Joseph Gubbins and Andreas Vlachos. 2013. Dependency language models for sentence completion. In *Proc. of EMNLP*, pages 1405–1410, Seattle, Washington, USA. Association for Computational Linguistics.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M. Bender. 2019. Neural text generation from rich semantic representations. In *Proc. of NAACL-HLT*, pages 2259–2266, Minneapolis, Minnesota. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proc. of LREC*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Daniel Hershcovich, Nathan Schneider, Dotan Dvir, Jakob Prange, Miryam de Lhoneux, and Omri Abend. 2020. Comparison by conversion: Reverse-engineering UCCA from syntax and lexical semantics. In *Proc. of COLING*, pages 2947–2966, Barcelona, Spain (Online). International Committee on Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proc. of EMNLP-IJCNLP*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proc. of NAACL-HLT*, pages 4129–4138, Minneapolis, MN, USA.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proc. of HLT-NAACL*, pages 57–60, New York City, USA.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In *Proc. of LAW*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proc. of *SEM*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of ICLR*.

Alexander Koller, Stephan Oepen, and Weiwei Sun. 2019. Graph-based meaning representations: Design and processing. In *Proc. of ACL: Tutorial Abstracts*, pages 6–11, Florence, Italy. Association for Computational Linguistics.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proc. of ACL*, pages 4077–4091, Online. Association for Computational Linguistics.

Ilia Kuznetsov and Iryna Gurevych. 2020. A matter of framing: The impact of linguistic formalism on probing results. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 171–182, Online. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *TACL*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL-HLT*, pages 1073–1094, Minneapolis, Minnesota.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proc. of ICLR*, New Orleans, LA, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. 2021. Provable Limitations of Acquiring Meaning from Ungrounded Form: What Will Future Language Models Understand? *TACL*, 9:1047–1060.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proc. of ACL-IJCNLP*, pages 511–517, Beijing, China. Association for Computational Linguistics.

Stefan Müller. 2020. *Grammatical theory: From transformational grammar to constraint-based approaches*. Language Science Press.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: a multilingual treebank collection. In *Proc. of LREC*, pages 1659–1666, Portorož, Slovenia.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proc. of LREC*, pages 4034–4043, Marseille, France. European Language Resources Association.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proc. of MRP at CoNLL*, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-framework meaning representation parsing. In *Proc. of MRP at CoNLL*, pages 1–27, Hong Kong. Association for Computational Linguistics.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proc. of LREC*, 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 *Proc. of 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.

Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. 2021. Structural guidance for transformer language models. In *Proc. of ACL-IJCNLP*, 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.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan. Association for Computational Linguistics.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of ACL*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer Grammars: Augmenting Transformer language models with syntactic inductive biases at scale. ArXiv: 2203.00633.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

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

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *Proc. of ICLR*.

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.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*, pages 3772–3782, Brussels, Belgium.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured Long Short-Term Memory networks. In *Proc. of ACL-IJCNLP*, pages 1556–1566, Beijing, China.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*, 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 *Proc. of ICLR*.

Sean Trott, Tiago Timponi Torrent, Nancy Chang, and Nathan Schneider. 2020. (Re)construing Meaning in NLP. In *Proc. of ACL*, pages 5170–5184, Online. Association for Computational Linguistics.

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 *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 *Proc. of ICLR*.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021. Infusing Finetuning with Semantic Dependencies. *TACL*, 9:226–242.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proc. of ACL*, pages 4166–4176, Online. Association for Computational Linguistics.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Proc. of NeurIPS*, volume 33, pages 21687–21698. Curran Associates, Inc.

Zdeněk Žabokrtský, Daniel Zeman, and Magda Ševčíková. 2020. Sentence meaning representations across languages: What can we learn from existing frameworks? *Computational Linguistics*, 46(3):605–665.

A Additional Modeling Details

A.1 Selecting Anchor Nodes

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: Select the anchor node with the most parents and children; if still a tie, select the anchor with the highest node ID (tends to be hierarchically lower, i.e., vertically closer to the token anchor).

A.2 Relative Types

$REL = \langle P, B, O, T, C, R \rangle$, 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). This is the anchor node's Markov blanket, plus siblings, grandparents, and aunts. We chose this set of relations based on general notions of linguistic hierarchy (predicate-argument, head-dependent) and preliminary experiments, but without tuning for specific formalisms. Precise definitions are given in table 7. Relative nodes are permanently associated with the label of the edge that got them selected.

A.3 Representing Tokens and Labels

We use GPT-2's pretrained global embeddings (from the lowest layer, before any local contextualization) to obtain embeddings for relative token anchors in the slice-vector. When a token anchor in a linguistic graph consists of multiple BBPE tokens, we average their embeddings. We reuse the transpose of the same embedding matrix again to project the last hidden state of the token-emission MLP into the vocabulary.

SLR edge labels are encoded as one-hot vectors in the slice vectors, which lowers the potential for unnecessary random initialization variance of from-scratch embeddings.

A.4 Distinguishing Dependencies from Constituencies

While this distinction—as defined in §5.2 in terms of the anchoring mapping between graph nodes and word tokens—can be subtle for individual sentences, it nonetheless affects slice encoding. In PSD, for example, auxiliaries are unanchored, whereas in PTG they are grouped with their main predicate (figure 2).

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

Table 7: Relative types and capacities.

A.5 Model Hyperparameters

We report our model and training hyperparameters in table 8. We did *not* perform explicit hyperparameter tuning, besides some manual testing early in development on a subset of the MRP shared task data. Those data are annotated with SLR frameworks other than the ones we compare here, and we ended up excluding them from our experiments for lack of overlap with most of the other frameworks' annotations.

A.6 Efficient Batching for R-GCN

In our incremental setting we need to apply the R-GCN to each token-level slice, which would lead to multiple days[15] of training for each model if done naively. We achieve a considerable speedup by exploiting the oracle graphs at training and evaluation time to pre-compute slices and running the R-GCN only once per sentence batch.

B Data Preprocessing

B.1 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.[18] We then select only those 35,513 train-dev / 1,401 eval sentences that appear in both the MRP 2019 and 2020 datasets (the 2019 corpus contains 143/1,958 more in train-dev/eval).[19] Next,

[15] Projected timeline based on a few iterations, which is confirmed by Yang and Deng (2020).

[16] For label set L. The factor 16 arises from the capacities chosen (table 7), and the extra embedding allocation is for averaged preceding unanalyzable/within-anchor tokens.

[17] For bidirectional label set L^*, which is twice as big as L.

[18] The sentence "It is." in DM and a 'sentence' consisting of the @-symbol in PTG.

[19] 'train-dev' refers to the data split that was used as training data in both the MRP and 2019 tasks, and which we split 90%/10% into our training and development data. 'eval' refers to the data that was used as evaluation data in MRP 2019 and as development data in MRP 2020, and which we evaluate our

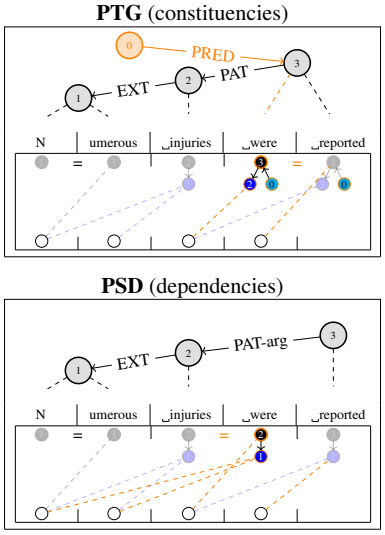

Figure 2: Example of subtle differences in constituency (PTG) and dependency (PSD) versions of the same underlying formalism, the Prague Functional Description. PTG has an abstract PRED node as well as a multiword anchor where PSD does not, which results in diverging slice representations for the last two tokens.

GPT-2	
Embedding dim	768
Vocabulary	50,257
Activation	GELU
Dropout	0.1
Learning rate	1e-6
MLP	
Input dim	$16 * \|L\| + 17 * 768$[16]
Layers	2
Hidden dims	1,024; 768
Activation	ReLU
Dropout	0.2
Learning rate	1e-4
R-GCN	
Input dim	768
Layers	2
Hidden dims	768; 768
Activation	ReLU
Basis matrices	$\lfloor 0.1 * \|L^*\| \rfloor$[17]
Learning rate	1e-4
Other training settings	
Epochs	10
Batch size	8

Table 8: Model and training hyperparameters

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 train-dev/eval 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-dev / 8 eval sentences, which leaves us with the final 26,325 train-dev and 921 eval sentences.

In a few cases, where the linguistic graph has no edges, we add an artificial edge with a dummy label.

B.2 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.

B.3 UD Conversion

Quasi-gold UD 2.0 trees are obtained from the UD converter released with the Java Stanford Parser v4.2.0 (`https://nlp.stanford.edu/software/lex-parser.html`) on the PTB trees.

B.4 PTB Labels

By convention, phrasal and functional labels in PTB are node labels. To match the labeled-edges-unlabeled-nodes format of the other formalisms, we losslessly convert them to edge labels (namely, on each node's single incoming edge), discarding the preterminal nodes' POS labels. In preliminary experiments we saw that including the POS tags is much more beneficial than phrase structure only; but since we do not include word-level tags in any of the other conditions, this would be an unfair comparison. We focus here on sentence-level structure and leave studies of word-level tags to future work.

B.5 Data Splits

We split the corpus into training/development and evaluation data following the MRP task setup. Specifically, we evaluate on the data split that was

models on.

used as evaluation data in MRP 2019 and as development data in 2020, as only for this data gold annotations in all formalisms have been released. We do not perform empirical hyperparameter tuning. In early development, a small subset of the data was used.

C Detailed Results

We report detailed results without early stopping (table 9), breakdowns by POS-class (table 10 and appendix C.1), as well as ablation experiments (table 11) for all SLR formalisms. In tables 4 and 10 and figure 3 we merge the POS tags {NOUN, PROPN} into 'noun', {ADJ, ADV} into 'mod', and {INTJ, SYM, X} into 'misc'.

C.1 Lexico-Semantic or Syntactic Knowledge?

In §6.3 we have found part-of-speech-specific patterns of model performance. But whenever, for a certain syntactic word class S, a formalism A is more conducive to next-word prediction than a formalism B, it is not clear whether this is the case because the choices get narrowed down to S itself or whether it is caused by either complementary or completely independent signals, perhaps at the lexical or semantic-structure levels.

We investigate this by rerunning the experiment with each token's UPOS tag as an additional input. If this is more or less the same information as is gained—to different extents—from the SLRs, then the results should be similar to before, and SLR-conditional differences should disappear.

A few particularly interesting POS subsets are shown in figure 3. We discuss them in order.

Among content words, nouns and verbs are similar both in terms of baseline performance and in how much easier it becomes to select the correct lexical item if the part-of-speech is known. At the same time, the individual SLR formalisms differ quite a lot in how much information they contribute about the POS class itself and about lexical choice within the part-of-speech. The respective best formalisms (EDS for nouns, PTB and UD for verbs) approximate oracle POS knowledge by themselves and still contribute substantial complementary information when the actual POS tag is revealed. In contrast, PTB does not seem to provide any useful signal about nouns to the incremental LM—neither independently nor in conjunction with the POS.

Modifiers (adjectives and adverbs) display a rather interesting behavior: the fact *that* a word of this type is coming next is very hard to predict from just the preceding raw context, which makes sense since they tend to add *optional* meaning on top of the (obligatory) logical and grammatical content. However, once the decision to modify has been made, the contextual choice becomes much easier than that for nouns or verbs. In both cases, all SLRs are quite helpful, with UD on the lower end and EDS leading the field.

We find similar tendencies among auxiliaries (≈function verbs) and pronouns (≈function nouns) as with (content) verbs and (content) nouns, but naturally at a much smaller scale. Despite their functional-grammatical distribution and behavior, the semantic frameworks EDS and PTG consistently outperform the syntactic ones UD and PTB even on these 'small' words. A possible explanation for this interaction with auxiliaries in particular could be that EDS and PTG do not analyze them separately at all, but rather group them, respectively, with the preceding context[20] or their main predicate. The models might be able to leverage this to focus on things like subject-verb agreement, local cohesion, or anticipating the main predicate. More explicit syntactic analyses of auxiliaries (incrementally inaccessible forward-pointing dependencies in UD; VP-nesting in PTB), in contrast, may restrict the model from directly making these connections. Adding POS information in the input decreases SLR-dependent differences.

For 'subordinators' in the broad sense, i.e., subordinating conjunctions at the clausal level and adpositions for nominal complements, PTB and PTG are particularly well-suited. By themselves they are already *at least* as informative as POS, and they still add a small but noticeable complementary signal when the POS is revealed.

Determiners and coordinating conjunctions, which both already show extremely low perplexity with some SLR models (namely, EDS, PSD, and PTG), entirely lose any reliance on particular SLRs when their POS is known.

[20] EDS, like PSD, actually has no anchors for auxiliaries; we attach them to the preceding semantic unit by default.

				Training Efficiency		Language Model Quality				
Model	**Scope/Struct**		**#Labels**	**Speed** ↑	**Size** ↓	**PPL** ↓	**H** [nats] ↓	**Acc** [%] ↑	**Conf** [%] ↑	**MRR** ↑
GPT-2				–	124.4M	59.3	4.09	30.0	**31.2**	.403
+ Domain				15		45.8 ±.03	3.61 ±.002	33.4 ±.05	35.3 ±.02	.436 ± .3e-3
+ UD	syn	dep	39	14	+54.1M	35.2 ±.24	3.09 ±.014	39.2 ±.11	42.3 ±.18	.488 ± .8e-3
+ DM	sem	dep	59	15	+54.4M	34.2 ±.32	3.05 ±.019	38.8 ±.15	42.5 ±.26	.490 ±1.0e-3
+ PSD	sem	dep	90	16	+54.9M	34.1 ±.43	2.96 ±.014	39.2 ±.17	44.0 ±.17	.491 ±1.4e-3
+ PTB-phr	syn	const	38	14	+54.1M	33.5 ±.30	2.97 ±.026	40.3 ±.09	43.9 ±.34	.500 ± .6e-3
+ PTB-fxn	syn	const	537	14	+62.7M	32.4 ±.37	2.92 ±.030	41.1 ±.18	44.8 ±.36	.507 ±1.3e-3
+ PTG	sem	const	72	15	+54.6M	29.6 ±.20	**2.68** ±.028	**43.4** ±.08	48.8 ±.32	.524 ± .5e-3
+ EDS	sem	const	10	15	+53.6M	**26.6** ±.09	2.78 ±.024	43.1 ±.10	46.6 ±.24	**.527** ± .8e-3

Table 9: Main results **without early stopping**: performance of language models combined with 7 SLR formalisms of different scope, structure, and label set (each corresponding to a $P_{Ensemble}$ in §4.3), compared to vanilla GPT-2 and a version of GPT-2 that has been domain-finetuned on the raw text of the SLR training corpus (P_{LM}). We report each quality metric as mean ± stdev over 5 random seeds. We also report model size in #parameters and training speed in sentences per second as measures of efficiency. Best results in each column are **bolded**. For confidence, 'best' means best-calibrated, i.e., the smallest relative difference to accuracy.

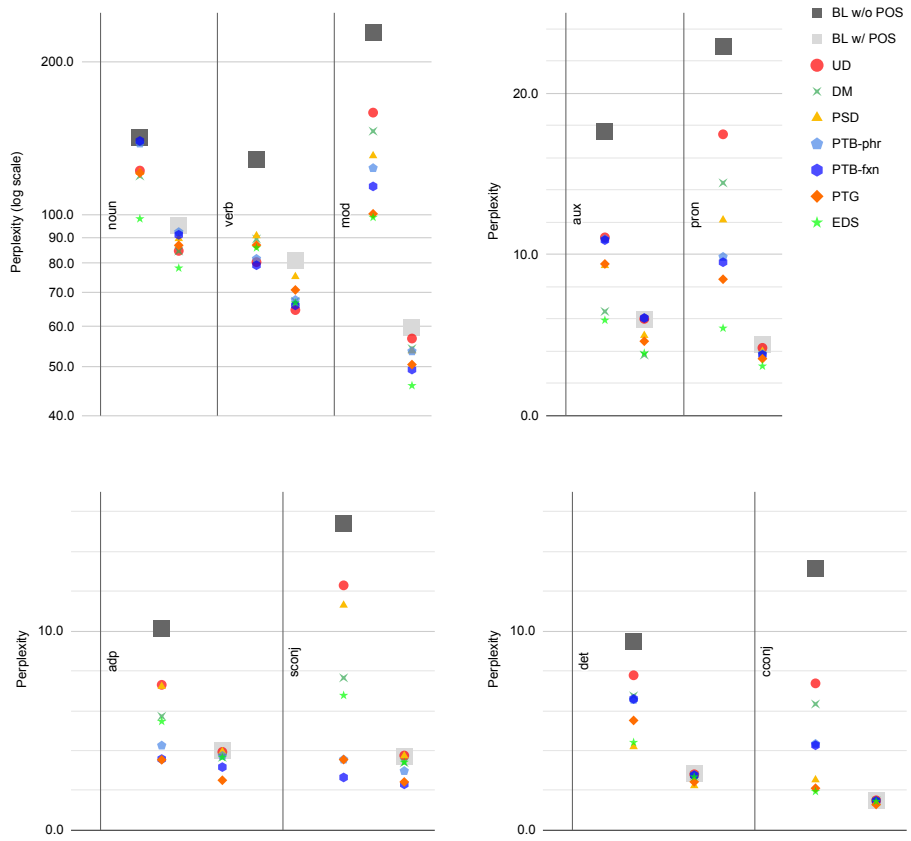

Figure 3: Model perplexity (lower is better) with UPOS as additional input. Top left: nouns, verbs, and modifiers; top right: auxiliaries and pronouns; bottom left: adpositions and subordinating conjunctions; bottom right: determiners and coordinating conjunctions. Big gray squares mark baseline (finetuned GPT-2) performance without (dark) and with (light) POS inputs and SLR-specific data points without/with POS inputs follow below the squares in each respective column. Mind the different y-axis scales, and in particular the log scale in the top-left plot, which makes it easier to read very big and slightly smaller (but still big) differences at the same time.

	Eval	Train	Perplexity						
POS	**Toks**	**Vocab**	**GPT-2**	**UD**	**DM**	**PSD**	**PTB-phr**	**PTB-fxn**	**PTG**	**EDS**
All	22,596	27,344	45.9 ± .1	32.7 ± 0.2	31.4 ± 0.1	30.7 ± 0.1	29.8 ± 0.2	29.0 ± 0.3	26.8 ± 0.3	**24.7** ± 0.3
noun	7,731	18,435	142.5 ± .7	122.0 ± 0.1	119.0 ± 1.0	120.9 ± 0.7	138.0 ± 2.5	139.6 ± 4.5	120.6 ± 2.5	**98.0** ± 3.6
verb	2,639	7,100	128.8 ± .8	**80.4** ± 1.2	89.4 ± 1.1	90.7 ± 0.7	81.7 ± 0.9	**79.3** ± 0.7	86.9 ± 2.5	85.9 ± 1.4
mod	2,235	6,292	228.7 ± .5	158.8 ± 3.8	145.9 ± 3.3	130.5 ± 2.0	123.4 ± 2.1	113.5 ± 3.3	**100.2** ± 4.0	**98.6** ± 4.0
aux	582	95	17.6 ± < .1	11.1 ± 0.1	6.5 ± 0.1	9.3 ± 0.2	11.0 ± 0.2	10.9 ± 0.1	9.4 ± 0.1	**5.9** ± 0.1
adp	1,957	232	10.1 ± < .1	7.3 ± 0.1	5.7 ± 0.1	7.2 ± 0.1	4.3 ± < 0.1	**3.6** ± < 0.1	**3.5** ± 0.1	5.5 ± 0.1
part	645	27	3.7 ± < .1	2.0 ± < 0.1	2.1 ± 0.1	2.3 ± 0.1	**1.7** ± 0.1	1.7 ± < 0.1	1.7 ± < 0.1	**1.6** ± < 0.1
sconj	268	96	15.4 ± .1	12.3 ± 0.1	7.7 ± 0.2	11.3 ± 0.6	3.5 ± < 0.1	**2.6** ± < 0.1	3.5 ± 0.1	6.8 ± 0.1
cconj	548	35	13.0 ± .1	7.4 ± 0.3	6.3 ± 0.5	2.5 ± 0.1	4.3 ± 0.1	4.3 ± 0.1	2.1 ± < 0.1	**1.9** ± < 0.1
det	1,726	91	9.4 ± .1	7.8 ± 0.1	6.7 ± 0.4	**4.2** ± 0.1	6.5 ± 0.4	6.6 ± 0.4	5.5 ± 0.6	**4.4** ± 0.2
pron	868	149	22.9 ± < .1	17.5 ± 0.2	14.4 ± 0.3	12.1 ± < 0.1	9.9 ± 0.5	9.5 ± 0.1	8.5 ± 0.1	**5.4** ± 0.1
num	719	1,059	72.6 ± .3	57.1 ± 0.7	55.2 ± 0.6	53.3 ± 0.5	58.4 ± 0.6	58.6 ± 2.5	**48.3** ± 0.4	**47.5** ± 1.0
punct	2,527	68	4.9 ± < .1	**2.3** ± < 0.1	2.8 ± < 0.1	3.2 ± < 0.1	**2.3** ± < 0.1	**2.4** ± 0.1	2.6 ± 0.1	2.7 ± < 0.1
misc	151	183	7.0 ± < .1	4.6 ± 0.1	5.1 ± 0.5	5.2 ± 0.3	**3.7** ± 0.1	**3.5** ± 0.2	5.6 ± 0.2	**4.0** ± 0.4

Table 10: PPL breakdown by UPOS classes of individual models (3-seed-averages), and the macro-average ± stdev over all SLR-combined models. We show token counts and observed vocabulary sizes for reference. mod—adjectives and adverbs; aux—auxiliary verbs; adp—adpositions; part—particles; sconj—subordinating conjunctions; cconj—coordinating conjunctions; det—determiners; pron—pronouns; num—numbers; punct—punctuation. Best results (within the variance) in each row are **bolded**.

Ablation	**Applied in**	**UD**	**DM**	**PSD**	**PTB-phr**	**PTB-fxn**	**PTG**	**EDS**
Full		32.7	31.4	30.7	29.8	29.0	26.8	24.7
⤮ Labels	testing	+10.2	+4.7	+5.8	+63.0	+73.9	+18.7	+21.8
⤮ Anchors	testing	+110.8	+34.8	+22.4	+177.0	+223.1	+103.4	+70.4
⤮ Both	testing	+92.4	+33.4	+19.2	+168.9	+207.4	+80.1	+70.1
⤮ Labels	training	+2.0	+1.4	+1.6	+8.4	+9.0	+5.3	+1.9
⤮ Anchors	training	+13.0	+8.5	+6.3	+16.9	+17.5	+18.1	+12.8
⤮ Both	training	+13.0	+7.8	+6.1	+17.3	+18.3	+18.5	+12.8
⤮ Labels	both	+2.0	+1.3	+1.7	+8.4	+9.3	+5.4	+2.0
⤮ Anchors	both	+13.1	+7.9	+6.0	+16.7	+17.5	+19.2	+14.0
⤮ Both	both	+13.3	+7.3	+6.0	+16.8	+18.1	+19.4	+14.1
− SLR	both	+13.2	+14.5	+15.2	+16.1	+16.9	+19.1	+21.2

Table 11: Ablations measured in absolute perplexity difference (ΔPPL). *Full* and −SLR correspond, respectively, to table 2's rows 4 (DM) / 6 (PTB-phr) and row 2 (GPT-2 +Domain).