Exploring Graph Representations of Logical Forms for Language Modeling

Michael Sullivan
University at Buffalo
Saarland University
msullivan@lst.uni-saarland.de

Abstract

We make the case for language models over logical forms (LFLMs), arguing that such models are more data-efficient than their textual counterparts. To that end, we introduce the *Graph-based Formal-Logical Distributional Semantics* (GFoLDS) prototype, a pretrained LM over graph representations of logical forms, as a proof-of-concept of LFLMs. Using GFoLDS, we present strong experimental evidence that LFLMs can leverage the built-in, basic linguistic knowledge inherent in such models to immediately begin learning more complex patterns. On downstream tasks, we show that GFoLDS vastly outperforms textual, transformer LMs (BERT) pretrained on the same data, indicating that LFLMs can learn with substantially less data than models over plain text. Furthermore, we show that the performance of this model is likely to scale with additional parameters and pretraining data, suggesting the viability of LFLMs in real-world applications.

1 Introduction

Although recent advances in LLMs have led to remarkable performance on a wide variety of benchmarks, the consistent improvements exhibited by SoTA LLMs are largely due to corresponding increases in model size (Villalobos et al., 2024; Muennighoff et al., 2024). Given the Chinchilla Scaling Laws (Hoffmann et al., 2022) and the rate at which SoTA LLMs are expanding, Villalobos et al. (2024) estimate that high-quality English training data will be exhausted at some point between 2026 and 2032: language model expansion is outpacing available natural language production. This suggests that—without models that use significantly less data than current approaches—LLMs' performance increases will begin to decelerate substantially in the near future.

However, there is a considerable amount of evidence in the literature (e.g. Xu et al., 2021; Wu et al., 2021b; Prange et al., 2022; Sachan et al., 2021; Zhou et al., 2020; Zhang et al., 2020, 2022, etc.) indicating that linguistically-informed LMs—models whose inputs and/or architectures are augmented by linguistic knowledge—can improve performance without consuming more text data.

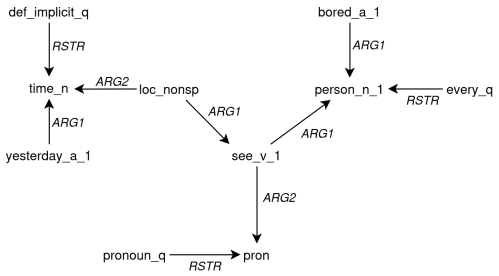

Figure 1: DMRS representation of the sentence "*every bored person saw her yesterday*."

In this paper, we argue for the use of LMs over logical forms (LFLMs): LMs that take as input semantic representations, rather than text. In particular, we posit the following hypothesis:

The Linguistic-Knowledge Catalysis Hypothesis (LKCH): *The (aspects of) linguistic knowledge incorporated into LFLMs greatly accelerates their learning of elementary linguistic phenomena, in turn accelerating the learning of more complex patterns.*

One relevant corollary of the LKCH is that LFLMs can learn with less data: the linguistic knowledge built into LFLMs facilitates more rapid learning of advanced phenomena.

We argue that the primary advantage of logical forms (as opposed to other types of linguistic knowledge) with respect to language modeling is the de-noising effect conferred by the function-argument structure inherent to such representations. Specifically, the translation of surface text to logical form has an equivalence-classing effect, so that all syntactic paraphrases of the same proposition—for

Findings of the Association for Computational Linguistics: ACL 2025, pages 12285–12307
July 27 - August 1, 2025

example, an active sentence and its passive counterpart—are mapped to the same representation. The benefit to an LM of the de-noising effect resulting from this equivalence-classing is clear: the model does not need to learn to equate periphrastic structures, and so can immediately begin learning co-occurrence relations between predicates.

To make the case empirically for LFLMs, we introduce the *Graph-based Formal-Logical Distributional Semantics* (GFoLDS; Section 3) model as a proof-of-concept of such LMs. GFoLDS is a pretrained, encoder graph transformer (Wu et al., 2021a) over structures derived from Dependency Minimal Recursion Semantics (DMRS; Copestake, 2009) representations (see Figure 1): directed, labeled acyclic graphs in which nodes correspond to predicates and edge labels denote relations (i.e. semantic roles) between them, in a similar manner to Abstract Meaning Representation (AMR; Banarescu et al., 2013) structures.

Unlike AMR, which abstracts away from morphosyntactic features such as tense and number, DMRS includes these features (see Figure 7 in the Appendix). Beyond yielding a more faithful representation of linguistic meaning, this additionally has the effect of further de-noising the model's input by offloading the morphological realization of these features to explicitly annotated labels: GFoLDS does not need to learn the surface patterns corresponding to inflection, as this information is instead explicitly provided through the DMRS representation. For example, the model does not need to understand that the suffix *–s* has the same effect on meaning as other, irregular realizations of pluralization (e.g. *goose* ⇒ *geese*), because plural nouns are directly labeled as such.

The contributions of this work are threefold: firstly, we provide experimental support towards the validity of the LKCH, demonstrating that—from the start of pretraining—GFoLDS achieves near-peak performance on tasks designed to evaluate its elementary linguistic knowledge, and that this translates to more rapid learning of complex phenomena (Section 4).

Secondly, we demonstrate the viability of pretrained LFLMs in Section 5, by comparing the performance of GFoLDS to that of BERT (trained on ∼6.5 times more data than GFoLDS; Devlin et al., 2019) on a range of downstream tasks. Although the actual BERT models outperform GFoLDS, our model outperforms—by a wide margin—BERT models pretrained on the same data as GFoLDS on all benchmarks, indicating that LFLMs can learn useful representations with much less data than their textual counterparts.

Thirdly, we establish that LFLMs have the potential to compete with textual LLMs at scale: in Section 6, we present evidence indicating that GFoLDS is likely to scale with respect to parameter count and pretraining dataset size.

We make all code for the GFoLDS model and the experiments conducted in this paper available on GitHub[1].

2 Related Work

As discussed in Section 1, there exists a body of research indicating that injecting graph representations of linguistic structures into textual LMs can improve downstream performance. Xu et al. (2021) fuse dependency parse graphs into pretrainined transformer encoders (e.g. BERT and RoBERTa; Liu et al., 2019), and surpass the then-SoTA results on relation classification, entity typing, and question answering tasks, demonstrating the general utility of linguistically-informed LMs. Similarly, Wu et al. (2021b) inject syntactic dependency parse graphs into a pretrained BERT model, yielding then-SoTA results on semantic role labeling and relation extraction tasks. These authors additionally compare the respective impacts of syntactic and semantic representations on performance, and find that semantic representations are more beneficial with respect to downstream performance than syntactic structures.

Prange et al. (2022) show that linguistic structures can be incorporated into the input of GPT-2 (Radford et al., 2018) to improve next word prediction accuracy and entropy. Of particular relevance is their finding that Elementary Dependency Structures (EDS; Oepen and Lønning, 2006)—a semantic framework that is related to DMRS—yield greater performance improvements than syntactic (or other semantic) representations.

All of the models described thus far in this section are hybrid architectures that merge textual and graph representations. Furthermore, in these approaches, the textual component of the model is initialized from a pretrained LM such as BERT, RoBERTa, or GPT-2. This contrasts with the GFoLDS model, which takes only graph representations as input and is pretrained from scratch.

[1] https://github.com/mjs227/GFoLDS

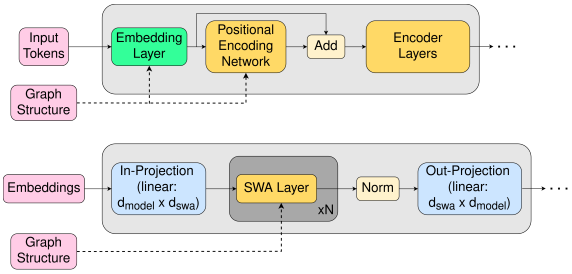

Figure 2: Top-level architecture of the GFoLDS model (top) and positional encoding network (bottom).

To the best of our knowledge, Functional Distributional Semantics at Scale (FDSAS; Lo et al., 2023) represents the only[2] extant LFLM aside from GFoLDS. FDSAS is a variational autoencoder over DMRS graphs that learns probabilistic truth-conditional functions for each predicate (node label). FDSAS is not, however, a transformer and so does not benefit from the flexibility afforded by such architectures: it is unclear how to scale the model to yield a deeper architecture, and it does not generate usable hidden states that can be passed to (for example) a classification head for downstream tasks. Additionally, its rigorous formal-semantic foundations give rise to a degree of inflexibility in this model: for example, Lo et al. (2023) discarded prepositions, quantifiers, and modal verbs from FDSAS' training dataset out of necessity.

3 GFoLDS

In this section, we describe the GFoLDS model architecture (Section 3.1), preprocessing steps that we employed on its DMRS graph inputs (Section 3.2), and the model's pretraining procedure (Section 3.3).

3.1 Architecture

The GFoLDS model is a variant of the graph transformer paradigm (Wu et al., 2021a), which was originally introduced for molecule graph classification. A graph transformer consists of a graph neural network (GNN) that encodes local neighborhood information, whose output is then fed to a permutation-invariant (i.e. without linear positional embeddings) transformer encoder for global message-passing (attention).

Unique to this work is the GNN component of GFoLDS, which consists of an embedding layer and the *positional encoding network* (see Figure 2). The output of the embedding layer is fed into the positional encoding network, which provides each node with a representation of its local neighborhood in the DMRS graph structure.

[2]Aside from its predecessor: Functional Distributional Semantics (Emerson, 2018).

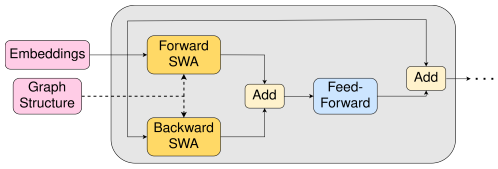

Figure 3: Architecture of an SWA layer in the positional encoding network.

Let n_i denote the ith node in a graph G, and let $F(n_i)$ be its set of DMRS features (person, number, tense, etc.; see Figure 7 in the Appendix). Then the output of the embedding layer $\vec{e_i} = E(X, G)_i$ (where X denotes the node labels of G) is the sum of the embedding of the node's label $\mathcal{E}_T(X_i)$ with the normalized sum of the embeddings $\mathcal{E}_F(\phi)$ of each feature $\phi \in F(n_i)$ (Equation 1).

$$\vec{e_i} = \mathcal{E}_T(X_i) + Norm\left(\sum_{\phi \in F(n_i)} \mathcal{E}_F(\phi)\right) \quad (1)$$

These summed feature and node/predicate embeddings are then passed to the positional encoding network (see Figure 2). This module consists of a linear layer (to project the embeddings from d_{model} to d_{SWA}) followed by a stack of *step-wise aggregation* (SWA) layers (see Figure 3)—in which the output of each SWA layer is fed to the subsequent layer—followed by a second linear projection (to project from d_{SWA} back to d_{model}).

In their respective adaptations of the GraphSAGE (Hamilton et al., 2017) and Graph Convolutional Network (GCN; Kipf and Welling, 2017) architectures to directed graphs, Xu et al. (2018) and Tong et al. (2020) introduce *forward* and *backward* node projection layers, which encode information about incoming and outgoing connections (respectively) for a given node. In a similar fashion, each SWA layer (see Figure 3) contains a forward (Equation 2) and a backward SWA block, which encode the nodes—and the semantic roles thereof—mapping into and out of a given node.

$$\vec{f_i} = Norm\left(\sum_{n_k \xrightarrow{\ell} n_i \in G} W_\ell^{(f)} \vec{h_k}\right) \quad (2)$$

For each node $n_i \in G$, its forward representation

$\vec{f_i}$ (i.e. the output of the forward SWA block) is the (normalized) sum of $W_\ell^{(f)}\vec{h_k}$ for each node n_k with an edge $n_k \xrightarrow{\ell} n_i$ in the graph structure, where $W_\ell^{(f)} \in \mathbb{R}^{d_{SWA} \times d_{SWA}}$ is the forward *edge projection* linear layer for the edge label ℓ. This representation of each edge label as a unique projection layer is conceptually similar to the label-specific matrices employed in Relational GCNs (Schlichtkrull et al., 2018) and Beck et al.'s (2018) adaptation of the Gated GNN (Li et al., 2016) architecture to labeled graphs.

The backward SWA block is architecturally identical to the forward block, but contains distinct edge projection matrices $W_\ell^{(b)}$ and operates on the transpose G^T of the graph. The respective forward and backward representations of each node n_i are summed together and passed through a two layer, feed-forward block (identical to the feed-forward blocks in the model's encoder layers).

The input to the encoder stack is then the sum of the outputs of the embedding layer and positional encoding network: $E(X, G) + P(E(X, G), G)$. This is analogous to (and inspired by) the approach taken by most transformer LMs (e.g. Devlin et al., 2019; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Dubey et al., 2024, etc.), where the encoder stack input for token t at position p is the sum of the token embedding for t and the positional embedding for p.

The encoder is *not* directly exposed to the graph structure: all nodes are able to attend to any other node(s). The encoder layers in the GFoLDS architecture are similar to those in BERT and Vaswani et al. (2017), with a few key differences with respect to the residual connections and layer normalization. A full description of the GFoLDS architecture is located in Appendix A.

3.2 Data Preprocessing

GFoLDS' pretraining corpus consisted of ∼17.5 million randomly-selected sentences from the November 1, 2023 English Wikipedia dump[3], constituting a total of ∼508 million words (∼6.5 times smaller than BERT's pretraining corpus). We first used Spacy's SentenceRecognizer[4] pipeline to extract individual sentences from the text. We then used the PyDelphin (Goodman, 2019) library with the ACE/ERG (Copestake and Flickinger, 2000) rule-based parser/grammar[5] to obtain a DMRS representation of each sentence, before preprocessing the resulting DMRS structures to yield GFoLDS input graphs. The ACE/ERG parser was able to parse ∼84% of the data, for a total of ∼14.6 million DMRS-derived graphs.

It was not feasible to tokenize named entities (CARGs) and out-of-vocabulary (OOV) items in the same manner as for the in-vocabulary DMRS predicates—i.e. by simply assigning an integer to each unique predicate string in the vocabulary. We therefore replaced all CARGs and OOV terms with the [MASK] token. These [MASK] tokens are *not* targets for prediction during the pretraining procedure (as they are OOV, so there is no possible target token): the goal is instead to have the model represent the OOV item with the closest in-vocabulary token, based on the context in which the OOV item appears. The removal of CARGs and OOV items is a stop-gap measure, and remains an open problem and barrier to the performance of the GFoLDS model. We defer the incorporation of CARGs and OOV items into the model's input structures to future work (see Section 8.1).

Further details on the preprocessing procedures that we employed are located in Appendix B.

3.3 Pretraining

We pretrained GFoLDS with the *masked-node modeling* (MNM) objective, which is analogous to the MLM objective used to pretrain encoder transformer LMs. The model trained for four epochs, with a total training time of ∼102 hours (∼25.5 hours per epoch) on a single NVIDIA A100 GPU. Further details on our pretraining procedure and hyperparameters are located in Appendix C.1.

We employed a GFoLDS model with two SWA layers and ten encoder layers (eight attention heads each), and set $d_{SWA} = d_{model} = 1024$. The MNM prediction head that we used is identical to BERT's MLM prediction head (aside from the difference in vocabulary size). This yields a total of ∼174 million parameters: for comparison, $\text{BERT}_{\text{base}}$ (12 encoder layers, d_{model} = 768) and $\text{BERT}_{\text{large}}$ (24 encoder layers, d_{model} = 1024) have ∼110 million and ∼335 million parameters, respectively.

4 Evaluating the LKCH

This section is dedicated to an investigation of the validity of the LKCH. The hypothesis can be bro-

[3] https://huggingface.co/datasets/wikimedia/wikipedia; CC-BY-SA-3.0 license.

[4] https://spacy.io/api/sentencerecognizer

[5] ERG-1214 release: https://github.com/delph-in/erg

ken down into two distinct claims: (i) that the (aspects of) linguistic knowledge incorporated into LFLMs greatly accelerates their learning of elementary linguistic phenomena; and (ii) that this in turn accelerates the learning of more complex patterns.

We therefore divided this experiment into two parts, which respectively probe the model's knowledge of elementary and complex linguistic phenomena. Due to the claim made in the LKCH that linguistically-informed LMs' learning of complex patterns is *accelerated*, we evaluate the model at regular intervals throughout pretraining, in order to measure the rate at which it is learning.

4.1 Comparison Models

As a baseline, we pretrained $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ (uncased) models from scratch on the same dataset as GFoLDS (the surface sentences), for the same number of epochs (four).

Recall that the ACE/ERG parser was only able to parse ∼84% of the sentences in GFoLDS' pretraining dataset (see Section 3.2). If GFoLDS were able to match the performance of BERT with ∼84% of the pretraining data (for example), then—given the ACE/ERG parser's ∼84% successful parse rate—this model would provide no practical benefit over its textual counterparts. We therefore chose to pretrain the BERT comparison models (BERT-C) on GFoLDS' entire pretraining dataset, rather than the parsable subset: the BERT-C models were pretrained with ∼1.19 times more data than GFoLDS.

We evaluated a variety of different pretraining hyperparameter configurations for BERT-C in order to yield the most rigorous comparison: a full description of these hyperparameter configurations is located in Appendix C.2.

4.2 Experimental Setup

This experiment consisted of two tasks that were designed to evaluate knowledge of elementary linguistic phenomena, and one to evaluate knowledge of more complex patterns: we evaluated GFoLDS and the BERT comparison models on the three tasks at twenty evenly-spaced intervals per pretraining epoch, for a total of eighty points of comparison.

Note that it is difficult—if not impossible—to define "elementary" and "complex" linguistic phenomena in absolute terms. In this paper, we consider these terms in a relative sense: the elementary tasks (Section 4.2.2) are undoubtedly *more elementary* than the complex task (Section 4.2.1), in that they do not require as much (if any) world knowledge, and deal entirely with awareness of basic linguistic categories. Conversely, this entails that our complex task is complex *relative to* our elementary tasks.

Assuming that the LKCH holds, we should expect to see GFoLDS outperform the BERT-C models on the complex task throughout the pretraining process, as—according to the hypothesis—GFoLDS is able to learn complex patterns faster than textual LMs.

On the elementary tasks, we again expect GFoLDS to outperform BERT-C, but also that GFoLDS' performance will improve substantially faster than it does on the complex tasks: the LKCH predicts that an LFLM's accelerated learning of elementary phenomena catalyzes its learning of complex patterns, so its learning of the former should therefore accelerate at a faster rate than that of the latter.

4.2.1 Complex Task

The complex task in this experiment is the RELPRON (Rimell et al., 2016) dataset. This dataset consists of *terms* (nouns), each paired with a hypernym and up to ten *properties*: relative clauses that restrict that hypernym (see Table 2 in the Appendix). The development set consists of 65 terms and 518 properties (∼8 properties per term on average), and the test set contains 73 terms and 569 properties (∼7.8 per term). The task is to retrieve the properties that apply to each term, without including those that do not: the evaluation metric is Mean Average Precision (MAP) score.

To evaluate the models, we constructed templates out of each (term, hypernym, property) triple: for example, the triple (*telescope*, *device*, *astronomers use*) yields the template "*a device that astronomers use is a telescope*". We then replaced the target term with the [MASK] token (e.g. "*a device that astronomers use is a* [MASK]"): the probability assigned to a given term under the masked distribution is taken as proportional to the probability that the property applies to that term.

As discussed in Section 3.2, CARGs and OOV items are masked in the input graphs: given that each template only contains four content words (the hypernym, verb, relative clause subject/object, and the target term), the GFoLDS model is effectively blind to (at least) one third of the context in templates that contain OOV items or CARGs. We therefore discarded all examples containing

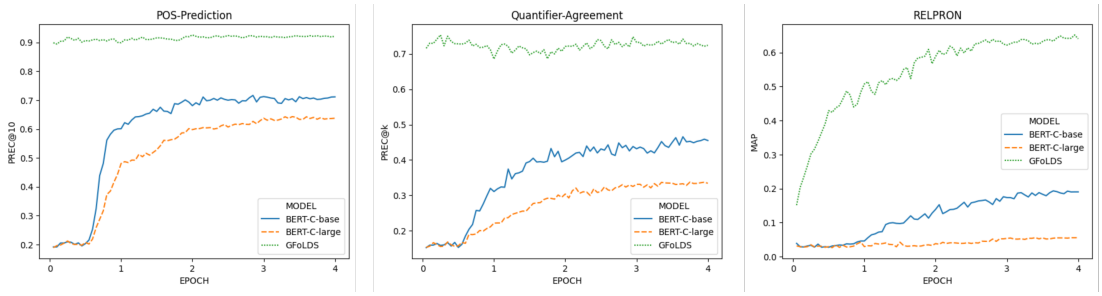

Figure 4: Precision scores on the two elementary tasks (left and center) and MAP scores on the RELPRON test set (right) across the 80 evenly-spaced training snapshots for GFoLDS (green) and the $\text{BERT}_{\text{base}}$/$\text{BERT}_{\text{large}}$ comparison models (blue/orange, respectively).

CARG-bearing predicates or OOV items for evaluation. This resulting subset of the test split (the "RELPRON-No-UNK/NE" column of Table 1) contains 63 terms and 421 properties, for a total of ∼6.68 properties per term on average.

Due to the small size of the dataset (and the lack of a training split), the frozen, pretrained models were used to obtain token probabilities for property ranking on this task.

4.2.2 Elementary Tasks

POS-Prediction. The first elementary task evaluates the LMs' ability to model the distribution of parts-of-speech: this is a commonly-employed probing task used to assess LMs' elementary linguistic competence (Waldis et al., 2024). We evaluated the models on 200 sentences (not drawn from their pretraining data) from English Wikipedia. For each sentence, we masked a single word and recorded its POS: we then extracted the models' probability distribution over the masked word of each sentence, and recorded their precision at ten with respect to the masked word's part-of-speech. For example, if the masked word was a noun, and nine of the model's top ten most-likely predictions were nouns, then the model received a score of 0.9 for that instance.

Quantifier-Agreement. The second elementary task evaluates the models' knowledge of quantifier number agreement—whether a quantifier restricts a singular or plural noun (or both)—across 179 sentences drawn from English Wikipedia (not used to pretrain the models). Although less common than POS prediction, quantifier agreement tasks have also been employed as a metric of language models' elementary lingustic competence (see e.g. Huebner et al., 2021; Waldis et al., 2024).

As in the POS-prediction task above, we masked a single quantifier per sentence, then extracted the probability distribution over the masked quantifier of each sentence to compute precision at k. The value of k varied as a function of the number of the noun in the masked quantifier's restriction, as there were more quantifiers that restrict singular nouns ($k = 14$) than plural nouns ($k = 13$).

Additional details on the setup and evaluation of the elementary tasks are given in Appendix F.

4.3 Results

The results of this experiment (Figure 4) conform almost exactly to the behavior predicted by the LKCH: the performance of GFoLDS on the elementary tasks remains relatively constant throughout pretraining, because the model starts near peak performance from the first checkpoint—this indicates that its learning of elementary patterns was complete within 5% of the first epoch. While the performance of the BERT models steadily increases on these tasks, it does so at a much lower rate.

On the RELPRON test set, GFoLDS begins improving immediately, $\text{BERT-C}_{\text{large}}$ does not improve substantially, and the performance of $\text{BERT-C}_{\text{base}}$ doesn't begin to meaningfully increase until the latter half of the first epoch (and at a lower rate than that of GFoLDS): the point at which it began to improve on the elementary tasks.

These results constitute strong evidence towards the LKCH. It is clear from this experiment that GFoLDS can model the elementary phenomena almost from the onset (and retains this ability throughout pretraining) and GFoLDS' performance on the RELPRON test set suggests that this accelerated learning of elementary phenomena translates to more rapid learning of more complex patterns.

5 Downstream Tasks

In order to demonstrate the viability of LFLMs, we evaluated GFoLDS on a series of downstream

tasks: our primary objective was to demonstrate that language models over logical forms are able to learn from less data than their textual counterparts.

To that end, we compared GFoLDS to the BERT comparison (BERT-C) models pretrained on the same data (see Section 4.1), in order to show that GFoLDS is able to outperform a textual LM when pretrained on the *same* amount of data: it then follows that GFoLDS would be able to perform as well as a textual LM with *less* pretraining data.

We additionally compared GFoLDS to the original $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ (uncased) models, both of which are pretrained on roughly 6.5 times more data than GFoLDS.

5.1 Datasets

We evaluated the models on four downstream tasks: the RELPRON task described in Section 4.2.1, SNLI (Bowman et al., 2015), the MegaVeridicality V2.1 factuality dataset (White et al., 2018), and the McRae et al. (2005) property inference dataset.

5.1.1 SNLI

All five models were fine-tuned end-to-end on SNLI with a two-layer, feed-forward MLP classifier over their mean-pooled token/node embeddings. The details of our SNLI fine-tuning hyperparameters are located in Appendix D.

Graph Representations. The GFoLDS model described in Section 3 cannot process multiple sentences. While this obviously represents a serious general limitation of the model (discussed further in Section 8), it also presents a more immediate complication in terms of the SNLI dataset, whose examples are given as (premise, hypothesis) pairs.

To overcome this obstacle, we construct a single graph $G(P_i, H_i)$ from each premise ($G(P_i)$) and hypothesis ($G(H_i)$) pair. $G(P_i, H_i)$ is derived from the disjoint union $G(P_i) \oplus G(H_i)$ by adding the node *if_x_then*, and then inserting edges[6] *if_x_then* $\xrightarrow{ARG1}$ $htop(G(H_i))$ and *if_x_then* $\xrightarrow{ARG2}$ $htop(G(P_i))$, as in Figure 5.

5.1.2 MegaVeridicality V2.1

The objective in this task is to determine whether the event denoted by a subordinate clause is true, given the context of the matrix clause (see Table 3 in the Appendix). The MegaVeridicality V2.1 dataset consists of 5,026 examples, each with ten annotations of the factuality of the event: for this experiment, we converted the data into a binary classification task via majority voting of the annotated labels (see Appendix E).

As with RELPRON (see Section 4.2.1), we removed all examples containing OOV items and/or CARG-bearing predicates for this task, due to the lower amount of content words per example in MegaVeridicality V2.1 in comparison to SNLI. This left 3,126 remaining examples (∼62%), of which we withheld 20% as a test set.

We fine-tuned all five models end-to-end for eight epochs with a two-layer (binary) classifier head over their mean-pooled embeddings, using a learn rate of 10^{-6}, a weight decay value of 10^{-5}, and a batch size of 16.

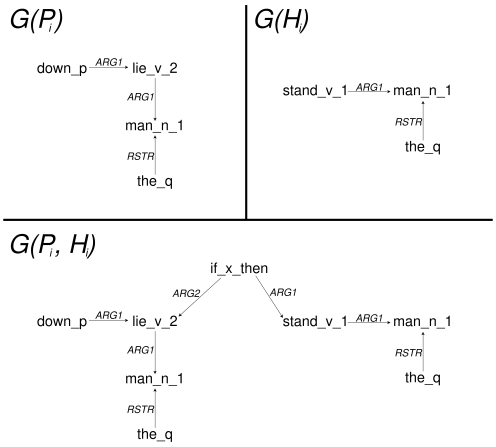

Figure 5: Illustration of the derivation of $G(P_i, H_i)$ (bottom) from $G(P_i)$ (top left) and $G(H_i)$ (top right) for the contradiction example (*the man is lying down, the man is standing*).

5.1.3 The McRae et al. (2005) Dataset

To evaluate the quality of GFoLDS' embeddings in comparison to BERT, we evaluated the models on a property inference task using the McRae et al. (2005) feature norm database. This database consists of a set of 541 *concepts* (words) W and 2,526 *features* F, where each concept $w \in W$ is assigned a feature vector $f(w) \in \mathbb{R}^{|F|}$: the value of $f(w)_Q$ is the value of the feature Q for the word w (see Table 4 in the Appendix).

Following Rosenfeld and Erk (2022), we created ten random folds consisting of 50 concepts each from the dataset. The concepts within each fold represent the set U of *unknown* words, and those outside of the fold represent the set $K = W - U$ of *known* words. For each $u \in U$, its feature vector

[6]The DMRS *htop* node is typically the (node corresponding to the) main verb of the sentence from which G is derived: see Copestake et al. (2005).

	RELPRON-All (MAP)	RELPRON-No-UNK/NE (MAP)	SNLI (Accuracy)	MegaVeridicality V2.1 (Accuracy)	McRae et al. (2005) (Spearman ρ)
GFoLDS	—	0.651	81.0%	81.3%	0.205
BERT-C_{base}	0.147	0.193	79.9%	78.1%	0.167
BERT-C_{large}	0.047	0.056	62.0%	76.2%	0.134
$BERT_{base}$	0.667	0.690	90.7%	84.2%	0.247
$BERT_{large}$	0.768	0.769	91.1%	85.6%	0.241
FDSAS	0.580	—	—	—	—

Table 1: Results for GFoLDS and the four BERT models on downstream tasks. BERT-$C_{base/large}$ indicates the BERT models pretrained on the same data as GFoLDS (see Section 4.1).

$f(u)$ is withheld: the task is to reconstruct $f(u)$.

As the goal of this experiment was to evaluate the quality of their embeddings, the models were not fine-tuned for this task: again following Rosenfeld and Erk (2022), we used the Modified Adsorption (Talukdar and Crammer, 2009) graph label-propagation algorithm to estimate properties, using the cosine distance between two words' embeddings to weight the graph edge between them.

The evaluation metric was Spearman's ρ: we averaged over all scores for each unknown word in each fold to yield each model's final score.

5.2 Results

The results of these experiments are given in Table 1. Although the original BERT models outperform GFoLDS, the BERT comparison models trained on the same data as our model (BERT-C; see Section 4.1) both lag far behind GFoLDS on all four benchmarks, demonstrating across a wide range of downstream tasks that our model is able to learn useful representations with less data than its textual counterparts.

The pretrained FDSAS model is not publicly available, so we were unable to evaluate it on the RELPRON No-UNK/NE subset to obtain a direct comparison to GFoLDS. However, based on the differences in MAP scores from the full dataset to the No-UNK/NE subset for $BERT_{base}$/BERT-C_{base} and $BERT_{large}$/BERT-C_{large} (+0.023/+0.019 and +0.001/+0.009, respectively), it is likely that GFoLDS would outperform FDSAS (0.580) on that data.

6 Scalability

While the results of Section 5 show that GFoLDS outperforms textual models trained on similar amounts of data, this model is still outperformed by the original BERT models. It is therefore crucial to establish the scalability of GFoLDS: the degree to which we would expect its downstream performance to scale if it were larger and/or pretrained on more data. To that end, we applied the techniques of Muennighoff et al. (2024) to GFoLDS, to determine the degree to which our model is under- or over-parameterized—and therefore, by the scaling laws established in Muennighoff et al. (2024), over- or under-trained.

As loss is not an exact predictor of downstream performance (Shin et al., 2022; Tay et al., 2022; Xia et al., 2023), we follow Hoffmann et al. (2022) and evaluate the impact of pretraining token count on a validation task as well. We again used the RELPRON dataset for this purpose, as SNLI introduces potential confounding factors in the form of the fine-tuning procedure.

6.1 Experimental Setup

We pretrained five GFoLDS models on 50%, 25%, 12.5%, 6.25%, and 3.125% of the pretraining data used in Section 3.3. Following Muennighoff et al. (2024), we ensured that the iterations pretrained on less data always use a randomly-selected subset of the dataset used in those with more data.

Aside from the differing number of pretraining tokens, all models were pretrained using the same procedure and hyperparameters as described in Section 3.3, as the focus of this experiment was the effect of pretraining tokens with a fixed parameter count.

6.2 Results

Figure 6 shows that final pretraining loss consistently decreases as the number of pretraining tokens increases from 3.125% to 50% of the data. After this point, the final loss value plateaus: the 50% run (1.3232) finishes with a slightly *lower* cross-entropy loss than the actual (100%) run (1.3331), although this minor difference is likely explained by the noise introduced by random initialization of the models' parameters.

In Appendix G, we prove that—assuming that an

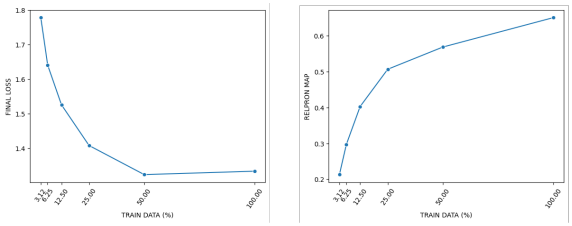

Figure 6: Final pretraining loss (left) and RELPRON MAP (right; No-UNK/NE test split). The 100% run denotes GFoLDS pretrained on the entire dataset.

analogue of the Muennighoff et al. (2024) scaling laws holds for the GFoLDS architecture—it can only be the case that the final loss for the 100% run is (roughly) equal to that for the 50% run if GFoLDS is underparameterized for both the 100% run *and* the 50% run. This is to say that GFoLDS requires much less pretraining data per parameter than textual models: the Chinchilla Scaling Laws (Hoffmann et al., 2022) predict that a textual LLM with the same parameter count ($\sim$174 million) as GFoLDS necessitates $\sim$5 billion pretraining tokens—roughly twenty times more than the $\sim$254 million tokens for which GFoLDS is overparameterized.

We also observe improvement of +0.082 in RELPRON MAP score from the 50% (0.569) to the 100% (0.651) run—higher than the +0.062 increase from the 25% (0.507) to the 50% run.

The results of this experiment lead to the conclusion that GFoLDS is likely scalable in terms of model size and pretraining data. The application of the Muennighoff et al. (2024) scaling laws to GFoLDS indicates that increasing the parameter count will decrease final pretraining loss, while its run-over-run performance on the RELPRON test set suggests that larger pretraining datasets will lead to corresponding increases in the model's downstream performance.

7 Conclusion

In Section 4, we provided direct evidence in support of the LKCH: critically, this finding indicates that *LFLMs can learn with less data than textual models*. This is supported by our findings in Section 6, which indicate GFoLDS is *under*parameterized with half of its pretraining data.

To the best of our knowledge, the experiments in Section 5 represent the first time that a language model pretrained solely over logical forms has been evaluated on a wide range of downstream tasks. GFoLDS' capacity to be applied to tasks ranging from RELPRON to SNLI—and consistently outperform BERT models pretrained on the same amount of data—demonstrates this model's versatility. These results therefore represent a significant step towards demonstrating the practical viability of LFLMs.

Although much work remains to be done before LFLMs can reach the same level of performance and utility as their textual counterparts (see Section 8), the results of Sections 4-6 suggest that such models present an exciting avenue for continuing the improvement of language models at a more sustainable rate of data consumption than LLMs over text.

8 Limitations

This section is divided into two parts: Section 8.1 overviews the limitations of the current GFoLDS pipeline laid out in Section 3, while Section 8.2 discusses the limitations of the experiments conducted in Sections 4-6.

8.1 Limitations of the Current Approach

Tokenization. As discussed in Section 3.2, we replaced all OOV node labels with the [MASK] token. Although this preprocessing step was necessary due to the tokenization scheme that we employed, it prevents the model from seeing every word in every input sentence. For the same reason, we removed CARGs (named entities), keeping only the CARG-bearing predicates (nodes): this amounts to, for example, replacing the sentence "*John went to the park in the spring of 2017*" with "[NAMED] *went to the park in the* [SEASON] *of* [YEAR]."

The removal of these items from its DMRS-derived input graphs almost certainly negatively impacts GFoLDS' performance on downstream tasks, and entirely precluded a direct comparison with FDSAS on the RELPRON benchmark (see Section 5.2). However, this limitation lies purely with the current GFoLDS architecture—rather than the proposed approach in and of itself—as we simply tokenized the nodes by assigning a unique integer to each in-vocabulary term (i.e. node label). The ACE/ERG parser can still parse OOV items: the node label of an OOV term is constructed by appending the suffix "*_unknown*" (along with a part-of-speech tag) to the corresponding string in the surface text.

Future work in this area must involve redesign-

ing the model's approach to node embeddings in order to overcome this limitation, for example by replacing GFoLDS' embedding layer with a small, character-level encoder transformer that is applied to each individual node. As we intended this work to primarily be a proof-of-concept of LFLMs, we left this issue to future research in order to retain focus on that goal.

Multiple-Sentence Inputs. For the SNLI dataset, we were able to overcome GFoLDS' inability to process sequences of multiple sentences by converting each premise/hypothesis pair into a single, connected graph. However, this task-specific remedy cannot be readily generalized to other multiple-sentence NLP benchmarks. In future work, we intend to address this deficiency through sentence-level positional encodings (e.g. rotary embeddings; Su et al., 2024): a sequence of sentences represented as the disjoint union of their DMRS graphs, where the positional encoding $\vec{p_i}$ is added (or concatenated) to the embedding of each node in the ith graph in the sequence.

DMRS Parsing. Although the ACE/ERG parser that we employed to derive DMRS graphs from natural language enjoys high precision (93.77%; Zamaraeva and Gómez-Rodríguez, 2024), it suffers from relatively low recall: only ∼84% of our pretraining dataset was able to be parsed. This can (and should) be addressed in future work, perhaps through the use of a neural DMRS parser (e.g. Buys and Blunsom, 2017), which trades lower precision for higher recall than a rule-based parser such as ACE/ERG.

Additionally, the grammar of the ACE/ERG parser (ERG: English Resource Grammar) is solely an *English* grammar, rendering our approach inapplicable to other languages[7].

Model Type. The GFoLDS model described in this work is an *encoder* model: in order to be able to perform the same range of tasks as current SoTA LLMs, LFLMs must have generative capabilities. In future research, we intend to adapt GFoLDS to construct a sentence-level, graph-to-graph generative LFLM, by adapting existing work in the domain of molecule graphs on autoregressive graph generation (e.g. Bacciu et al., 2020; Goyal et al., 2020; Bacciu and Podda, 2021) that we believe can be readily extended to DMRS, using (for example) a breadth-first search to impose a canonical generation order on the graph nodes during training.

[7]Although broad-coverage grammars do exist for some other higher-resource languages (e.g. the Spanish Resource Grammar; Zamaraeva et al., 2024)

8.2 Experimental Limitations

Logical Representations. While the GFoLDS model demonstrates that language models over DMRS representations are able to learn with less data than textual models, we did not evaluate the LKCH using other logical-form representations such as AMR. AMR in particular is a problematic representational format for the GFoLDS model, as this framework utilizes ∼100 edge labels (Banarescu et al., 2013) (our DMRS-derived graphs have nine; see Table 5 in the Appendix). Recall from the discussion in Section 3.1 that each unique edge label corresponds to unique forward/backward edge projection layers within each SWA layer (see also Appendix B): with the architectural configuration of the GFoLDS model used in this work (i.e. two SWA layers with $d_{SWA} = 1024$), the additional AMR edge labels would add ∼380 million parameters to the positional encoding module (the model in this work has only 174 million *total* parameters).

Although they may require significant architectural modifications, future work should investigate the use of AMR, EDS, and other graph-based semantic representations for language modeling, in order to determine the most effective framework for this application.

Dataset Size. Due to computational limitations, we did not investigate the GFoLDS model at scale (i.e. with the same amount of training data as BERT): it required roughly one month to parse the training dataset used in this work into DMRS[8]. Although this does introduce the risk that our proposed method may only show superiority on a small-scale dataset, the results of Section 6 indicate that this risk is relatively low: extrapolating from the pattern shown in Figure 6 (see the discussion in Section 6.2), we would expect GFoLDS to surpass $\text{BERT}_{\text{large}}$ (the original model) on the RELPRON test set with four times the current training data (still less than that of $\text{BERT}_{\text{large}}$, which addition-

[8]We do not view this as a severe limitation of the proposed approach: DMRS parsing can be achieved in parallel (each parse is independent of the others), and so the speed of parsing scales linearly with the number of CPU cores. With sixteen times as many cores as used in this work, for example, a dataset the size of BERT's pretraining data could be parsed in roughly two weeks.

ally has twice the parameter count).

Model Size. We did not investigate the effect of parameter size on the GFoLDS model: these experiments were only performed with a 174 million parameter LFLM.

Acknowledgments

Computational resources for these experiments were provided by the Center for Computational Research at the University at Buffalo.

References

Davide Bacciu, Alessio Micheli, and Marco Podda. 2020. Edge-based sequential graph generation with recurrent neural networks. *Neurocomputing*, 416:177–189.

Davide Bacciu and Marco Podda. 2021. Graphgen-redux: A fast and lightweight recurrent model for labeled graph generation. In *2021 International Joint Conference on Neural Networks*, pages 1–8.

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 *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada. Association for Computational Linguistics.

Rich Caruana, Steve Lawrence, and C Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in Neural Information Processing Systems*, 13.

Ann Copestake. 2009. **Invited Talk:** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.

Ann A Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage english grammar using hpsg. In *LREC*, pages 591–600.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Guy Emerson. 2018. *Functional Distributional Semantics: Learning Linguistically Informed Representations from a Precisely Annotated Corpus*. Ph.D. thesis, University of Cambridge.

Varun Godbole, George E. Dahl, Justin Gilmer, Christopher J. Shallue, and Zachary Nado. 2023. Deep learning tuning playbook. Version 1.0.

Michael Wayne Goodman. 2019. A python library for deep linguistic resources. In *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*, Singapore.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. 2020. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference 2020*, pages 1253–1263.

Diego Granziol, Stefan Zohren, and Stephen Roberts. 2022. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Philip A. Huebner, Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. BabyBERTa: Learning more grammar with small-scale child-directed language. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 624–646, Online. Association for Computational Linguistics.

Magnus Jacobsen, Mikkel H Sørensen, and Leon Derczynski. 2021. Optimal size-performance tradeoffs: Weighing pos tagger models. *arXiv preprint arXiv:2104.07951*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated graph sequence neural networks. In *Proceedings of ICLR'16*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Chun Hei Lo, Hong Cheng, Wai Lam, and Guy Emerson. 2023. Functional distributional semantics at scale. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 423–436, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2024. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36.

Ewa Muszynska. 2020. *Semantic chunking*. Ph.D. thesis, University of Cambridge.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *LREC*, pages 1250–1255.

Jakob Prange, Nathan Schneider, and Lingpeng Kong. 2022. Linguistic frameworks go toe-to-toe at neuro-symbolic language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4375–4391, Seattle, United States. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. 2016. RELPRON: A relative clause evaluation data set for compositional distributional semantics. *Computational Linguistics*, 42(4):661–701.

Alex Rosenfeld and Katrin Erk. 2022. An analysis of property inference methods. *Natural Language Engineering*, pages 1–27.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online. Association for Computational Linguistics.

Shaeke Salman and Xiuwen Liu. 2019. Overfitting mechanism and avoidance in deep neural networks. *arXiv preprint arXiv:1901.06566*.

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.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, and Nako Sung. 2022. On the effect of pretraining

corpora on in-context learning by a large-scale language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5168–5186, Seattle, United States. Association for Computational Linguistics.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2022. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*.

Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. 2020. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*.

University at Buffalo. Center for Computational Research.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2024. Position: Will we run out of data? limits of llm scaling based on human-generated data. In *Forty-first International Conference on Machine Learning*.

Andreas Waldis, Yotam Perlitz, Leshem Choshen, Yufang Hou, and Iryna Gurevych. 2024. Holmes: Benchmark the linguistic competence of language models. *arXiv preprint arXiv:2404.18923*.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2023. Should you mask 15% in masked language modeling? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2985–3000, Dubrovnik, Croatia. Association for Computational Linguistics.

Aaron Steven White, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2018. Lexicosyntactic inference in neural models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4717–4724, Brussels, Belgium. Association for Computational Linguistics.

Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. 2021a. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021b. Infusing finetuning with semantic dependencies. *Transactions of the Association for Computational Linguistics*, 9:226–242.

Mengzhou Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and Veselin Stoyanov. 2023. Training trajectories of language models across scales. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13711–13738, Toronto, Canada. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. 2021. Syntax-enhanced pre-trained model. 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 5412–5422, Online. Association for Computational Linguistics.

Olga Zamaraeva and Carlos Gómez-Rodríguez. 2024. Revisiting supertagging for faster HPSG parsing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11359–11374, Miami, Florida, USA. Association for Computational Linguistics.

Olga Zamaraeva, Lorena S. Allegue, and Carlos Gómez-Rodríguez. 2024. Spanish resource grammar version 2023. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15093–15104, Torino, Italia. ELRA and ICCL.

Shuaicheng Zhang, Qiang Ning, and Lifu Huang. 2022. Extracting temporal event relation with syntax-guided graph transformer. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 379–390, Seattle, United States. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware bert for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9628–9635.

Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4450–4461, Online. Association for Computational Linguistics.

Term	Hypernym	Properties	Corresponding Templates
telescope	*device*	*astronomers use*	"A device that astronomers use is a telscope"
		observatory has	"A device that an observatory has is a telescope"
		dome houses	"A device that a dome houses is a telescope"
		observer points	"A device that an observer points is a telescope"
		has a mirror	"A device that has a mirror is a telescope"
		uses a lens	"A device that uses a lens is a telescope"
		detects planets	"A device that detects planets is a telescope"
		views stars	"A device that views stars is a telescope"
		tracks the sky	"A device that tracks the sky is a telescope"
		collects light	"A device that collects light is a telescope"
assignment	*document*	*student writes*	"A document that a student writes is an assignment"
		student submits	"A document that a student submits is an assignment"
		teacher reads	"A document that a teacher reads is an assignment"
		receives a grade	"A document that receives a grade is an assignment"
ruin	*building*	*archaeologist discovers*	"A building that an archaeologist discovers is a ruin"
		dig excavates	"A building that a dig excavates is a ruin"
		archaeologist studies	"A building that an archaeologist studies is a ruin"
		collector restores	"A building that a collector restores is a ruin"
		jungle covers	"A building that the jungle covers is a ruin"
		excavation reveals	"A building that excavation reveals is a ruin"

Table 2: RELPRON dataset entries for the terms *telescope*, *assignment*, and *ruin*, including their respective hypernyms, properties, and the verbalized templates derived from each (term, hypernym, property) triple.

Sentence	Label
A particular person didn't mean to do a particular thing	1
Someone didn't tell a particular person to do a particular thing	0
John wasn't upset that a particular thing happened	1
John didn't find that a particular thing happened	0
A particular person was thrilled to do a particular thing	1
A particular person yearned to have a particular thing	0

Table 3: Examples of sentences (with subordinate clauses underlined) and their corresponding labels from the MegaVeridicality 2.1 dataset. A label of 1 indicates that the subordinate event is portrayed as true, while a label of 0 indicates that is not.

Feature	Value
a-utensil	0.634 (19/30)
found-in-kitchens	0.600 (18/30)
used-with-forks	0.534 (16/30)
a-cutlery	0.500 (15/30)
is-dangerous	0.467 (14/30)
a-weapon	0.367 (11/30)

Table 4: (McRae et al., 2005) feature norms for the concept *knife* (feature values are obtained from experiment participants' judgments). For all other features Q, $F(knife)_Q = 0$.

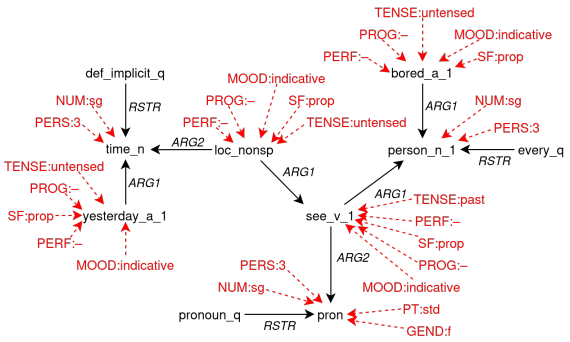

Figure 7: DMRS graph from Figure 1, with node features included. Features are highlighted in red; a dashed arrow $\phi \rightarrow x$ indicates that ϕ is a feature of the node x.

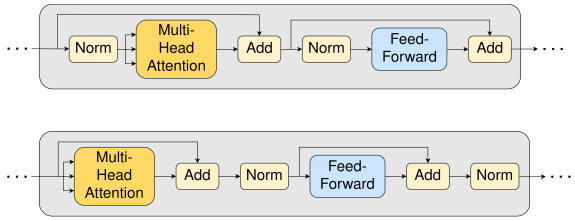

Figure 8: Architecture of a GFoLDS (top) and BERT (bottom) encoder layer.

A Encoder Architectural Details

As discussed in Section 3.1, the outputs of the embedding layer and positional encoding network ($E(X, G)$ and $P(E(X, G), G)$, respectively) are summed together and passed to the encoder stack.

The encoder layers in the GFoLDS architecture are similar to those in BERT and Vaswani et al. (2017)[9], but contain a few key differences—in particular with respect to the residual connections and interrelated layer normalization. As shown in Figure 8, in a BERT encoder layer, the layer input is directly passed to the multi-head attention module, immediately followed by a skip connection and layer normalization. The output of this first layer normalization is then passed to the feed-forward layer, which is again immediately followed by a skip connection and second layer normalization.

In a GFoLDS encoder layer, the input is first layer-normalized *before* being passed to the multi-head attention module, which is followed by a skip connection. The next skip connection—in contrast to BERT—is copied *before* layer normalization, which itself is followed by the feed-forward layer (which is identical to a BERT encoder feed-forward layer) and a second skip connection; this skip connection is *not* followed by layer normalization.

[9]Which is also the architecture employed in Wu et al.'s (2021a) original formulation of the graph transformer.

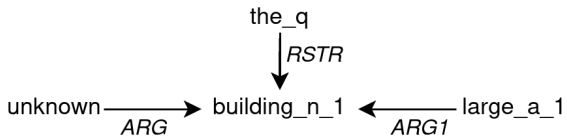

Figure 9: DMRS representation of the noun phrase "*the large building*".

These architectural differences are motivated by the fact that—since the introduction of BERT—normalization *outside* of the residual connection (i.e. $Norm(x + f(x))$) has been shown to be problematic. Godbole et al. (2023) instead recommend normalization *inside* the residual (i.e. $x + f(Norm(x))$), which we implemented in GFoLDS.

B Data Preprocessing

After removing CARGs and OOV items from the graph structures (see Section 3.2), there remain a few—relatively less significant—preprocessing steps that we took in order to transform DMRS representations into inputs for the GFoLDS model.

For the sake of semantic well-formedness, the ACE/ERG parser attempts to represent all inputs as if they were entire sentences. For example, given the input "*the large building*", the parser will parse the noun phrase, then insert the predicate *unknown*[10] and an edge *ARG*: *unknown* $\rightarrow$ *building_n_1* (see Figure 9), which indicates that there is some unknown (presumably verbal) predicate for which the building plays an (again unknown) semantic role (indicated by *ARG*). Such constructions are the *only* context in which the *ARG* edge label appears in DMRS.

While this representational choice is sensible from the perspective of formal semantics, it is undesirable from the viewpoint of machine learning. The fact that *ARG* only links *unknown* to other predicates (and is *always* included when *unknown* is in the graph) makes that predicate extremely predictable: if, during pretraining, the model is given a graph with a masked *unknown* predicate, it needs only look for the *ARG*-labeled edge to know that *unknown* is the masked node. This means that the model does not need to learn any co-occurrence relations between *unknown* and other nodes in the graph in order to learn to reliably predict the distribution of *unknown*.

[10]Not to be confused with out-of-vocabulary/"unknown" items, which are represented in a different manner in DMRS.

Original Label	Interpretation/Role	Replacement
ARG1	First-place argument	—
ARG2	Second-place argument	—
ARG3	Third-place argument	—
ARG4	Fourth-place argument	—
MOD	Indicates a shared handle between two predicates	—
RSTR	Restriction of a quantifier	—
ARG	Argument of the "*unknown*" predicate	*MOD*
L-INDEX	Left-hand conjunct of two coordinated variables	*INDEX*
R-INDEX	Right-hand conjunct of two coordinated variables	*INDEX*
L-HNDL	Left-hand conjunct of two coordinated handles	*HNDL*
R-HNDL	Right-hand conjunct of two coordinated handles	*HNDL*

Table 5: DMRS edge labels (left), the role that they play in describing meaning (center), and the edge labels that they are replaced with (right) during preprocessing (if any: " — " indicates that a label is retained after preprocessing).

Second, recall that each unique argument label is assigned unique forward and backward $d_{SWA} \times d_{SWA}$ edge projection layers in each SWA layer (see Section 3.1). This is to say that each unique edge label corresponds to $2n(d_{SWA})^2$ parameters in the GFoLDS model, where n denotes the number of SWA layers. If, for example, $n = 2$ and $d_{SWA} = 1024$, then each additional edge label adds 4,194,304 parameters to the model architecture. Given the highly specialized function of the *ARG* edge label, it seems rather unreasonable to allocate so many parameters to its representation.

Therefore, during preprocessing, we converted each *ARG* label to the *MOD* label (see Table 5): another purely structural DMRS edge label that is used to indicate handle equality between predicates when other argument-label edges alone are insufficient to do so (Muszynska, 2020).

Additionally, we equivalence-classed argument labels involved in coordination structures (corresponding to logical conjunction and disjunction): for a predicate such as *and_c*, DMRS includes the argument labels *L-HNDL*: *and_c* $\rightarrow X$ and *R-HNDL*: *and_c* $\rightarrow Y$ (*L-INDEX* and *R-INDEX*, respectively, when the conjuncts are variables rather than handles) denoting the left- and right-hand conjuncts X/Y (respectively) of the coordinated structure. Logically, however, conjunction and disjunction are commutative operators: $\phi \wedge \psi = \psi \wedge \phi$ and $\phi \vee \psi = \psi \vee \phi$. Therefore, we replaced the edge labels *L-HNDL*/*R-HNDL* and *L-INDEX*/*R-INDEX* with *HNDL* and *INDEX* (respectively; see Table 5), ignoring the surface order of the conjuncts. This preprocessing step has the added benefit of reducing the overall size of the GFoLDS model by $4n(d_{SWA})^2$ parameters, as discussed above.

Finally, we removed from the graph structures all instances of *focus_d* and *parg_d*: predicates with a purely discourse-pragmatic role, which indicate that the predicates that they modify are focus-topicalized or the subject of a passivized verb (respectively).

C Pretraining Details

This section describes additional details and hyperparameters of the pretraining processes for GFoLDS (Section C.1) and the BERT comparison models used in Sections 5-4 (Section C.2).

C.1 GFoLDS

Pretraining Objective During BERT's pretraining procedure, while 15% of the input tokens are *selected* for prediction, only 80% of the selected tokens are masked: this is to account for the mismatch between the model's pretraining and fine-tuning distributions that arises from the fact that the [MASK] token only occurs during pretraining.

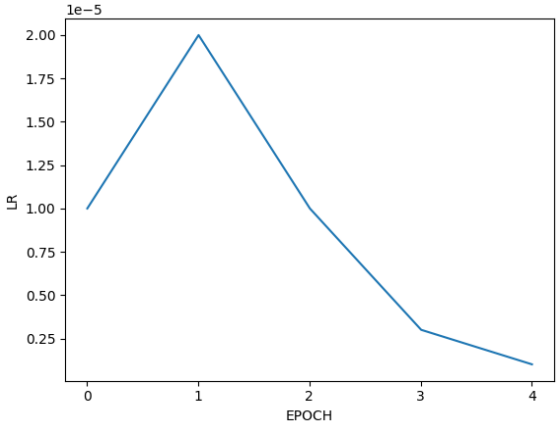

Figure 10: GFoLDS pretraining learn rate schedule.

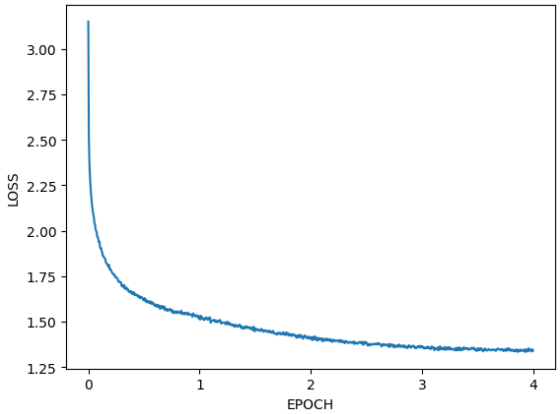

Figure 11: GFoLDS pretraining cross-entropy loss.

However, for GFoLDS, the [MASK] token *does* occur during fine-tuning as well, due to OOV items (as discussed in Section 3.2). Furthermore, Wettig et al. (2023) find that higher selection rates—and higher *masking* rates—result in improved performance on downstream tasks, when compared to the selection and masking/replacement rates reported in Devlin et al. (2019). For these reasons, we chose to mask 100% of the selected tokens during pretraining, with the slightly higher selection probability of 20%.

The MNM prediction head that we used is identical to BERT's MLM prediction head (aside from the difference in vocabulary size): a $d_{model} \times d_{model}$ linear layer, followed by GeLU activation, layer norm, and a $d_{model} \times 22077$ (the size of the vocabulary) linear layer.

Hyperparameters We pretrained GFoLDS with a batch size of 16 for four epochs with the AdamW optimizer (Loshchilov and Hutter, 2017) and a weight decay value of 10^{-5}. We set an initial learn rate of 10^{-5}, with a linearly interpolated learn rate between values of 2×10^{-5} at the end of the first epoch, 10^{-5} at the end of the second, 3×10^{-6} at the end of the third, and 10^{-6} at the end of the fourth (see Figure 10). That is to say that the learn rate increased linearly from 10^{-5} to 2×10^{-5} during the first epoch, decreased linearly from 2×10^{-5} to 10^{-5} during the second epoch, and so on.

At the specified batch size of 16, the model trained at a rate of roughly 25 hours and 36 minutes per epoch on a single NVIDIA A100 GPU, for a total training time of 102 hours and 24 minutes. GFoLDS converged to a cross-entropy loss of $\sim$1.3331 at the end of the fourth epoch (see Figure 11).

C.2 BERT Comparison Models

Given the differences in size and modality between the BERT and GFoLDS models, the best-performing set of pretraining hyperparameters for BERT on this dataset is not likely to be identical to those of GFoLDS. We therefore evaluated a variety of different hyperparameter configurations for BERT in order to yield the most equitable comparison with GFoLDS. Due to the relatively higher cost associated with training $\text{BERT}_{\text{large}}$ (over three times larger than $\text{BERT}_{\text{base}}$), we performed the majority of the trials with $\text{BERT}_{\text{base}}$, then transferred the best-performing configuration found during these experiments to $\text{BERT}_{\text{large}}$.

We first evaluated $\text{BERT}_{\text{base}}$ on three different configurations across which the learn rate schedule, weight decay, and masking rates varied: all three configurations employed the next sentence prediction (NSP) secondary pretraining task (Devlin et al., 2019). Due to hardware constraints, we were limited to using a batch size of 16 for all of the BERT pretraining trials.

The first configuration (a) was identical to that which we employed for GFoLDS (see Appendix C.1): a weight decay value of 10^{-5}; a linearly interpolated learn rate between values of 2×10^{-5} at the end of the first epoch, 10^{-5} at the end of the second, 3×10^{-6} at the end of the third, and 10^{-6} at the end of the fourth (with an initial learn rate of 10^{-5}); and a token selection probability of 20% with a masking probability of 100%.

In the second configuration (b), we used a hyperparameter configuration that was identical to that of the original BERT models: a weight decay value of 10^{-2}; linear learn rate warmup to 10^{-4} across the first 1% of the training run (with linear

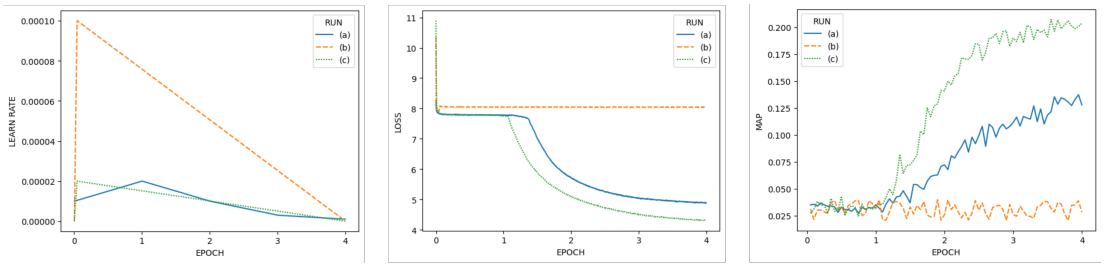

Figure 12: Learning rates (left), cross-entropy loss values (center), and MAP scores on the RELPRON development split (right) across training steps for the $\text{BERT}_{\text{base}}$ pretraining trials (a)-(c).

decay thereafter); and a token selection probability of 15% with a masking and replacement rates of 80% and 10%, respectively.

However, the original BERT models were pretrained with a batch size of 256, while trials (a) and (b) use a batch size of 16. Although we were not able to increase the batch size due to hardware constraints (as mentioned above), Granziol et al. (2022) show that proportional (to batch size) learn rate scaling can be used to control for the effect of batch size on training loss. We therefore introduced a third trial (c): this configuration was identical to that of (b) with the exception of the peak learn rate value, which we scaled down to 2×10^{-5} to account for the difference in batch size.

We then evaluated trials (a)-(c) with respect to learn rate and a validation task that does not require fine-tuning: the development split of the RELPRON (Rimell et al., 2016) dataset (see Section 4.2.1). The results of these experiments are shown in Figure 12.

The model clearly failed to learn with the original BERT pretraining hyperparameters (b)—likely due to the mismatch in batch size discussed above—and finished training with a minimum cross-entropy loss of 7.8864 and a peak MAP score of 0.040 on the RELPRON development set. Of the two remaining configurations, (c) outperforms (a) both in terms of minimum cross-entropy (4.2988 vs. 4.8716) and peak MAP score (0.207 vs. 0.137).

Liu et al. (2019) suggest that pretraining with the secondary NSP objective does not improve (and in some cases may even hinder) model performance. We therefore conducted a fourth hyperparameter trial with $\text{BERT}_{\text{base}}$, using the same configuration as in (c) above, but excluding the NSP task. The results of this experiment are shown in Figure 13.

The variant of configuration (c) without NSP outperforms the original trial in terms of cross-entropy loss (3.8339 vs. 4.2988, respectively; see Figure 13)—this is to be expected: the loss values reported for the variant with NSP are the sum of the NSP loss with the masked language modeling (MLM) loss. However, the non-NSP configuration also outperforms its NSP counterpart in terms of peak MAP score on the RELPRON development set: 0.267 vs. 0.207 (respectively). For this reason, we selected the non-NSP variant of the model pretrained with hyperparameter configuration (c) as the $\text{BERT}_{\text{base}}$ comparison model to be used in the experiments in this work.

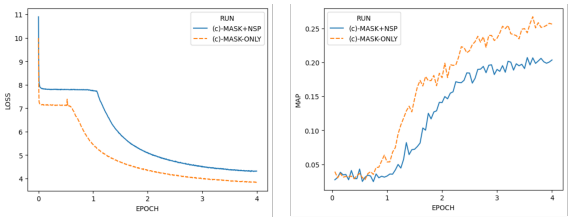

Figure 13: Cross-entropy loss values (left), and MAP scores on the RELPRON development split (right) across training steps for the $\text{BERT}_{\text{base}}$ pretraining configuration (c) with and without the secondary NSP objective.

We then pretrained $\text{BERT}_{\text{large}}$ with (non-NSP) hyperparameter configuration (c). As shown in Figure 14 (configuration large-(a)), this model failed to converge: it finished training with a minimum cross-entropy loss of 7.1856 and a peak MAP score of 0.039 on the RELPRON development set. As larger neural networks are more prone to overfitting (Caruana et al., 2000; Salman and Liu, 2019), we scaled the peak learn rate by a factor of $1/10$ in trial large-(b) to account for the difference in size between the $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ models.

Trial large-(b) vastly outperformed large-(a) in terms of both minimum cross-entropy loss (5.1998 vs. 7.1856) and peak RELPRON development split MAP score (0.039 vs. 0.083), although it trails far behind the best-performing $\text{BERT}_{\text{base}}$ configuration by both metrics (3.8339 vs. 5.1998 cross-entropy;

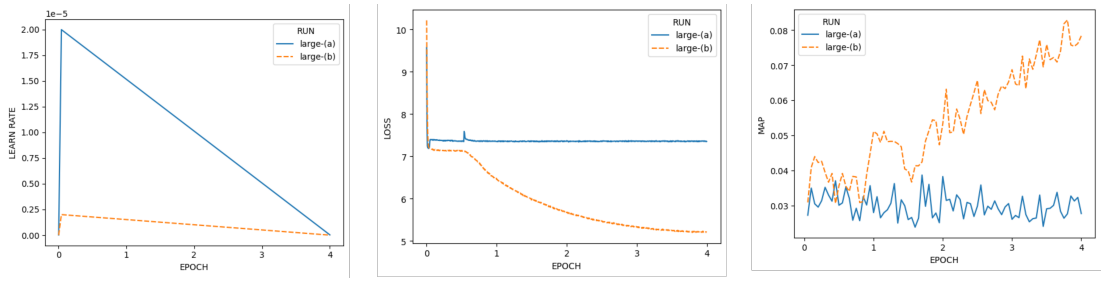

Figure 14: Learning rates (left), cross-entropy loss values (center), and MAP scores on the RELPRON development split (right) across training steps for the $\text{BERT}_{\text{large}}$ pretraining trials (a)-(b).

0.267 vs. 0.083 RELPRON MAP). This substantial difference in performance between the base and large BERT variants is to be expected: it is well-known that larger neural networks require more training data in order to properly converge (see e.g. Hoffmann et al., 2022; Muennighoff et al., 2024), and $\text{BERT}_{\text{large}}$ has over triple the amount of parameters (and double the number of encoder layers) as the base version of the model. Further compounding this issue is the fact that we pretrained these comparison models on 6.5 times less data (and for ten times fewer epochs) than was intended for the BERT models.

With a batch size of 16, $\text{BERT}_{\text{large}}$ trained at a rate of roughly 35 hours and 20 minutes per epoch on a single NVIDIA H100 GPU, for a total training time of 141 hours and 20 minutes. For comparision, GFoLDS and $\text{BERT}_{\text{base}}$—each trained on an NVIDIA A100—required 102 hours and 24 minutes, and 46 hours and 36 minutes, respectively (total training time).

D SNLI Fine-Tuning Details

We fine-tuned GFoLDS and $\text{BERT}_{\text{base}}$ for five epochs with a batch size of 16, a weight decay value of 10^{-5}, an initial learn rate of 10^{-5}, and a linearly-interpolated learn rate (updated at each batch) between values of 2×10^{-5} at the end of the first epoch, 3×10^{-5} at the end of the third, 10^{-6} at the end of the fourth, and 10^{-7} at the end of the fifth (see Figure 15). We fine-tuned $\text{BERT}_{\text{large}}$ with identical hyperparameters, except all learn rates mentioned above were multiplied by 1/10 for this model ($\text{BERT}_{\text{large}}$ was unstable with the higher learn rate used for GFoLDS and $\text{BERT}_{\text{base}}$).

The fine-tuning hyperparameters for the BERT models were admittedly not selected using as rigorous of a search procedure as that employed during pretraining (see Appendix C.2). However, the accuracy that the (original) $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$

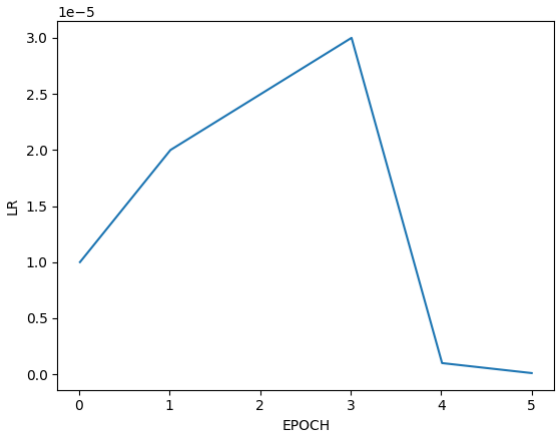

Figure 15: SNLI fine-tuning learn rate schedule for GFoLDS and the $\text{BERT}_{\text{base}}$ models.

models attained with the hyperparameter configurations described in the above paragraph (see Table 1) matched that reported for those models on the SNLI dataset in Zhang et al. (2020). It is therefore reasonably safe to assume that this set of hyperparameters is (near-)optimal for the BERT models with respect to this data.

E MegaVeridicality V2.1 Task Setup

We converted this dataset to a binary classification task by assigning a values of 1, 0, and -1 to the labels *yes*, *maybe*, and *no* (respectively). We then assigned each example a value of 1 (i.e. the subordinate event is portrayed as true) if its mean value was greater than zero, and 0 otherwise.

F Elementary Tasks

POS-prediction As discussed in Section 4.2, we evaluated the models on 200 sentences drawn from English Wikipedia. We first parsed each sentence using the ACE/ERG DMRS parser (see Section 3.2), which automatically labels the part-of-speech of each predicate in the DMRS representation of a

Type	Quantifiers
SG	another, either, neither, that, this, every, a(n), each
PL	these, certain, most, those, all, such, both
BOTH	some, the, any, enough, no, which

Table 6: Quantifier types—along with a list of the quantifiers belonging to each type—used in the quantifier-agreement task.

sentence. We then randomly selected a single word to mask from each parsed sentence—subject to the condition that the selected word must be mapped to a single token by the BERT tokenizer (in order to facilitate the evaluation of the BERT comparison models)—and recorded to part-of-speech of the selected word. This resulted in a dataset consisting of 58 masked-quantifier, 28 masked-preposition, 33 masked-verb, 63 masked-noun, and 18 masked-adjective sentences.

Determining the part-of-speech for each prediction of the GFoLDS model was trivial: DMRS predicates include part-of-speech tags, so we simply checked the tag of each predicted predicate. The BERT models, however, necessitated the use of the NLTK POS tagger[11]. For each sentence s, and each of the model's top-ten predicted tokens w for s, we created a new sentence s_w by replacing the masked word of s with the prediction w, and ran the NLTK POS tagger over s_w to obtain the tag for w. Note that, while the NTLK POS tagger is not perfect, it does achieve 95+% accuracy on English-language data (Jacobsen et al., 2021), and therefore is sufficiently robust to yield an estimate of the model's performance.

We chose to use (bounded) precision as the evaluation metric for this task because of the large amounts of positive examples for each class (especially nouns, verbs, and adjectives), which precluded the calculation of metrics that incorporate false negatives (e.g. recall and F1). We recorded each model's mean precision across all 200 sentences as its final score for this task.

Quantifier-agreement We first parsed each sentence with the ACE/ERG parser, which explicitly labels the number of each noun: this allowed the automatic extraction of the number of the noun in the restriction of a given quantifier. We then randomly selected a single quantifier from each sentence to mask, and recorded the number of the noun in the quantifier's restriction.

We sorted all of the quantifiers into one of three categories/types (see Table 6): *singular* (can only restrict singular nouns), *plural* (can only restrict plural nouns), and *both* (can restrict either kind of noun). Note that the both-type quantifiers were used only for evaluation: the type of the masked quantifiers was recorded only as singular or plural—all nouns are either singular or plural, and the target type was determined by the number assigned by the ACE/ERG parser to the noun in the quantifier's restriction.

When computing precision for this task, all non-quantifier words in the models' top-k predictions were treated as false positives, while both-type quantifiers in the top k were treated as true positives, regardless of the target type (singular or plural). As in the POS-prediction task above, we recorded each model's mean precision across all 179 sentences as its final score for this task.

G Scalability

G.1 Background and Notation

(Hoffmann et al., 2022) investigate the relationship between model size (i.e. number of parameters) and amount of pretraining data, and the final pretraining cross-entropy loss of LLMs. The optimal number of parameters and amount of data (number of tokens) for a fixed compute budget—expressed in terms of floating-point operations (FLOPs)—is given in Equation 3, where N_{opt} denotes the optimal number of model parameters, D_{opt} the optimal number of training tokens, C the compute budget, and $L(N, D)$ the model's final pretraining loss on D tokens with N parameters.

$$(N_{opt}, D_{opt}) = \underset{(N,D)\,:\,FLOPs(N,D)=C}{argmin} L(N, D) \tag{3}$$

The authors fit a model of LLM final pretraining loss as a function of N and D. This model is given in Equation 4, where E, A, B, α, and β are learned constants[12].

[11] https://www.nltk.org/api/nltk.tag.pos_tag.html

[12] $E = 1.69$, $A = 406.4$, $B = 410.7$, $\alpha = 0.34$, $\beta = 0.28$

$$L(N, D) \approx \hat{L}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \quad (4)$$

However, (Hoffmann et al., 2022) only consider *unique* training data—i.e. pretraining a language model for a single epoch. (Muennighoff et al., 2024) extend Hoffmann et al.'s (2022) experiments to the case of *repeated* training data: pretraining the model for multiple epochs on the same dataset. The authors fit an analogous loss-prediction function to that given in Equation 4 (Equation 5a), $\hat{L}^{(r)}(N, U_D)$, which estimates the final loss for a language model with N parameters trained on U_D tokens for r repetitions (i.e. epochs).

$$L^{(r)}(N, U_D) \approx \hat{L}^{(r)}(N, U_D) = E + \frac{A}{\hat{N}^{\alpha}} + \frac{B}{\hat{D}^{\beta}} \quad (5a)$$

$$\hat{D} = U_D + U_D R_D^*(1 - e^{-r/R_D^*}) \quad (5b)$$

$$\hat{N} = U_N + U_N R_N^*(1 - e^{-R_N/R_N^*}) \quad (5c)$$

$$U_N = \mathit{min}\{N, N_{\mathit{opt}}(U_D)\} \quad (5d)$$

$$R_N = \frac{N}{U_N} - 1 \quad (5e)$$

Where U_D denotes the number of *unique* tokens (i.e. the amount of tokens in a single epoch), $N_{\mathit{opt}}(U_D)$ is estimated as in (Hoffmann et al., 2022) (see Equations 3 and 4), and E, A, B, α, β, R_D^*, and R_N^* are learned constants[13].

G.2 Proof

The term D in Equation 5 denotes the number of pretraining *tokens*. This metric is likely not a perfect predictor for GFoLDS: recall from Section 3.1 that the model contains separate projection layers for each DMRS edge label type. The model therefore receives training signal not only from the node labels (i.e. tokens), but also from the *edge* labels. For example, an input graph with six tokens and five edges will update fewer model parameters than a graph with six tokens and ten edges.

It is beyond the scope of this work to establish exact scaling laws for GFoLDS and determine the graph-based analogue to the term U_D in Equation 5. However, it is clear that such a U_D value scales (more or less) linearly with the number of graphs in the pretraining dataset: as with a textual model, U_D can be expressed as the sum of the amounts of data (regardless of how it is quantified) contributed by each individual input graph (respectively, sequence) in the dataset.

It is therefore reasonable to assume that $U_{D'} \approx U_D/2$, where U_D denotes the dataset for the 100% run, and $U_{D'}$ that of the 50% run (see Section 6). We may express the relationship between the 50% run loss and the 100% loss in the notation introduced in Appendix G.1 (while leaving the exact definition of U_D to future work): $L^{(4)}(N, U_D) \approx L^{(4)}(N, U_D/2)$. Recall from Equation 5a that the term E is a constant, and so can be ignored. We are then left with the following (approximate) equality in Equation 6a.

$$\frac{A}{\hat{N}_1^{\alpha}} + \frac{B}{\hat{D}_1^{\beta}} \approx \frac{A}{\hat{N}_2^{\alpha}} + \frac{B}{\hat{D}_2^{\beta}} \quad (6a)$$

$$\hat{D}_1 = U_D + U_D R_D^*(1 - e^{-4/R_D^*}) \quad (6b)$$

$$\hat{D}_2 \approx (U_D/2) + (U_D/2) R_D^*(1 - e^{-4/R_D^*}) \quad (6c)$$

$$\hat{N}_1 = U_N^{(1)} + U_N^{(1)} R_N^*(1 - e^{-R_N^{(1)}/R_N^*}) \quad (6d)$$

$$\hat{N}_2 = U_N^{(2)} + U_N^{(2)} R_N^*(1 - e^{-R_N^{(2)}/R_N^*}) \quad (6e)$$

$$U_N^{(1)} = \mathit{min}\{N, N_{\mathit{opt}}(U_D)\} \quad (6f)$$

$$U_N^{(2)} \approx \mathit{min}\{N, N_{\mathit{opt}}(U_D/2)\} \quad (6g)$$

$$R_N^{(k)} = \frac{N}{U_N^{(k)}} - 1 \quad (6h)$$

Where the left-hand expression in Equation 6a corresponds to the 100% run ($\hat{L}^{(4)}(N, U_D)$), and the right-hand expression corresponds to the 50% run ($\hat{L}^{(4)}(N, U_D/2)$). Note that $\hat{D}_2 \approx \hat{D}_1/2$ (Equation 7).

$$\begin{aligned} \hat{D}_2 &\approx \frac{U_D}{2} + \frac{U_D}{2} R_D^*(1 - e^{-4/R_D^*}) \\ &= \frac{U_D + U_D R_D^*(1 - e^{-4/R_D^*})}{2} = \frac{\hat{D}_1}{2} \end{aligned} \quad (7)$$

Given the vast architectural and modal differences between GFoLDS and textual transformer models, we do not assume that the coefficients E, A, B, α, β, R_D^*, and R_N^* fitted in (Muennighoff et al., 2024) are identical for GFoLDS. Moreover, as stated above, it is beyond the scope of this work to establish exact scaling laws for this model. However, from the fact that final loss decreases as pretraining data increases from the 3.125% to the 50% runs (see Section 6.2), we know that the coefficients B and β in 6 must be positive: this—in

[13] $E = 1.88$, $A = 523.22$, $B = 1480.30$, $\alpha = 0.35$, $\beta = 0.35$, $R_D^* = 15.39$, $R_N^* = 5.31$

conjunction with the (approximate) equality in Equation 7—means that it cannot be the case that $B/\hat{D}_1^\beta > B/\hat{D}_2^\beta$.

Assume that $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$: given the equality in Equation 6a, it must then be the case that $A/\hat{N}_1^\alpha - A/\hat{N}_2^\alpha \approx B/\hat{D}_2^\beta - B/\hat{D}_1^\beta$, and therefore that $A/\hat{N}_2^\alpha \ll A/\hat{N}_1^\alpha$.

Assume further that the model is underparameterized at 100% of the data (i.e. that $N < N_{opt}(U_D)$): then $U_N^{(1)} = min\{N, N_{opt}(U_D)\} = N$ and $R_N^{(1)} = (N/U_N^{(1)}) - 1 = 0$, which implies that $\hat{N}_1 = U_N^{(1)} + U_N^{(1)}(1-1) = U_N^{(1)} = N$ (see Equation 6d). As the terms A and α are constants in Equation 5a (and therefore in Equation 6a), we may reduce to the inequality expressed in Equation 8 (where the last logical equivalence is by definition of $R_N^{(2)}$; see Equation 6h).

$$\begin{aligned}
&\hat{N}_1^{-\alpha} \gg \hat{N}_2^{-\alpha} \leftrightarrow \hat{N}_1 \ll \hat{N}_2 \\
&\leftrightarrow N \ll U_N^{(2)} + U_N^{(2)} R_N^*(1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow \frac{N - U_N^{(2)}}{U_N^{(2)}} \ll R_N^*(1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow \frac{N}{U_N^{(2)}} - 1 \ll R_N^*(1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow R_N^{(2)}/R_N^* \ll 1 - e^{-R_N^{(2)}/R_N^*}
\end{aligned} \tag{8}$$

But this is impossible: by Bernoulli's inequality, $1 + x \leq e^x$ for all x. This implies that $x \leq e^x - 1$, which in turn implies $-x \leq 1 - e^x$, which then implies $x > 1 - e^{-x}$. As it therefore cannot be the case that $R_N^{(2)}/R_N^* < 1 - e^{-R_N^{(2)}/R_N^*}$, we know that the model cannot be underparameterized if $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$.

Now assume that the model is either over- or well-parameterized at 100% of the data (i.e. that $N \geq N_{opt}(U_D)$): then $U_N^{(1)} = min\{N, N_{opt}(U_D)\} = N_{opt}(U_D)$. It is reasonable to assume that $N_{opt}(U_D/2) < N_{opt}(U_D)$ (i.e. that the model's optimal number of parameters scales monotonically with the amount of pretraining data), which implies that $U_N^{(2)} = min\{N, N_{opt}(U_D/2)\} = N_{opt}(U_D/2) < N_{opt}(U_D) = U_N^{(1)}$. With a fixed number of parameters N, $\hat{N}$ (Equations 5c, 6d-e) is a monotonic function of U_N (by construction; Muennighoff et al., 2024), so we have $U_N^{(1)} > U_N^{(2)} \rightarrow \hat{N}_1 > \hat{N}_2$.

But this implies that $A/\hat{N}_1^\alpha < A/\hat{N}_2^\alpha$. This—along with the assumption that $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$—contradicts the observed result that $L^{(4)}(N, U_D) \approx L^{(4)}(N, U_D/2)$, and so we know that the model cannot be well- or over-parameterized if $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$.

Given that it cannot be the case that the model of the 100% run is under-, well-, or over-parameterized if $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$ (and that it must be one of the three), it therefore cannot be the case that $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$. As argued above, the coefficients B and β must be positive, so it also cannot be the case that $B/\hat{D}_1^\beta \gg B/\hat{D}_2^\beta$: we must then conclude that $B/\hat{D}_1^\beta \approx B/\hat{D}_2^\beta$. As $\hat{D}_1 \approx 2\hat{D}_2$ (see Equation 7), we know that $B/\hat{D}_1^\beta \approx (B/\hat{D}_2^\beta)/2^\beta$, (and that $2^\beta > 1$, as β must be positive). It must therefore be the case that $B/\hat{D}_1^\beta \approx 0$.

Given that $B/\hat{D}_1^\beta \approx B/\hat{D}_2^\beta$, the equality in Equation 6a implies that $A/\hat{N}_1^\alpha \approx A/\hat{N}_2^\alpha$, which in turn implies that $\hat{N}_1 \approx \hat{N}_2$. If GFoLDS were over- or well-parameterized at 100% of the data, then the monotonicity of $\hat{N}$ would imply that $\hat{N}_1 > \hat{N}_2$ (as discussed above), contradicting the conclusion that $\hat{N}_1 \approx \hat{N}_2$.

On the other hand, replacing the inequalities in Equation 8 with (approximate) equalities yields $R_N^{(2)}/R_N^* \approx 1 - e^{-R_N^{(2)}/R_N^*}$: note that the only value of x for which $x = 1 - e^{-x}$ is 0. If GFoLDS is underparameterized at the 50% run, then $U_N^{(2)} = min\{N, N_{opt}(U_D/2)\} = N$, which implies that $R_N^{(2)} = 0$, which in turn implies that $R_N^{(2)}/R_N^* = 0$. Given that $N_{opt}(U_D)$ is (likely) greater than $N_{opt}(U_D/2)$, it is also the case that $R_N^{(1)}/R_N^* = 0$ and $U_N^{(1)} = min\{N, N_{opt}(U_D)\} = N = U_N^{(2)}$: this implies that $A/\hat{N}_1^\alpha = A/\hat{N}_2^\alpha$ (see Equations 6d and 6e), which is congruent with the conclusion that $\hat{N}_1 \approx \hat{N}_2$.