Language Modeling over Logical Forms

by

Michael Sullivan

May 9, 2025

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

Department of Linguistics

Contents

Abstract

This dissertation introduces the research program of language modeling over logical forms: the employment of language models (LMs) that take as input semantic representations. The use of such models is motivated by the Accelerated Learning Hypothesis (ALH), which posits that linguistic information—and semantic structure in particular—has a catalyzing effect on LM (pre-)training, allowing LMs over logical forms to learn with less data than superficial (i.e. over plain text) models. This dissertation additionally presents arguments that the use of logical input representations improves the reasoning abilities of such models. As the current, rapid improvements in large language models' advanced abilities are primarily driven by their leveraging of shallow heuristics and unsustainable rate of data consumption, LMs over logical forms represent a promising new direction for continued progress in the field of Language AI.

The main objectives of this dissertation are twofold: first, to empirically support the validity of the ALH, demonstrating that LMs over logical forms have the potential to learn with less data; and second, to establish the feasibility (of implementation) and utility (i.e. applicability to downstream tasks) of such models, demonstrating their viability as general-purpose LMs. To that end, this dissertation introduces two LMs over logical forms, including the neurosymbolic *Graph-based Formal-Logical Distributional Semantics* (GFoLDS) model.

The pretrained GFoLDS model is applied to a wide range of downstream tasks, vastly outperforming superficial models pretrained on the same amount of data, thereby demonstrating the viability of LMs over logical forms. Additional experiments with the GFoLDS model then provide direct evidence towards the ALH. This dissertation furthermore lays out potential future avenues of research into language modeling over logical forms, outlining the subsequent steps of this research program.

Chapter 1

Introduction

Recent advances in large language models (LLMs) such as Llama-3 (Dubey et al., 2024), DeepSeek-V3 (DeepSeek-AI, 2024), and GPT-3/4 (Brown et al., 2020; OpenAI, 2023) have led to near-human performance on a wide variety of NLP tasks. In some cases, state-of-the-art (SoTA) LLMs even exceed average human performance on tasks intended for human evaluation: for example, GPT-4 scores in the 90$^{\text{th}}$ percentile on both sections of the SAT, above the 50$^{\text{th}}$ percentile on all three sections of the GRE, and in the 88$^{\text{th}}$ percentile on the LSAT (OpenAI, 2023). GPT-4 outperforms its predecessor (GPT-3.5[1]) on nearly all evaluation tasks, and in some cases (e.g. LSAT, GRE Quantitative, etc.) more than doubles GPT-3.5's performance: a staggering rate of improvement, given the mere two-year interval between the respective release dates of these models.

These consistent increases in performance exhibited by SoTA LLMs are largely due to corresponding increases in model size (Muennighoff et al., 2024): GPT-2 (Radford et al., 2018) contains 1.5 billion parameters, while the parameter count of its successor, GPT-3, totals a staggering 175 billion—a more than hundred-fold increase in less than three years[2]. Given the massive parameter counts of SoTA LLMs and the advanced logical reasoning abilities seemingly entailed by these models' performance on tasks such as those discussed

[1]https://openai.com/blog/gpt-3-edit-insert

[2]The GPT-4 architecture has not been publicly released, but the model has been estimated to contain over one trillion parameters.

above, it would not be unreasonable to wonder whether LLMs have achieved some degree of human-like logical reasoning capability. In fact, this point of view is frequently expressed in the literature: for example, Piantadosi and Hill (2022) claim that LLMs "likely capture important aspects of meaning, and moreover work in a way that approximates a compelling account of human cognition in which meaning arises from conceptual role."

There is, however, a wide body of evidence (e.g. McCoy, Pavlick, and Linzen, 2019; Chien and Kalita, 2020; Richardson et al., 2020; Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023, etc.) suggesting that, rather than developing actual, human-like reasoning capabilities, language models (LMs) merely learn to leverage shallow heuristics to attain such remarkable performance on logical-reasoning-oriented tasks. This indicates that while LMs may be able to excel at artificially-created tasks intended to evaluate and improve logical reasoning abilities, this heuristically-derived success cannot adequately translate to efficacy in real-world logical-reasoning scenarios.

Furthermore, while the ever-growing financial cost (in terms of hardware upgrades, electricity, etc.) associated with training and deploying ever-growing LLMs may present a barrier to further advances in the field, there is another rapidly-approaching bottleneck on the horizon: the so-called *Chinchilla Scaling Laws* state that the optimal amount of data required to train an LLM scales proportionally to the model's size (Hoffmann et al., 2022; Muennighoff et al., 2024). Given the Chinchilla Scaling Laws, and the respective rates at which SoTA LLMs and the stock of available language model training data are expanding, Villalobos et al. (2024) estimate that high-quality English training data will be exhausted sometime between the years 2026 and 2032: language model expansion is outpacing available natural language production. This suggests that—without models that can learn with significantly less data than current approaches—LLMs' performance increases will begin to decelerate substantially in the near future.

This does not necessarily mean that progress in the field of Language AI is condemned to stall: *linguistically-informed* language models (LMs augmented by linguistic knowledge)

offer the potential to improve over—without consuming more training data than—*superficial* LMs (i.e. language models over plain text). To be clear, I do not use the term *superficial* to suggest that such LMs have a simpler architecture and/or (pre-)training regimen than other models, but rather to indicate that these models are purely over *surface* text.

A substantial amount of recent research indicates that the injection of linguistic structure into the architecture and/or (pre-)training data of an LM can drastically improve its performance on downstream tasks: for example, Xu et al. (2021) inject syntactic dependency parses into pretrainined transformer encoders—namely, BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019)—improving over the baseline models and achieving (then-)SoTA results on relation classification, entity typing, and question answering tasks. Similarly, Sachan et al. (2021) improve over the SoTA in semantic role labeling and relation extraction through the fusion of dependency parse graphs into a pretrained BERT model.

Of particular interest to this dissertation is the Linguistics-Informed Multi-Task BERT model (LIMIT-BERT; Zhou et al., 2020), a BERT architecture pretrained with additional, syntactically- and semantically-informed objectives[3] that uses the same amount of pretraining data as the original BERT, and outperforms that BERT model on a series of logical-reasoning-oriented tasks, including reading comprehension, natural language inference (NLI), and question-answering. This result is supported by Zhang et al. (2020c), who find that combining semantic role label embeddings with a pretrained BERT model vastly increases its performance on reading comprehension and NLI tasks.

In this dissertation, I argue that the injection of linguistic information into LMs has a catalyzing effect on their learning abilities, allowing linguistically-informed LMs to learn faster than superficial models. In particular, I propose the following hypothesis:

[3]Part-of-speech prediction, semantic role prediction, constituency span identification, and dependency span identification.

The Accelerated Learning Hypothesis:

i. Linguistically-informed LMs immediately begin learning more complex patterns, because the aspects of linguistic knowledge incorporated into such models obviate the need to learn elementary linguistic phenomena.

ii. This accelerated learning of complex patterns allows linguistically-informed LMs to learn from less data than their superficial counterparts.

Critically, part (ii) of the Accelerated Learning Hypothesis (ALH) indicates that that the use of linguistically-informed language models has the potential to mitigate the shortage of LLM training data predicted by the Chinchilla Scaling Laws.

The question, then, is what kind of linguistic knowledge—syntactic or semantic—is most useful for language model augmentation. Wu, Peng, and Smith (2021) combine pretrained RoBERTa models with syntactic and semantic representations, and find that semantic representations yield greater performance increases than syntactic parses on all of their evaluation tasks: reading comprehension, textual similarity, question-answering, and NLI. Extending this result to decoder models, Prange, Schneider, and Kong (2022) demonstrate that GPT-2 equipped with semantic representations outperforms a syntax-augmented GPT-2 model on next-word prediction accuracy and entropy benchmarks. These results suggest that semantic representations present the most beneficial structures for the purposes of augmenting LMs with linguistic knowledge. Furthermore, there is evidence that the use of semantic representations to augment LMs also has the potential to improve these models' reasoning abilities: as discussed above, Zhou et al. (2020), Zhang et al. (2020c), and Wu, Peng, and Smith (2021) find empirically that semantic structure yields improvement over baseline, superficial models on logical reasoning tasks such as reading comprehension, question-answering, and NLI.

In this dissertation, I propose taking the premise of the Accelerated Learning Hypothesis to its extreme, eschewing surface text altogether and employing language models (pre-)trained

solely over linguistic structure. Given the findings of Wu, Peng, and Smith (2021) and Prange, Schneider, and Kong (2022) discussed above, I specifically advocate the use of LMs that take as input semantic representations: language models over logical forms, which I define precisely in Section 1.1.

1.1 Defining Language Models over Logical Forms

In order to formally define a language model over logical forms (Section 1.1.2), I first provide a precise definition of the term *logical form* as used in this dissertation—and further motivate the use of logical forms for language modeling—in Section 1.1.1.

1.1.1 Logical Forms

Definition 1 (Logical Form)**.** *A sentence in a formal language L, such that L carries a predicate-argument structure that can be used to represent (aspects of) truth-conditional linguistic meaning.*

Note that this definition does not necessarily require that logical forms be interpretable within a model structure: for example, sentences in Minimal Recursion Semantics (MRS; Copestake et al., 2005) cannot be directly interpreted in a model due to underspecification, but still fall under Definition 1.

I argue that the primary benefit of logical forms with respect to language modeling is the de-noising effect conferred by the function-argument structure inherent to such representations (by Definition 1)—which can be considered to be elementary linguistic knowledge (see Chapter 3). For example, although the syntactic position of the (proto-)agent (see e.g. Dowty, 1991) of a sentence may vary depending on various structural factors (topicalization, passivization, etc.), the (proto-)agent is always the first-place argument in a semantic representation. The translation of surface text to logical form therefore has an equivalence-classing effect, so that all syntactic paraphrases of the same proposition—for example, an active sentence and its

passive counterpart—are mapped to the same representation. The benefit to a language model 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 (and/or arguments; see Chapter 3).

In a similar vein, certain logical-form representations, including the framework that I employ in this dissertation (MRS; see Chapters 3 and 4), include morphosyntactic features such as number, tense, person, etc., further de-noising the model's input by offloading the morphological realization of these properties to explicitly annotated labels. This is to say that a model over logical forms does not need to learn the surface patterns corresponding to inflection, as this information is instead explicitly provided through the semantic structure. For example, an LM over logical forms does not need to learn that the suffix *–s* denotes a plural noun—or irregular realizations of pluralization, e.g. *goose/geese*—because plural nouns are directly labeled as such.

1.1.2 Language Models

In order to precisely define the term *language model over logical forms*, it is first necessary to establish a definition of the term *language model*. Formally, a language model is a family of joint probability distributions $p(w_1, \ldots, w_n)$ over sequences of words (Li, 2022). Equivalently, we may define a language model as the conditional distribution $p(w_n \mid w_1, \ldots, w_{n-1})$: $p(w_1, \ldots, w_n)$ is then calculated as in Equation 1.1.

$$p(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i \mid w_1, \ldots w_{i-1}) \tag{1.1}$$

However, this definition does not encompass *masked* language models such as BERT (Devlin et al., 2019), in which a portion of the words in a sequence $W = w_1, \ldots, w_n$ are replaced with a [MASK] symbol, and the model yields a probability distribution over each of the masked words. Therefore, I employ a more general definition of a language model

(Definition 2) in this dissertation.

Definition 2 (Language Model)**.** *A probability distribution $p(\hat{w}_i = x \mid \hat{W})$, where $W = w_1, \ldots, w_n$ is a sequence of words, $\hat{W}$ is derived by replacing one or more words in W with the [MASK] symbol, and $\hat{w}_i$ = [MASK].*

Note that the definition in Equation 1.1 is a special case of Definition 2, as demonstrated in Equation 1.2.

$$\begin{aligned} p(w_1, \ldots, w_n) &= p(w_1) \cdot \ldots \cdot p(w_n \mid w_1, \ldots, w_{n-1}) \\ &= p(\hat{w}_1 = w_1 \mid [\text{MASK}]) \cdot \ldots \cdot p(\hat{w}_n = w_n \mid w_1, \ldots, w_{n-1}, [\text{MASK}]) \end{aligned} \tag{1.2}$$

A language model over logical forms can then be defined as a special case of a language model, in which the words w_i are symbols in a formal language (Definition 3).

Definition 3 (Language Model over Logical Forms)**.** *A probability distribution $p(\hat{w}_i = x \mid \hat{W})$, where $W = w_1, \ldots, w_n$ is a sequence of symbols comprising one or more logical forms, $\hat{W}$ is derived by replacing one or more symbols in W with the [MASK] symbol, and $\hat{w}_i$ = [MASK].*

In the remainder of this dissertation, I make the case for LMs over logical forms empirically, introducing two such models and demonstrating via a series of experimental results that these models are able to learn with substantially less data than their superficial counterparts, and are applicable to a wide range of NLP tasks.

1.2 Objectives

This dissertation has two major objectives. I first aim to provide evidence in support of the validity of the Accelerated Learning Hypothesis (ALH): in light of the Chinchilla Scaling Laws and the consequent, impending shortage of LLM training data discussed above, findings in support of the ALH would be of considerable importance to the field of Language AI, opening

possibilities for the continued improvement of large language models at a more sustainable rate of data consumption.

The second—but no less important—goal is to provide a proof-of-concept of the viability of language modeling over logical forms. To be clear, I do not construct a fully-capable LM over logical forms in this dissertation, and I leave the resolution of some limitations to future work. However, I endeavor to demonstrate the *feasibility* and *utility* of language modeling over logical forms in pursuit of this goal.

I use the term *feasibility* to refer to the practicality of implementing an LM over logical forms with currently-available tools: specifically, I show that the noise introduced by the use of an out-of-the-box, rule-based semantic parser does not drastically impair the (pre-)training of these models; that such a model can be implemented using current machine learning frameworks and architectures; and that existing NLP (pre-)training and evaluation techniques can be readily adapted to this modality. *Utility* refers to the downstream applicability of language models over logical forms: I demonstrate that these models can perform the same tasks as a superficial LM, or (in some cases) that it will be feasible to improve the prototype model(s) introduced in this dissertation to be able to do so.

The two primary objectives of this dissertation are highly interdependent: if the ALH does not hold, then the feasibility of language models over logical forms is irrelevant, as they would offer no benefit over existing, superficial LMs. Conversely, if language modeling over logical forms is not feasible, then the ALH would be nothing more than an esoteric conjecture with little practical importance.

1.3 Contributions

This dissertation is interdisciplinary in nature and makes contributions to the fields of Linguistics and Machine Learning, the most important of which being a model architecture and paradigm (the language model over logical forms introduced in Chapter 4) capable of

learning from less data, providing a potential route to overcome the impending bottleneck predicted by the Chinchilla Scaling Laws. I additionally make publicly available the code necessary to parse and process the data, construct and pretrain this model, and reproduce all of the experiments conducted in this dissertation[4].

The second major contribution is the introduction and development of the research program of language modeling over logical forms: I provide theoretical and empirical arguments motivating further research in this area, and introduce two LMs over logical forms. This dissertation also lays out future directions for this research program, outlining the next major step in the evolution of language models over logical forms.

Thirdly, this dissertation provides a deeper understanding of the reasons behind the performance increases yielded by linguistic-knowledge augmentation that are demonstrated in Xu et al. (2021); Sachan et al. (2021); Zhou et al. (2020); Zhang et al. (2020c); Wu, Peng, and Smith (2021); Prange, Schneider, and Kong (2022), etc.: specifically, the Accelerated Learning Hypothesis and the empirical evidence that I present strongly supporting its validity. In a similar vein, the massive improvement in performance and data-efficiency conferred by semantic representations that is empirically demonstrated in this dissertation constitutes evidence suggesting the importance for modeling meaning of the function/argument structure assumed by most semantic theories.

1.4 Outline

The remainder of this dissertation (Chapters 2-8) is organized as follows:

In Chapter 2, I investigate the logcial reasoning abilities of (near-)SoTA superficial natural language inference (NLI) models through the lens of double-negation cancellation. The experimental results of this chapter indicate that superficial models do not learn to reason logically when fine-tuned on NLI datasets, supporting existing work in the literature indicating

[4]Chapter 2: https://github.com/mjs227/FoLDS, Chapter 3: https://github.com/mjs227/AdversarialNLI, Chapters 4-6: https://github.com/mjs227/GFoLDS

that they are instead learning to leverage shallow heuristics (McCoy, Pavlick, and Linzen, 2019; Chien and Kalita, 2020; Richardson et al., 2020; Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023, etc.). I use these results to demonstrate that even relatively simple logical-reasoning tasks continue to confound near-SoTA superficial LMs, thereby providing further motivation for the language models over logical forms that I introduce in Chapters 3 and 4. Chapter 2 also provides the experimental framework that I later use in Chapter 6 to evaluate the logical reasoning abilities of the model that I introduce in Chapter 4, and to probe it for limitations and weaknesses.

Chapter 3 introduces the *Formal-Logical Distributional Semantics* (FoLDS) model: a non-neural model that generates complex-valued count vectors drawn from a fuzzy-logical model world imperatively constructed from logical-forms (i.e. semantic representations). I use FoLDS to demonstrate the feasibility of language modeling over logical forms, and show that FoLDS outperforms superficial models on a downstream task—while using significantly less data. This chapter also provides further arguments and empirical evidence in support of the Accelerated Learning Hypothesis.

However, the FoLDS model suffers from several severe limitations that hinder its utility (i.e. its applicability to a wide range of tasks): in Chapter 3, I use an analysis of the strengths and weaknesses of FoLDS to establish a set of fundamental desiderata to consider when constructing a language model over logical forms.

In Chapter 4, I use the desiderata established in Chapter 3 to motivate the *Graph-based FoLDS* (GFoLDS) model: a modified variant of the transformer encoder architecture (Vaswani et al., 2017) over graph representations of logical forms. I further motivate the particular design choices that I made when implementing this model through an analysis of the existing literature on graph transformers (Wu et al., 2021) and the limitations of graph neural networks (GNNs), the use of GNNs in NLP, and graph-based methods for incorporating linguistic structure/knowledge into language models. I then discuss the pretraining of the GFoLDS model: relevant statistics of its pretraining dataset, the data preprocessing steps that I

employed, and the model's pretraining objective and hyperparameters.

In Chapter 5, I evaluate the GFoLDS model introduced and pretrained in Chapter 4 on a series of downstream tasks, in order to demonstrate the utility of this model: to the best of my knowledge, the experiments conducted in Chapter 5 represent the first time that a language model pretrained solely over logical forms has been evaluated on a wide range of downstream tasks. In order to evaluate against a comparable superficial model, I pretrain two BERT comparison models (the *base* and *large* architectural variants) on the same dataset as GFoLDS. I then show that the GFoLDS model vastly outperforms the BERT comparison models—and remains competitive with the original BERT models, which were pretrained on far more data—on all four evaluation tasks, thereby providing evidence in support of the Accelerated Learning Hypothesis and the use of language models over logical forms as more data-economical LMs.

Chapter 6 is dedicated to an in-depth analysis of GFoLDS. In order to motivate continued research into language models over logical forms, I investigate the scalability of GFoLDS and show that this model is likely to scale in terms of pretraining data and parameter count, indicating that a larger model trained on more data would continue to improve on downstream tasks. I then present experimental results that directly support the validity of the Accelerated Learning Hypothesis. Finally, in order to inform future directions in the development of language models over logical forms, I use the experimental framework developed in Chapter 2 to probe the GFoLDS model: I identify several limitations of this model, all of which I trace back to a single weakness in the graph-positional encoding component of the model's architecture.

Chapter 7 outlines future directions for GFoLDS and the research program of language modeling over logical forms in general. I first discuss a potential application of the GFoLDS model introduced in Chapter 4 that was left unexplored in this dissertation: namely, its use with lower-resource languages. I then suggest improvements to GFoLDS that are intended to address limitations uncovered in Chapters 4-6. Chapter 7 concludes with a proposal for what

I envision to be the next major phase in the research program of language modeling over logical forms: graph-to-graph generative LMs.

The conclusion (Chapter 8) of this dissertation summarizes its findings, contributions, and limitations.

Chapter 2

It is not True that (Superficial) Transformers are Inductive Learners[1]

In this chapter, I introduce a probing task designed to investigate language models' (LMs') ability to learn to reason logically and evaluate a range of (near-)state-of-the-art (SoTA) superficial LMs on this task. Specifically, I investigate near-SoTA transformer (Vaswani et al., 2017) natural language inference (NLI) models' ability to inductively learn the law of the excluded middle (LEM) in the context of *external negation*: negation that occurs externally to the proposition that is negated, e.g. *"it is not true that apples are red"*. I argue that a model's ability to comprehend negation represents an effective proxy for its general logical reasoning capabilities, due to the critical role that negation plays in logical reasoning: for example, $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ both form functionally complete sets—any possible boolean operator can be expressed through an expression consisting of only negation and conjunction (or negation and disjunction). The use of *external* negation facilitates the data generation procedure, permitting the automatic construction of challenge examples from NLI datasets that modify the original examples' class labels in a predictable manner (see Section 2.4.1.1).

In these experiments, I demonstrate that models fine-tuned on NLI datasets learn to

[1]A modified version of this chapter has been published in Sullivan (2024).

treat external negation as a distractor—effectively ignoring its presence in hypothesis sentences—and that several near-SoTA encoder and encoder-decoder transformer models fail to inductively learn the law of the excluded middle for a single external negation prefix with respect to NLI tasks (Section 2.4). I then show that those models which are able to learn the law of the excluded middle for a single prefix are unable to generalize this pattern to similar prefixes (Section 2.5), and that this inability to generalize is due catastrophic forgetting of the similarity between highly similar external negation prefixes (Section 2.6).

The experimental results contained in this chapter indicate that transformer models do not learn to reason logically when fine-tuned on NLI datasets, lending further support to existing hypotheses in the literature (see e.g. McCoy, Pavlick, and Linzen, 2019) that they are instead learning to leverage shallow heuristics. In Section 2.3, I present evidence (Theorem 1) that this failure of superficial transformer models to inductively learn LEM arises from deficiencies in their training procedure and/or the structure (or lack thereof) of their input data, rather than flaws inherent to transformer architectures themselves (see the discussion in Section 2.7).

The primary objective of this chapter is to construct an evaluation task that allows for the in-depth analysis of the reasoning abilities of LMs: I aim to motivate the language model over logical forms introduced in Chapter 4, using this task to demonstrate that there exist relatively simple logical-reasoning tasks that continue to confound even near-SoTA superficial LMs. Although—as with the superficial LMs—the model that I introduce in Chapter 4 fails to accomplish the double-negation cancellation task (see Chapter 6), the detailed analysis permitted by the experiments presented in the present chapter allows us to contrast the reasons for its failure with those of superficial LMs, and to shed light on the significance of these models' weaknesses with respect to their logical reasoning capacity.

2.1 Background: NLI

NLI tasks require detecting inferential relations between pairs of sentences (Fyodorov, Winter, and Francez, 2000). For NLI datasets such as MultiNLI (MNLI; Williams, Nangia, and Bowman, 2017) and Stanford NLI (SNLI; Bowman et al., 2015), the task proceeds as follows: given a pair of sentences (P, H), an NLI model must determine whether the *premise* P *entails* the *hypothesis* H, H *contradicts* P, or P and H are *neutral* with respect to one another (i.e. P does not entail H and H does not contradict P).

These tasks require logical reasoning capabilities that extend beyond basic linguistic competence (Richardson et al., 2020). For example, understanding that *"Jane is travelling to Algeria"* entails *"Jane is travelling to Africa"* requires mereological world knowledge (Hovda, 2009): an agent must know that Algeria is contained within Africa. To understand that *"Jane is traveling to Algeria"* does *not* entail *"Jane is traveling to Algiers"*, the agent must understand that Algiers is contained within Algeria, but that Algeria is not solely comprised of the city of Algiers.

Because of the considerable amount of reasoning that is required to accomplish NLI tasks, it is important to scrutinize the degree to which current NLI models are *actually* learning to reason logically. McCoy, Pavlick, and Linzen's (2019) findings suggest, for example, that even (then-)SoTA NLI models such as BERT (Devlin et al., 2019) adopt shallow, textual heuristics to achieve high-scoring results on the MNLI dataset, although the MNLI dataset itself is likely to be—at least partially—at fault, as discussed in Section 2.2.

2.2 Related Work

There is a large body of existing work on probing NLI models to gain insight into their reasoning abilities (Belinkov and Glass, 2019). As mentioned in Section 2.1, McCoy, Pavlick, and Linzen (2019) find that language models fine-tuned on MNLI learn to leverage shallow heurisics to achieve exceptionally high accuracy on this dataset. Similarly, Chien and Kalita

(2020) and Richardson et al. (2020) probe NLI models' performance with respect to specific syntactic and semantic phenomena (e.g. coordination, quantification, monotonicity, etc.). They find that SoTA models fine-tuned on MNLI and SNLI perform poorly on challenge examples generated to evaluate the models with respect to these phenomena, but can be easily fine-tuned to master the challenge data, while retaining their high performance on the original datasets.

2.2.1 Inoculation by Fine-Tuning

In all three of the aforementioned papers, their respective authors utilize the method of *inoculation by fine-tuning*. Liu, Schwartz, and Smith (2019) introduce this paradigm as a technique for differentiating between deficiencies in a model's training data and deficiencies in the model itself. Inoculation by fine-tuning assumes that there is an *original* dataset that is divided into train/test splits and a smaller *challenge* dataset (also divided into train and test splits), and that the model's performance on the challenge dataset is significantly lower than on the original dataset. The idea is to fine-tune the model on the challenge dataset until validation performance on the *original* test set has not improved for five epochs, then measure the newly fine-tuned (*inoculated*) model on the challenge test set. If the inoculated model maintains its performance on the original test set and performs (nearly) as well on the challenge test set, this suggests that the model's poor performance on the challenge data was due to flaws (e.g. a lack of diversity) in the original training data. Conversely, if the model's performance on the challenge test set remains significantly worse than on the original data after inoculation, this suggests that its poor performance on the challenge data is due to a deficiency in the model itself.

The experiments in this chapter probe various NLI models' logical reasoning abilities with respect to external negation, using automatically-generated challenge data along with the inoculation by fine-tuning paradigm. Unlike the closely-related notion of *adversarial attacks*, which seek to perturb input examples without altering their class labels, the external

negation prefixes used to generate challenge examples in Experiments 1-3 (Sections 2.4, 2.5, 2.6) do alter the original examples' class labels, albeit in a predictable manner. This is similar to the challenge data that Niven and Kao (2019) construct from the Argument Reasoning Comprehension Task (Habernal et al., 2018); these authors find that BERT *cannot* be inoculated against such challenge data, and conclude that (superficial) transformer models' inability to ground text to real-world concepts presents an insurmountable barrier to their logical-reasoning abilities.

2.2.2 Probing LMs with Negation

Naik et al. (2018) conduct "stress tests" on NLI models by concatenating logical distractor strings such as *"and false is not true"* to the input examples, and find that such distractors drastically reduce SoTA NLI models' performance on these tasks. While these authors investigate NLI models' performance with respect to logical reasoning, their experiments regarding negation are limited to negation items appearing in these distractor terms, rather than negating the original hypothesis sentence itself. On the other hand, Hossain et al. (2020) probe NLI models and datasets by negating the original premise and/or hypothesis sentence(s) using an automatic dependency parser; these automatically-generated challenge examples are then checked for accuracy and re-assigned class labels by human annotators. These authors find that models fine-tuned on the original NLI datasets perform poorly on development sets consisting of these negation-augmented examples, and that while fine-tuning on the challenge data improves performance on negation-augmented test splits derived from SNLI, fine-tuning does *not* significantly increase model accuracy on MNLI-derived examples. Note that, unlike the present work, Hossain et al. (2020) do not study repeated/embedded negation or double-negation cancellation.

Yuan et al. (2023) examine pretrained language models' (PLMs) *deductive* reasoning abilities via cloze tests. These authors find that PLMs are unable to fully generalize rules of logical deduction to arbitrary contexts. Furthermore, they observe that these

models struggle to differentiate between positive statements and their negated counterparts, in line with a wide body of recent literature suggesting that transformers have difficulty processing and comprehending negation (e.g. Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023; Laverghetta Jr. et al., 2021; Rogers, Kovaleva, and Rumshisky, 2020; Ettinger, 2020; Laverghetta Jr. and Licato, 2022; Kassner and Schütze, 2020). They find that while inoculating PLMs for deductive reasoning tasks improves performance, it results in catastrophic forgetting of previous knowledge; likewise, in Sections 2.5 and 2.6 of this chapter, I find that inoculating pretrained NLI models against challenge data augmented with external negation prefixes causes catastrophic forgetting of prior knowledge of their similarity to related prefixes.

Jang, Kwon, and Lukasiewicz (2022) evaluate the consistency of language models across various axes. Of particular interest to the current discussion is their analysis of *negational consistency*: the degree to which a given language model's predictions differ between texts having opposite meanings. These authors find that negational consistency remains low across a variety of models and tasks—in particular, RoBERTa (Liu et al., 2019) and BART (Lewis et al., 2020) exhibit low negational consistency on the SNLI dataset.

In an experiment highly related to the present work, Laverghetta Jr. and Licato (2022) probe NLI models' performance with respect to negation, and find that the models struggle with certain types of negation more so than others. In line with the results we observe in Section 2.4, they find that the models have difficulty inoculating against those problematic negation categories. Unlike the experiments in this chapter, Laverghetta Jr. and Licato (2022) do not construct challenge examples involving negation, but rather use examples drawn from NLI datasets that already contain negation.

Unique to the work conducted in this chapter is the evaluation of transformers' ability to learn the law of the excluded middle (LEM) and our finding that, while many cannot learn this pattern, a few transformer NLI models are in fact able to inductively learn LEM for a single external negation prefix. Additionally, the results of Experiments 2 and 3 (Sections

2.5 and 2.6), extend Yuan et al.'s (2023) results regarding catastrophic forgetting resulting from inoculation in the context of deductive reasoning tasks, to double negation-cancellation in the setting of NLI tasks. Finally, Theorem 1 (see Section 2.3) is the first known proof that there exists (at least, in principle) an encoder transformer capable of modeling LEM for arbitrary-length sequences of any combination of external negation prefixes with respect to any NLI dataset. This theorem sheds further light on evidence in the literature (Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023; Laverghetta Jr. et al., 2021; Rogers, Kovaleva, and Rumshisky, 2020; Ettinger, 2020; Laverghetta Jr. and Licato, 2022; Kassner and Schütze, 2020, etc.) indicating that transformers struggle to model negation, and suggests that this observed failure is not due to an inherent flaw in transformer architectures themselves, but instead may be due to deficiencies in their training procedure and/or the structure of their input data (see the discussion in Section 2.7).

2.3 Can Transformers Model LEM?

Before evaluating NLI models' ability to inductively *learn* the law of the excluded middle (LEM), I first establish whether—learnability aside—it is theoretically possible for transformer architectures to model LEM at all: Theorem 1 proves that (encoder) transformer architectures are in fact capable of modeling LEM with respect to NLI tasks for arbitrary-length sequences of any combination of external negation prefixes. Note that—with the exception of the BART models—the NLI datasets, transformer models, and set of external negation prefixes used in Experiments 1-3 satisfy the assumptions of Theorem 1.

Theorem 1. *Let $D = \{(P_i, H_i, L_i)\}_{i \in I}$ be a finite-cardinality NLI dataset, and for any NLI model M, let $Acc(M, D)$ denote the classification accuracy of M on D. Let Σ' be a finite alphabet such that $D \subset (\Sigma')^* \times (\Sigma')^* \times \Lambda$ (where $\Lambda = \{\mathcal{E}, \mathcal{N}, \mathcal{C}\}$ denotes the set of labels). Let $N \subset (\Sigma')^*$ be any finite-cardinality set of external-negation prefixes such that no prefix is a*

substring of one or more other prefixes[2].

Then there exists an alphabet $\Sigma \supset \Sigma'$ *and an injective* $f\colon (\Sigma')^* \to \Sigma^*$ *such that for any fixed (finite)* $w > max_{i \in I}|P_i H_i|$ *and any fixed-precision transformer encoder (with an NLI classification head)* T*, there exists a fixed-precision transformer encoder* T' *such that* T' *matches the accuracy of* T *on* D *and on any dataset* D' *formed by prefixing any* $\eta \in N^*$ *to each hypothesis sentence in* D[3].

Proof. Section 2.9. □

The proof of Theorem 1 relies on a function f that re-structures the input data; the transformer NLI models evaluated in Experiments 1-3 (Sections 2.4, 2.5, 2.6) are obviously not equipped with such a function, and the ability of transformers to model LEM with respect to unstructured plain text is not established in Theorem 1. Furthermore, Theorem 1 merely states that there exists an encoder transformer capable of modeling LEM for external negation with respect to NLI tasks, and makes no claim regarding its architectural configuration (i.e. layer size, floating-point precision, etc.). It is unclear whether the transformer models evaluated in this chapter have the specific architecture required to accomplish this task.

Critically, the proof of Theorem 1 does not make any claims regarding the (inductive) learnability of LEM; while it is theoretically possible to *model* LEM with an encoder transformer, these language models' ability to *inductively learn* LEM remains uncertain from the conclusions of Theorem 1 alone. In the following section (Experiment 1), we observe experimental evidence demonstrating that some transformer NLI models (in particular, RoBERTa) are able to learn LEM for a single external negation prefix.

[2]Formally: for all $\eta \in N$, $\eta', \eta'' \in (N - \{\eta\})^*$, there does not exist i, j such that $\eta = \eta'_{i:} \mid\mid \eta''_{:j}$

[3]Formally: $Acc(T', f(D)) = Acc(T, D)$, and for any $\eta \in N^*$ such that $max_{i \in I}|P_i \eta H_i| \leq w$: $Acc(T', \{f(P_i \eta H_i)\}_{i \in I}) = Acc(T, D)$

2.4 Experiment 1

Experiment 1 probes six different transformer NLI models' ability to inductively learn the law of the excluded middle (LEM) with respect to external negation. The DeBERTa (He et al., 2021) model, denoted $DeBERTa_S$[4], is DeBERTa-large fine-tuned on SNLI. The first BART model, denoted $BART_M$[5], is BART-large fine-tuned on MNLI, while the second, $BART_{SMFA}$[6], is BART-large fine-tuned on MNLI, SNLI, FEVER (Thorne et al., 2018), and ANLI (Nie et al., 2020). The first RoBERTa model, $RoBERTa_M$[7], is RoBERTa-large fine-tuned on MNLI, and the second, $RoBERTa_S$[8], is RoBERTa-large fine-tuned on SNLI, while the third, $RoBERTa_{SMFA}$[9], is RoBERTa-large fine-tuned on SNLI, MNLI, FEVER, and ANLI.

2.4.1 Experimental Setup

For each $1 \leq n \leq 5$ and each NLI dataset $D \in \{\text{MNLI}, \text{SNLI}\}$, let $D^{\leq n}$ denote the *depth-$\leq n$ challenge set* (consisting of train and development splits). $D^{\leq n}$ is generated from examples randomly drawn from the original dataset's train splits: each $\text{MNLI}^{\leq n}$ consists of 4,906 entailment, neutral, and contradiction examples (14,718 total; 9,813 train/4,905 development), and each $\text{SNLI}^{\leq n}$ consists of 14,997 examples (4,999 per class; 9,999 train/4,998 development).

2.4.1.1 Challenge Data Generation

Challenge sets. For each $1 \leq k \leq n$, $1/n^{th}$ of the examples in each class in $D^{\leq n}$ are depth-k negated by prepending the *trigger* prefix T_{NT} = *"it is not true that"* to the original hypothesis sentence k times (i.e. by converting (P, H) to $(P, (T_{NT})^k H)$; see Table 2.1). For

[4]https://huggingface.co/pepa/deberta-v3-large-snli
[5]https://huggingface.co/facebook/bart-large-mnli
[6]https://huggingface.co/ynie/bart-large-snli_mnli_fever_anli_R1_R2_R3-nli
[7]https://huggingface.co/roberta-large-mnli
[8]https://huggingface.co/pepa/roberta-large-snli
[9]https://huggingface.co/ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli

example, in $D^{\leq 5}$, $1/5^{th}$ of the examples in each class are depth-5 negated, $1/5^{th}$ are depth-4 negated, $1/5^{th}$ are depth-3 negated, etc. One may object that concatenating T_{NT} five times (for example) in front of the original hypothesis does not a result in a particularly natural sentence, and that a model is highly unlikely to encounter such a sentence in real-world text data. Regardless of its naturality, however, this pattern is fairly trivial (for a human) to learn: given a challenge example $(P, (T_{NT})^k H)$—derived from an original example (P, H)—simply count the number k of occurrences of T_{NT}.

The class label remains the same if k is even or the class label is *neutral*, and *contradiction* flips to *entailment* (and vice-versa) if k is odd: given a premise, hypothesis, label triple (P, H, L) in an NLI dataset, the label L is defined as in Equation 2.1, where $\mathcal{E}$, $\mathcal{C}$, and $\mathcal{N}$ denote *entailment*, *contradiction*, and *neutral*, respectively.

$$L = \mathcal{E} \Leftrightarrow P \rightarrow H \tag{2.1a}$$

$$L = \mathcal{C} \Leftrightarrow P \rightarrow \neg H \tag{2.1b}$$

$$L = \mathcal{N} \Leftrightarrow (L \neq \mathcal{E} \wedge L \neq \mathcal{C}) \tag{2.1c}$$

Where the left-hand sides of the bidirectional arrows in Equation 2.1a-b hold in every logically possible state of affairs. Now consider the pair (P, H'), where $H' = \neg H$. The label L' of the modified example (P, H') is a function of the label L of the original example (P, H): if $L = \mathcal{E}$, then we have the following logical equivalences in Equation 2.2.

$$(P \rightarrow H) = (P \rightarrow \neg\neg H) = (P \rightarrow \neg H') \Leftrightarrow (L' = \mathcal{C}) \tag{2.2}$$

Where the above equivalences follow from the law of the excluded middle, the definition of H', and Equation 2.1b (respectively). Therefore, $L = \mathcal{E} \leftrightarrow L' = \mathcal{C}$. Similarly, if $L = \mathcal{C}$, then we have the following logical equivalences in Equation 2.3 (by the definition of H' and Equation 2.1a, respectively).

$$(P \to \neg H) = (P \to H') \Leftrightarrow (L' = \mathcal{E}) \tag{2.3}$$

By Equation 2.3, $L = \mathcal{C} \leftrightarrow L' = \mathcal{E}$.

Finally, suppose that $L = \mathcal{N}$. By the above discussion (Equations 2.2 and 2.3), we have $L = \mathcal{E} \leftrightarrow L' = \mathcal{C}$ and $L = \mathcal{C} \leftrightarrow L' = \mathcal{E}$. Therefore $L' \notin \{\mathcal{E}, \mathcal{C}\}$ and so (by Equation 2.1c) $L' = \mathcal{N}$. So we have $L = \mathcal{N} \to L' = \mathcal{N}$. Swapping L and L' in the above discussion in this paragraph, we have $L' = \mathcal{N} \to L = \mathcal{N}$. Therefore, $L = \mathcal{N} \leftrightarrow L' = \mathcal{N}$.

Test sets. Now, for all $m > 1$ and each NLI dataset $D \in \{\text{MNLI}, \text{SNLI}\}$, let D^m_{NT} denote the *depth-m test set*. D^m_{NT} is the size of the development split of $D^{\leq m}$, and the procedure for generating D^m_{NT} is nearly identical to that of $D^{\leq m}$, with the exception that D^m_{NT} consists *only* of depth-m externally-negated examples.

Note that the two datasets (MNLI and SNLI) contain many examples that are not complete sentences—but rather sentence fragments—in which case the external negation prefix T_{NT} = *"it is not true that"* is grammatically nonsensical. To account for this, the pool of possible examples to be included into the challenge and test datasets consists only of those in which the hypothesis H is a complete sentence. If the first word in H is (part of) a named entity—as determined by SpaCy's *EntityRecognizer*[10] named entity recognition pipeline—then the augmented (i.e. challenge) hypothesis is set to $(T_{NT})^n H$. If the first word in H does *not* belong to a named entity, then the augmented hypothesis is $(T_{NT})^n H_0$, where H_0 is formed from H by lower-casing the first character. This is to control for potential confounding factors due to irregular capitalization.

2.4.1.2 Inoculation and Evaluation

For all $1 \leq n \leq 5$, I inoculated each NLI model against the challenge set(s) $D^{\leq n}$. Following the paradigm of inoculation by fine-tuning, the models were fine-tuned on the train split

[10] https://spacy.io/api/entityrecognizer

Premise	**Template**	**Hypothesis**	**Label**
A young boy dressed in plaid about to take a picture.	(P, H)	A young boy is about to take a picture.	Entailment
	$(P, (T_{NT})^1 H)$	It is not true that a young boy is about to take a picture.	Contradiction
	$(P, (T_{NT})^2 H)$	It is not true that it is not true that a young boy is about to take a picture.	Entailment
	$(P, (T_{NT})^3 H)$	It is not true that it is not true that it is not true that a young boy is about to take a picture.	Contradiction
A race between friends at the park.	(P, H)	The park is deserted.	Contradiction
	$(P, (T_{NT})^1 H)$	It is not true that the park is deserted.	Entailment
	$(P, (T_{NT})^2 H)$	It is not true that it is not true that the park is deserted.	Contradiction
	$(P, (T_{NT})^3 H)$	It is not true that it is not true that it is not true that the park is deserted.	Entailment
People kneeling on the ground.	(P, H)	People are praying.	Neutral
	$(P, (T_{NT})^1 H)$	It is not true that people are praying.	Neutral

Table 2.1: Examples of depth-n negated challenge data points $(P, (T_{NT})^n H)$ generated from SNLI (some examples have been slightly modified for presentability).

of $D^{\leq n}$, and validated at each epoch on the *original* NLI dataset's development split, with early-stopping if validation performance did not improve after five epochs. Once inoculated on the depth-$\leq n$ external negation data ($D^{\leq n}$), I evaluated the models on D^{m}_{NT} for multiple values of $m > n$. This was intended to measure the degree to which the models are able to generalize LEM beyond the number of external negation prefixes seen during inoculation.

I evaluated and inoculated each model on the challenge dataset(s) generated from the dataset(s) that the model was originally fine-tuned on: $BART_M$ and $RoBERTa_M$ were evaluated on MNLI-derived examples, $RoBERTa_S$ and $DeBERTa_S$ on SNLI-derived examples, and $BART_{SMFA}$ and $RoBERTa_{SMFA}$ on examples derived from both datasets. All models were fine-tuned with a batch size of 64 at a learning rate of 10^{-5} using the Adam (Kingma and Ba, 2014) optimizer.

2.4.2 Results

Model original/challenge development set accuracies pre- and post-inoculation are located in Table 2.2. Most models were able to inoculate against the depth-$\leq n$ external negation data for all $1 \leq n \leq 5$: they retained their high-performing accuracy on the original development sets, and performed as well (or nearly so) on the challenge development sets after inoculation. The notable exceptions were $BART_M$ and $BART_{SMFA}$, which struggled to inoculate for $n \in \{1, 4, 5\}$ and $n \in \{3, 4\}$, respectively—recall that BART is the only model architecture evaluated in this experiment that does not satisfy the assumptions of Theorem 1.

Model	Depth	Initial (Original)	Initial (Challenge)	Inoculated (Original)	Inoculated (Challenge)
$BART_M$	1	0.89	0.52	0.77	0.94
$RoBERTa_M$	1	0.89	0.51	0.87	0.93
$DeBERTa_S$	1	0.9	0.39	0.9	0.91
$RoBERTa_S$	1	0.88	0.57	0.88	0.89
$BART_{SMFA}$	1	0.89	0.69	0.87	0.92
$RoBERTa_{SMFA}$	1	0.87	0.51	0.86	0.91
$BART_M$	2	0.89	0.61	0.86	0.94
$RoBERTa_M$	2	0.89	0.63	0.87	0.97
$DeBERTa_S$	2	0.9	0.48	0.9	0.96
$RoBERTa_S$	2	0.88	0.66	0.88	0.94
$BART_{SMFA}$	2	0.89	0.72	0.88	0.95
$RoBERTa_{SMFA}$	2	0.87	0.65	0.88	0.95
$BART_M$	3	0.89	0.53	0.87	0.95
$RoBERTa_M$	3	0.89	0.54	0.87	0.96
$DeBERTa_S$	3	0.9	0.45	0.9	0.96
$RoBERTa_S$	3	0.88	0.57	0.88	0.93
$BART_{SMFA}$	3	0.89	0.6	0.76	0.93
$RoBERTa_{SMFA}$	3	0.87	0.54	0.88	0.94
$BART_M$	4	0.89	0.61	0.62	0.75
$RoBERTa_M$	4	0.89	0.62	0.74	0.88
$DeBERTa_S$	4	0.9	0.54	0.89	0.76
$RoBERTa_S$	4	0.88	0.64	0.89	0.89
$BART_{SMFA}$	4	0.89	0.66	0.62	0.86
$RoBERTa_{SMFA}$	4	0.87	0.61	0.88	0.89
$BART_M$	5	0.89	0.55	0.32	0.74
$RoBERTa_M$	5	0.89	0.56	0.88	0.93
$DeBERTa_S$	5	0.9	0.5	0.9	0.91
$RoBERTa_S$	5	0.88	0.58	0.88	0.89
$BART_{SMFA}$	5	0.89	0.59	0.87	0.88
$RoBERTa_{SMFA}$	5	0.87	0.54	0.86	0.87

Table 2.2: Model accuracy on the original and challenge development sets before (*initial*) and after (*inoculated*) depth-$\leq n$ inoculation ($1 \leq n \leq 5$).

Model	Depth-m test	No inoc.	Depth-1 inoc.	Depth-$\leq$2 inoc.	Depth-$\leq$3 inoc.
$BART_M$	2	0.71	0.32	—	—
$BART_M$	3	0.36	0.93	0.31	—
$BART_M$	4	0.82	0.36	0.94	0.31
$BART_M$	5	0.33	0.88	0.31	0.94
$BART_M$	6	0.86	0.41	0.94	0.31
$RoBERTa_M$	2	0.77	0.36	—	—
$RoBERTa_M$	3	0.34	0.89	0.33	—
$RoBERTa_M$	4	0.85	0.33	0.97	0.32
$RoBERTa_M$	5	0.32	0.88	0.33	0.95
$RoBERTa_M$	6	0.89	0.34	0.97	0.33
$DeBERTa_S$	2	0.56	0.62	—	—
$DeBERTa_S$	3	0.4	0.61	0.32	—
$DeBERTa_S$	4	0.84	0.64	0.96	0.5
$DeBERTa_S$	5	0.3	0.51	0.32	0.96
$DeBERTa_S$	6	0.88	0.77	0.96	0.36
$RoBERTa_S$	2	0.74	0.32	—	—
$RoBERTa_S$	3	0.4	0.89	0.3	—
$RoBERTa_S$	4	0.84	0.35	0.94	0.39
$RoBERTa_S$	5	0.34	0.88	0.3	0.74
$RoBERTa_S$	6	0.83	0.33	0.93	0.53
$BART_{SMFA}$	2	0.77	0.37	—	—
$BART_{SMFA}$	3	0.33	0.91	0.31	—
$BART_{SMFA}$	4	0.84	0.34	0.94	0.29
$BART_{SMFA}$	5	0.3	0.85	0.31	0.92
$BART_{SMFA}$	6	0.86	0.41	0.94	0.28
$RoBERTa_{SMFA}$	2	0.79	0.35	—	—
$RoBERTa_{SMFA}$	3	0.32	0.93	0.32	—
$RoBERTa_{SMFA}$	4	0.83	0.31	0.95	0.32
$RoBERTa_{SMFA}$	5	0.32	0.94	0.32	0.94
$RoBERTa_{SMFA}$	6	0.84	0.32	0.95	0.32
Mean	2	0.72	0.39	—	—
Mean	3	0.36	0.86	0.32	—
Mean	4	0.84	0.39	0.95	0.35
Mean	5	0.32	0.82	0.32	0.91
Mean	6	0.86	0.43	0.95	0.35

Table 2.3: Accuracy for all models on depth-(m>n) external negation (D^m_{NT}) after depth-$\leq n$ inoculation ($n \in \{1, 2, 3\}$) on $D^{\leq n}$. For the sake of convenience, mean accuracy across the models is reported at the bottom of the table; most individual model accuracies do not substantially deviate from these mean values.

Model	Depth-m test	No inoc.	Depth-$\leq$4 inoc.
$BART_M$	5	0.33	0.34
$RoBERTa_M$	5	0.32	0.34
$DeBERTa_S$	5	0.30	0.33
$RoBERTa_S$	5	**0.34**	**0.89**
$BART_{SMFA}$	5	0.30	0.32
$RoBERTa_{SMFA}$	5	**0.32**	**0.79**
$BART_M$	6	0.86	0.93
$RoBERTa_M$	6	0.89	0.95
$DeBERTa_S$	6	0.88	0.95
$RoBERTa_S$	6	**0.83**	**0.93**
$BART_{SMFA}$	6	0.86	0.93
$RoBERTa_{SMFA}$	6	**0.84**	**0.95**

Table 2.4: Accuracy for all models on depth-m external negation after depth-$\leq$4 inoculation ($m \in \{5, 6\}$).

In spite of their ability to inoculate against depth-$\leq n$ challenge data, the models struggled to generalize this knowledge to depth-m negation for values of $m > n$. Table 2.3 reports model accuracy on depth-m>n external negation after depth-$\leq n$ inoculation for $1 \leq n \leq 3$, $2 \leq m \leq 6$. A clear pattern emerges in this table: before any inoculation, we observe high model accuracy ($\sim$80%) on the depth-m negtion data for even values of m, and near-random-chance accuracy ($\sim$34%) for odd values of m. This indicates that, before inoculation, the models were essentially entirely ignoring the external negation prefixes and treating them as distractors; depth-m negation does not alter the class label for even values of m, and so a model treating the prefix as a distractor will retain high accuracy on those examples, purely by chance. To reiterate: these models—ostensibly fine-tuned on a logical-reasoning task—*have learned to entirely ignore external negation* when predicting inferential relations.

Furthermore, when inoculated against depth-1 external negation, the pattern reverses: we note near-random-chance accuracy for *even* values of m, and high accuracy for *odd* values of m. After depth-1 inoculation, the models have learned to treat *any* depth-m external negation prefix as equivalent to a depth-1 (i.e. single) prefix.

Interestingly, after depth-$\leq$2 inoculation, the models revert to the original pattern of high

Model	Depth-m test	No inoc.	Depth-$\leq$5 inoc.
$BART_M$	6	0.86	0.34
$RoBERTa_M$	6	**0.89**	**0.91**
$DeBERTa_S$	6	0.88	0.31
$RoBERTa_S$	6	**0.83**	**0.94**
$BART_{SMFA}$	6	0.86	0.30
$RoBERTa_{SMFA}$	6	**0.84**	**0.95**
$BART_M$	7	0.32	0.93
$RoBERTa_M$	7	**0.32**	**0.96**
$DeBERTa_S$	7	0.28	0.95
$RoBERTa_S$	7	**0.36**	**0.94**
$BART_{SMFA}$	7	0.29	0.92
$RoBERTa_{SMFA}$	7	**0.31**	**0.95**

Table 2.5: Accuracy for all models on depth-m external negation after depth-$\leq$5 inoculation ($m \in \{6, 7\}$).

accuracy for even values of m, and poor performance for odd values. Despite training on both depth-1 and depth-2 external negation, the models merely memorize the effect of depth-1 negation on class labels, and do not generalize to odd values of $m > 1$. A similar pattern emerges after depth-$\leq$3 inoculation: after fine-tuning on depth-1, depth-2, and depth-3 external negation, the models memorize the effect (or lack thereof) of depth-2 negation on class labels, and do not generalize to even values of $m > 2$.

However, Table 2.4 indicates that, after depth-$\leq$4 inoculation, the $RoBERTa_S$ and $RoBERTa_{SMFA}$ models do in fact inductively learn to repeatedly cancel double negation for values of $m > 4$. After depth-$\leq$5 inoculation, $RoBERTa_M$ also learns the desired pattern (see Table 2.5): all three RoBERTa models have inductively learned LEM for arbitrary values of m.

Given all six models' difficulty with inoculation against depth-m external negation for arbitrary values of m, it is reasonable to question the RoBERTa models' ability to generalize the negation-cancellation patterns that they have learned after depth-$\leq$5 inoculation to external negation strings beyond the trigger T_{NT} = *"it is not true that"* that they saw during inoculation. The following experiment (Section 2.5) evaluates the three RoBERTa models'

ability to repeatedly cancel double negation with respect to the prefix *"it is false that"*, after inoculation against depth-$\leq$5 *"it is not true that"* prefixes ($D^{\leq 5}$).

2.5 Experiment 2

This experiment restricts its analysis to the three RoBERTa models, as they were the only models of the six evaluated in Experiment 1 (Section 2.4) that were able to fully generalize depth-m negation-cancellation to arbitrary values of $m > 5$.

2.5.1 Experimental Setup

For all $m \geq 1$ and each NLI dataset $D \in \{\text{MNLI}, \text{SNLI}\}$, let D_F^m denote the depth-m challenge test set. Each D_F^m was created in an identical manner to the depth-m challenge test sets D_{NT}^m defined in Section 2.4.1 above: D_F^m consists only of examples drawn from the dataset's original development split that are modified to have depth-m externally-negated hypothesis sentences with an equal number of examples per class label, and $|D_F^m| = |D_{NT}^m|$.

However, in place of the trigger T_{NT} = *"it is not true that"* used to construct D_{NT}^m, in this experiment D_F^m was generated using the trigger T_F = *"it is false that"*. These two triggers are effectively semantically equivalent; the phrase *"not true"* has simply been replaced with the (virtually) synonymous *"false"*. Assuming that the models have truly learned the law of the excluded middle (LEM), we should expect to see similar performance on D_F^m to that of D_{NT}^m.

After inoculation on the depth-$\leq$5 T_{NT} external negation data, I evaluated each of the three RoBERTa models ($RoBERTa_S$, $RoBERTa_M$, $RoBERTa_{SMFA}$) on D_F^m for all $1 \leq m \leq 8$. As in the procedure for Experiment 1 (see Section 2.4.1), each model was evaluated on the challenge dataset(s) generated from the dataset(s) that the model was originally fine-tuned on.

2.5.2 Results

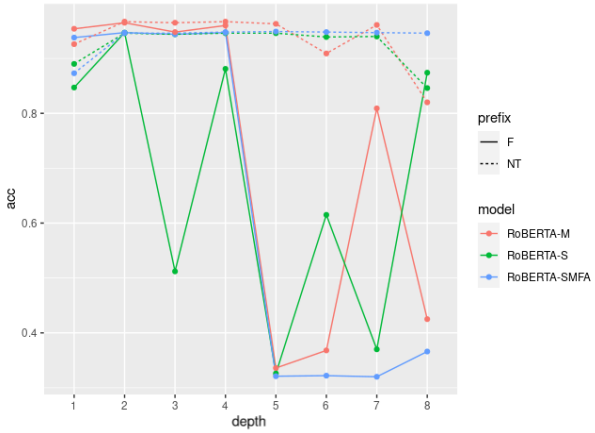


Figure 2.1: Accuracy for the depth-$\leq$5 T_{NT}-inoculated RoBERTa models on depth-m externally-negated examples with T_{NT} (dashed) and T_F (solid).

Figure 2.1 shows the results of this experiment: $RoBERTa_S$ failed to generalize LEM from T_{NT} to T_F for values of $m > 2$, while $RoBERTa_M$ and $RoBERTa_{SMFA}$ experience precipitous decreases in accuracy at $m = 5$ (and erratic accuracy thereafter). While these models are able to generalize external negation-cancellation to arbitrary-length repeated T_{NT} prefixes, they clearly cannot extend this pattern to near-synonymous prefixes.

One may object that the models have failed to learn the pattern for T_F because they did not see it during inoculation. This objection may be valid, but belies the critical point: *these models have failed to generalize LEM* from T_{NT} to T_F. While the models very well may learn to cancel external negation prefixes after fine-tuning on all possible sequences of this type (see the discussion in Section 2.7), at that point they are not learning the general function of negation, but rather memorizing it for each possible surface realization.

Given the conclusions of Theorem 1 and the RoBERTa models' ability to generalize double-negation cancellation for T_{NT} as observed in Experiment 1 (Section 2.4), the results of Experiment 2 beg the question as to *why* the RoBERTa models cannot fully generalize LEM from T_{NT} to T_F. In the following experiment (Section 2.6), I examine the embeddings generated by the RoBERTa models pre- and post-inoculation, shedding light on the root of

their failure to learn to generalize LEM to unseen prefixes.

2.6 Experiment 3

As in Experiment 2 (Section 2.5), this experiment restricts its analysis to the three RoBERTa models.

2.6.1 Experimental Setup

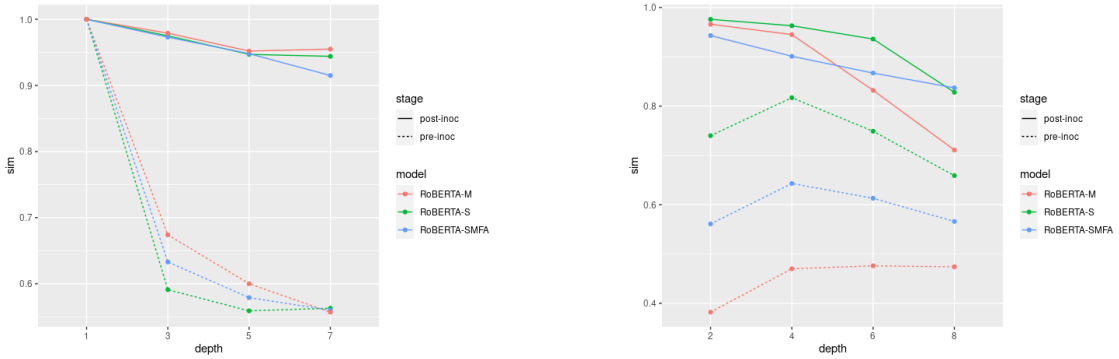
As mentioned above, this experiment probes the embeddings that these models generate before and after depth-$\leq$5 T_{NT} inoculation. For each dataset $D \in \{\text{MNLI}, \text{SNLI}\}$, I took a subset D' of the original development set, containing $\sim$50-100 examples of each class, depending on the size of the dataset. Then, for each $1 \leq m \leq 8$ and each $(P_i, H_i) \in D'$, I computed the cosine similarity between the mean-pooled embeddings of $(T_{NT})^m H_i$ and $(T_F)^m H_i$.

For even values of m, I additionally computed the cosine similarity between $(T_{NT})^m H_i$ and $(T_F)^2 H_i$; $(T_{NT})^2 H_i$ and $(T_F)^m H_i$; $(T_{NT})^m H_i$ and $(T_{NT})^2 H_i$; $(T_F)^m H_i$ and $(T_F)^2 H_i$; $(T_{NT})^m H_i$ and H_i; and $(T_F)^m H_i$ and H_i. For odd m, I computed the similarity between $(T_{NT})^m H_i$ and $(T_{NT})^1 H_i$; $(T_F)^m H_i$ and $(T_F)^1 H_i$; $(T_{NT})^m H_i$ and $(T_F)^1 H_i$; and $(T_F)^m H_i$ and $(T_{NT})^1 H_i$.

As in Experiments 1 and 2 (Sections 2.4 and 2.5, respectively), each model was evaluated using the challenge dataset(s) generated from the dataset(s) that the model was originally fine-tuned on.

2.6.2 Results

We observe that depth-$\leq$5 inoculation drastically increases the similarity between $(T_{NT})^m H_i$ and $(T_{NT})^1 H_i$ for all three models for odd values of m (Figure 2.2a), but *decreases* the similarity between $(T_F)^m H_i$ and $(T_F)^1 H_i$ for $m \geq 5$ (Figure 2.2b)—recall that for odd m, $(T_{NT})^m H_i$ / $(T_F)^m H_i$ should be synonymous with $(T_{NT})^1 H_i$ / $(T_F)^1 H_i$. The results are



(a) $sim((T_{NT})^n H_i, (T_{NT})^1 H_i)$ (b) $sim((T_F)^n H_i, (T_F)^1 H_i)$

Figure 2.2: Mean cosine similarity between $(T_{NT})^n H_i$ / $(T_{NT})^1 H_i$ (Figure 2.2a) and $(T_F)^n H_i$ / $(T_F)^1 H_i$ (Figure 2.2b) for the three RoBERTa models before (dashed) and after (solid) depth-$\leq$5 T_{NT} inoculation.

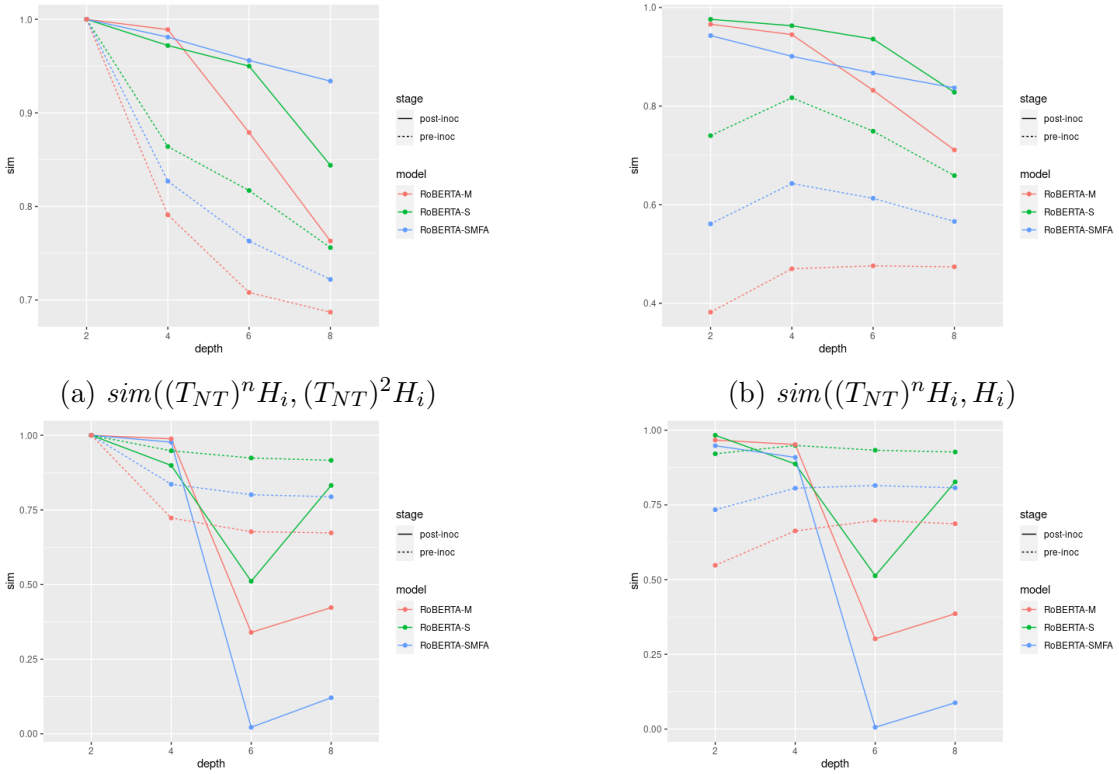
Model	Before	After
$RoBERTa_M$	0.996	0.268
$RoBERTa_S$	0.996	0.712
$RoBERTa_{SMFA}$	0.996	0.646

Table 2.6: Cosine similarity between the RoBERTa models' (mean-pooled) embeddings of the strings *"false"* and *"not true"* before and after depth-$\leq$5 inoculation.

analogous for even values of m (see Figure 2.3).

Additionally, as m increases, mean cosine similarity decreases between $(T_F)^m H_i$ and $(T_{NT})^2 H_i$, and $(T_F)^m H_i$ and $(T_{NT})^1 H_i$ (see Figures 2.4a and 2.4b, respectively). We also observe decreases in cosine similarity between $(T_F)^m H_i$ and $(T_{NT})^m H_i$ for even and odd $m > 4$ (see Figure 2.4c).

These results indicate that the inoculation procedure conducted in Experiment 1 (Section 2.4) has lead to catastrophic forgetting. In particular, it seems that learning to cancel double negation for T_{NT} has drastically altered the models' encodings of the string *"not true"*, pulling its representation in the embedding space away from those of similar phrases such as *"false"*. This conjecture is supported by Table 2.6: we observe that—before inoculation—the models' representations of the strings *"not true"* and *"false"* are nearly identical. However, after depth-$\leq$5 T_{NT} inoculation, the models' representations of the two strings are substantially further apart in the embedding space.



(a) $sim((T_{NT})^n H_i, (T_{NT})^2 H_i)$

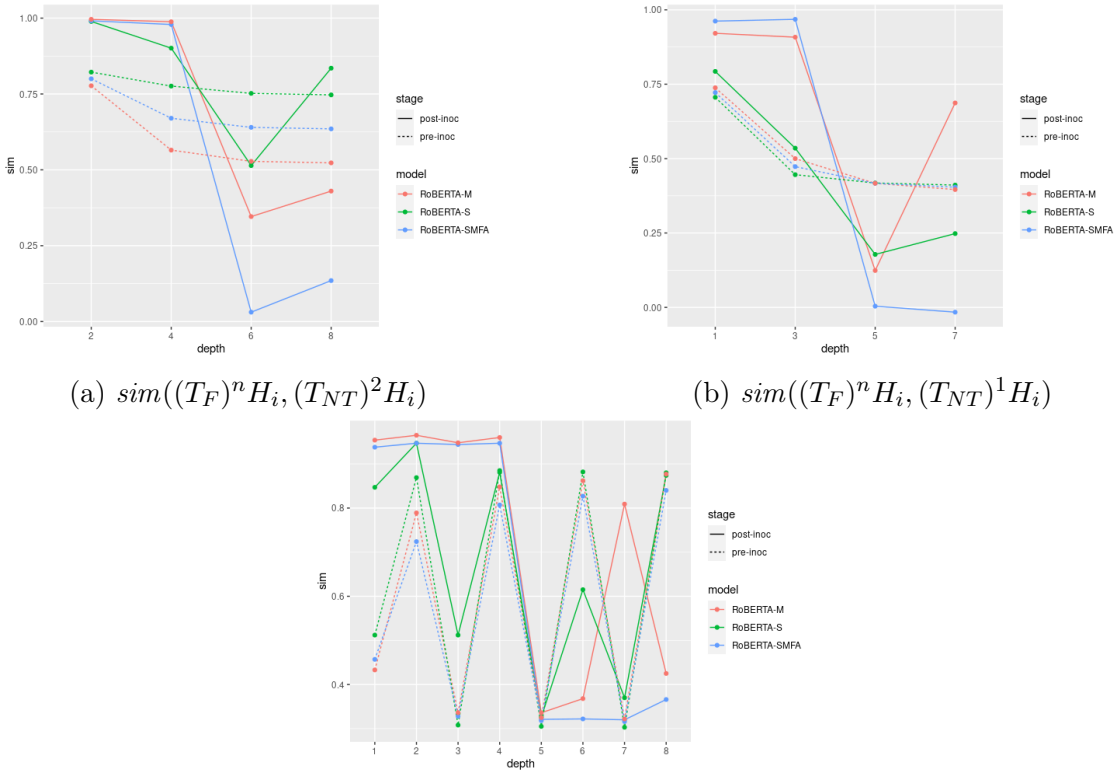
(b) $sim((T_{NT})^n H_i, H_i)$

(c) $sim((T_F)^n H_i, (T_F)^2 H_i)$

(d) $sim((T_F)^n H_i, H_i)$

Figure 2.3: Mean cosine similarity between even-depth externally-negated examples and their original/depth-2-negated counterparts for the three RoBERTa models before (dashed) and after (solid) depth-$\leq$5 T_{NT} inoculation.

Furthermore, the results of this experiment indicate that the models have not learned the linguistic function of negation during pre-training or original fine-tuning on the MNLI and SNLI datasets, analogous to the findings of Yuan et al. (2023) with respect to deductive reasoning tasks. Aside from the results in Table 2.3 indicating that these NLI models simply treat external negation prefixes as distractors before inoculation, note that if the models already understood the logical function of prefixes such as T_{NT}, then further refining the models' knowledge of the function of that prefix—i.e. fine-tuning on the depth-$\leq$5 T_{NT} data—should not significantly alter its representation in the embedding space relative to highly similar prefixes such as T_F, contrary to what we observe in Table 2.6.



(a) $sim((T_F)^n H_i, (T_{NT})^2 H_i)$

(b) $sim((T_F)^n H_i, (T_{NT})^1 H_i)$

(c) $sim((T_F)^n H_i, (T_{NT})^n H_i)$

Figure 2.4: Mean cosine similarity between examples modified with the T_F and T_{NT} prefixes, for the three RoBERTa models before (dashed) and after (solid) depth-$\leq$5 T_{NT} inoculation.

2.7 Discussion

The results of Experiments 1-3 (Sections 2.4, 2.5, 2.6) raise the question as to *why* these models are unable to inductively learn the law of the excluded middle (LEM)—especially in light of Theorem 1, which states that (in theory) transformers are able to model LEM with respect to NLI tasks. A reasonable explanation for this seemingly paradoxical state of affairs can be found within the conclusions of Theorem 1 itself.

Note that the proof of Theorem 1 relies on a function f that re-structures the input data (as mentioned in Section 2.3); it is possible that the structure (or lack thereof) of purely textual data may be insufficient for transformers to inductively learn to model LEM.

Additionally, recall that the proof of Theorem 1 does *not* establish the (inductive) *learnability* of LEM; it may be the case that the specific parameter values required to model

the role of external negation in the context of NLI tasks cannot be reached by training on any NLI dataset using gradient descent or any other currently known training procedures. It may also be the case that the function of external negation is in fact learnable, but only via the brute-force approach of training these models on multiple-depth external negation for every such prefix. In other words, encoder transformers may not be capable of *inductively* learning LEM—at least not with standard training procedures.

Hosseini et al. (2021) and Asai and Hajishirzi (2020) propose training procedures designed to enhance language models' ability to learn the role of negation, which may provide fruitful avenues for improving transformer NLI models' performance on the tasks laid out in Experiments 1-3 (Sections 2.4, 2.5, 2.6). Hosseini et al. (2021) introduce *unlikelihood with reference* training for masked language models, which penalizes models for predicting unlikely tokens in negated contexts—for example, a model would be penalized for predicting *fly* in the context *"birds cannot [MASK]"*. After unlikelihood with reference training, the authors record marginal improvement (∼1-2%) for BERT on negation-augmented SNLI and MNLI datasets (see Hossain et al., 2020).

Asai and Hajishirzi (2020) use logic-based regularization and data augmentation to improve language models' *transitive* and *symmetric* consistency (c.f. Jang, Kwon, and Lukasiewicz, 2022)—in particular, negation is subsumed under their notion of symmetric consistency. Using this approach, the authors record marked improvement over the SoTA on a variety of question-answering tasks, although they do not evaluate this regularization method on any NLI datasets.

However, Hosseini et al. (2021) and Asai and Hajishirzi (2020) do not explicitly study the efficacy of their respective training methods with respect to double-negation cancellation. Therefore, it is unclear whether the improvements obtained by their approaches would translate to a task such as LEM, and I leave an evaluation thereof to future work.

2.8 Conclusion

The results of Experiments 1-3 (Sections 2.4, 2.5, 2.6) demonstrate that near-SoTA transformer NLI models struggle to inductively learn the law of the excluded middle (LEM). Furthermore, the results of Experiment 1 (Section 2.4) strongly suggest that all six NLI models studied in this work learned to treat the external negation prefix *"it is not true that"* as a distractor when initially fine-tuned on the NLI dataset(s) (see Table 2.3). Experiment 1 also suggests that DeBERTa and BART models are incapable of learning to inductively generalize LEM, despite extensive fine-tuning.

These findings lend further support to a large body of existing evidence (e.g. Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023; Laverghetta Jr. et al., 2021; Rogers, Kovaleva, and Rumshisky, 2020; Ettinger, 2020; Laverghetta Jr. and Licato, 2022; Kassner and Schütze, 2020) indicating that transformers are unable to model the meaning of negation. Unique to this work is our finding that certain encoder transformers (in particular, RoBERTa) can learn LEM for a single external negation prefix.

While the three RoBERTa models did manage to grasp the function of the prefix *"it is not true that"*, the process of learning this behavior resulted in catastrophic forgetting, entirely inhibiting their generalization of this pattern to the highly similar prefix *"it is false that"* (see Sections 2.5 and 2.6).

Theorem 1 proves that encoder transformers are—in principle—capable of modeling LEM for arbitrary-length sequences of any combination of external negation prefixes with respect to any NLI dataset. This suggests that these superficial models' inability to inductively learn LEM might not be a consequence of their transformer architectures, but rather may result from the (lack of) structure of their input data and/or the procedure used to train them.

In addition to exposing this critical weakness of near-SoTA NLI models, the experiments in this chapter provide an excellent battery of tests with which to probe the language model over logical forms that I introduce in Chapter 4 (see Chapter 6), and allow for a fine-grained comparison between that model's logical reasoning abilities and those of the superficial models

evaluated here.

2.9 Proof of Theorem 1

In this section, I formally prove Theorem 1 of Section 2.3: uninterested readers may safely skip to Chapter 3. In Section 2.9.1, I discuss the fragment of first-order logic in which this proof takes place, and introduce the notation that I employ in Section 2.9.2; the proof itself is located in Section 2.9.3.

2.9.1 FOC[+;MOD]

Chiang, Cholak, and Pillay (2023) prove that FOC[+;MOD] (a variant of first-order logic defined over strings over a finite alphabet Σ; see Immerman, 2012) is both an upper bound for fixed-precision transformer encoders and a lower bound for arbitrary-precision encoder transformer encoders, in the sense that every language that is recognizable by a fixed-precision encoder transformer binary classifier is definable by a sentence of FOC[+;MOD] (Chiang, Cholak, and Pillay, 2023, Theorem 2), and every language defined by a sentence of FOC[+;MOD] is recognizable by an arbitrary-precision encoder transformer binary classifier (Chiang, Cholak, and Pillay, 2023, Theorem 5). Given an FOC[+;MOD] formula ϕ, the language defined by ϕ is the set of all strings $\sigma \in \Sigma^*$ such that ϕ holds with respect to σ.

The syntax of FOC[+;MOD] consists of two sorts:

- *Positions*: positive integer variables p that range over positions in strings σ.
- *Counts*: variables x ranging over the rational numbers, and terms $c_0 + c_1 x_1 + \cdots + c_n x_n$, where each c_i is a (constant) rational number and each x_i is a count variable.

Formulas of FOC[+;MOD] are defined as one of:

- $\top$ (true) or $\bot$ (false).

- $Q_a(p)$, where $a \in \Sigma$, and $Q_a(p) := \sigma_p = a$
- $MOD_b^a(p)$, where $a \geq 0$, $b > 0$, and p is a position variable; $MOD_b^a(p) := p \equiv_b a$
- $\phi \wedge \psi$, $\phi \vee \psi$, or $\neg\psi$, where ϕ and ψ are formulas.[11]
- $x_1 = x_2$ or $x_1 < x_2$, where x_1, x_2 are in the sort of counts.[12]
- $\exists x.\phi$ or $\forall x.\phi$, where x is a count variable and ϕ is a formula.
- $\exists^{=x} p.\phi$, where x is a count variable, p is a position variable ($\exists^{=x} p.\phi$ binds p but leaves x free), and ϕ is a formula; $\exists^{=x} p.\phi$ holds if and only if ϕ is true for exactly x values of p.

Note that FOC[+;MOD] does *not* permit arithmetic operations (addition or multiplication) or comparisons ($=$, $<$) of position variables, only of count variables. This is the primary motivation for much of the machinery introduced in the proof of Theorem 1 (Section 2.9.3).

2.9.2 Notation

I now introduce additional notation employed in the proof of Theorem 1:

- $\sigma \,||\, \sigma'$: denotes the concatenation of the strings σ and σ'. Note that when convenient (and unambiguous), I omit the operator and write $\sigma\sigma'$ to denote $\sigma \,||\, \sigma'$.
- $\Big|\Big|_{i=k}^{n} (\ldots)$: denotes iterated string concatenation.
- $|\sigma|$: unless otherwise specified, denotes the length of the string σ.
- σ_i: denotes the i^{th} character of the string σ.

[11] We can derive $\phi \rightarrow \psi$ and $\phi \leftrightarrow \psi$ as $\psi \vee \neg\phi$ and $\phi \rightarrow \psi \wedge \psi \rightarrow \phi$, respectively.

[12] We can derive $x_1 \leq x_2$ as $x_1 = x_2 \vee x_1 < x_2$, $x_1 > x_2$ as $x_2 < x_1$, $x_1 \geq x_2$ as $x_2 \leq x_1$, and $x_1 \neq x_2$ as $\neg(x_1 = x_2)$.

- $\Sigma^* = \bigcup_{i=1}^{\infty} \Sigma^i$: denotes the set of all *non-empty* strings over the alphabet Σ. Note that unless otherwise specified, I slightly abuse notation and let A^* (for any $A \subseteq \Sigma^*$) denote the set of "flattened" strings of A—i.e. $A^* = \bigcup_{i=1}^{\infty} \bigcup_{a \in A^i} \{ \mathop{\Big\|}_{k=1}^{|a|} a_k \}$ so that for all $a' \in A^*$, $a' \in \Sigma^*$.

- ϵ: denotes the empty string.

- $\sigma_{i:j} = \mathop{\Big\|}_{k=i}^{j} \sigma_k$: denotes the substring spanning the i^{th} to j^{th} (inclusive) characters of σ; if $i = j$, then $\sigma_{i:j} = \sigma_i$.

- $\sigma_{i:}$, $\sigma_{:j}$: denote $\sigma_{i:|\sigma|}$ when $i \leq |\sigma|$ and $\sigma_{1:j}$ when $j \geq 1$, respectively. If $i > |\sigma|$, then $\sigma_{i:} = \epsilon$.

- $\sigma^n = \mathop{\Big\|}_{i=1}^{n} \sigma$: denotes the string σ repeated n times ($\sigma^0 = \epsilon$).

- $\phi[x \Rightarrow y] = \lambda x.[\phi](y)$: denotes the formula obtained from ϕ by replacing all instances of the free variable x with the variable (or constant) y.

- $[\phi](\sigma) = \sigma \models \phi$: indicates that the formula ϕ holds for the string σ (i.e. σ belongs to the language defined by ϕ).

2.9.3 Proof

Let $\Lambda = \{\mathcal{E}, \mathcal{N}, \mathcal{C}\}$ denote the set of NLI labels and let Σ' denote the input alphabet of (i.e. set of tokens for) the transformer T—we may assume without loss of generality that Λ and Σ' are disjoint (i.e. $\Lambda \cap \Sigma' = \emptyset$); Theorem 1 applies only to *encoder* transformers, so we need not consider the labeling approach taken by encoder-decoder or decoder-only transformers.

By Chiang, Cholak, and Pillay (2023) Theorem 2, T corresponds to the FOC[+;MOD] formula S_T defined in Equation 2.4. To be explicit: Chiang, Cholak, and Pillay (2023) Theorem 2 guarantees that there exists some FOC[+;MOD] formula S_T that defines the language recognized by T. For each $(P_k, H_k, L_k) \in D$, the input to S_T is the string $P_k H_k L_k$:

for all $x \in \Lambda$, $[S_T](P_k H_k L_k)$ holds if and only if the transformer T assigns the label L_k to (P_k, H_k).

$$S_T := \bigwedge_{x \in \Lambda} \phi_x \leftrightarrow \exists^{=1} p.Q_x(p) \tag{2.4}$$

Note that we may assume the existence of $\phi_{\mathcal{E}}$, $\phi_{\mathcal{N}}$, and $\phi_{\mathcal{C}}$ (and therefore S_T) as in Equation 2.4 without loss of generality. Regardless of the approach that the particular transformer T takes to predicting labels, the output of T with respect to an input $\sigma \in (\Sigma')^*$ ($\mathcal{O}_T(\sigma)$) must be an element of Λ. As such, for each $x \in \Lambda$ and $\sigma \in (\Sigma')^*$, $[\phi_x](\sigma) := \mathcal{O}_T(\sigma) = x$.

Let $\Sigma = \Sigma' \cup \{\Omega\}$, where Ω is a special padding character introduced for formal reasons, and distinct from the actual padding character used by the transformer T. For any $\sigma \in (\Sigma')^*$, define $f(\sigma) \in \Sigma^*$ as follows in Equation 2.5, where w is the fixed input length specified in Theorem 1.

$$f(\sigma) := \mathop{||}_{i=1}^{|\sigma|+1} (\ \Omega^{i-1} \ || \ \sigma_{i:} \ || \ \Omega^{w-|\sigma|} \ ) \tag{2.5}$$

For all integer count terms $1 < b \leq w$, define $MODC_b(a, x)$ as follows (Equation 2.6), where a and x are count variables:

$$MODC_1(a, x) := \top \tag{2.6a}$$

$$MODC_b(a, x) := \bigvee_{y=-w}^{w} yb + a = x \tag{2.6b}$$

Note that by Chiang, Cholak, and Pillay (2023) Theorem 1, we may assume without loss of generality that each ϕ_x in Equation 2.4 is in normal form (for some integer $k \geq 0$), as in Equation 2.7.

$$\phi_x = \exists z_1 \ldots \exists z_k [\bigwedge_{i=1}^{k} \exists^{=z_i} p.(\phi_x)_i \wedge \chi] \tag{2.7}$$

Where each $(\phi_x)_i$ is quantifier-free and has no free count variables, and χ is quantifier-free.

Now, for each such $(\phi_x)_i$, construct $\alpha((\phi_x)_i)$ as follows: for each $a \in \Sigma'$ such that $Q_a(p)$ appears in $(\phi_x)_i$, replace $Q_a(p)$ with $Q'_a(p)$ as defined in Equation 2.8—where p is a position variable in the former, and a count variable in the latter—and replace each instance of a modular predicate $MOD^x_y(p)$ with $MODC_y(x, p)$ (where again p is a position variable in the former, and a count variable in the latter).

$$Q'_a(p) := \exists^{=p} p'[Q_a(p') \wedge \bigvee_{i=1}^{w} (MOD^i_w(p') \wedge p = i)] \tag{2.8}$$

Lemma 2.1. *For any $\sigma \in (\Sigma')^*$ such that $|\sigma| \leq w$, all $a \in \Sigma'$, and all $1 \leq p \leq w$:* $[Q'_a(p)](f(\sigma)) \leftrightarrow [Q_a(p)](\sigma)$

Proof. First, assume $[Q_a(p)](\sigma)$ holds. By assumption, $\sigma_p = a$, so by construction (Equation 2.5), $f(\sigma)_{yp} = a$ for all $1 \leq y \leq p$ and $f(\sigma)_{y'p} = \Omega$ for all $y' > p$. Therefore $[Q_a(p)](\sigma) \rightarrow [Q'_a(p)](f(\sigma))$ by definition (Equation 2.8).

Now, assume $[Q'_a(p)](f(\sigma))$ holds. By assumption and construction (Equation 2.5), $f(\sigma)_{yp} = a$ for all $1 \leq y \leq p$, so in particular $f(\sigma)_p = a$. By construction, $f(\sigma)_{:|\sigma|} = \sigma$. This implies that $\sigma_p = a$; therefore $[Q'_a(p)](f(\sigma)) \rightarrow [Q_a(p)](\sigma)$. □

Now, for any count variables p, z and any FOC[+;MOD] formula ϕ, define $E(p, z, \phi)$ as follows (Equation 2.9).

$$E_1^i(p,\phi) := \bigwedge_{j=1}^{i} \phi[p \Rightarrow m_j] \wedge m_j \leq w \tag{2.9a}$$

$$E_2^i := \bigwedge_{a=1}^{i-1} \bigwedge_{b=a+1}^{i} m_a \neq m_b \tag{2.9b}$$

$$E_3^{i+2}(p,\phi) := \exists m_1 \ldots m_{i+2}[E_1^{i+2}(p,\phi) \wedge E_2^{i+2}] \tag{2.9c}$$

$$E_3^1(p,\phi) := \exists m_1 . E_1^1(p,\phi) \tag{2.9d}$$

$$E_3^0(p,\phi) := \top \tag{2.9e}$$

$$E(p,z,\phi) := \bigvee_{i=0}^{w} (E_3^i(p,\phi) \wedge z = i) \tag{2.9f}$$

Where $E_1^i(-,-)$, E_2^i, and $E_3^i(-,-)$ are defined for all integers $1 \leq i \leq w$, $2 \leq i \leq w$, and $0 \leq i \leq w$, respectively.

Now, for each $(\phi_x)_i$ in Equation 2.7, define $A((\phi_x)_i)$ as in Equation 2.10, where z_i and p are free count variables.

$$A_1((\phi_x)_i) := E(p, z_i, \alpha((\phi_x)_i)) \tag{2.10a}$$

$$A_2((\phi_x)_i) := \neg\exists y[y > z_i \wedge E(p, y, \alpha((\phi_x)_i))] \tag{2.10b}$$

$$A((\phi_x)_i) := A_1((\phi_x)_i) \wedge A_2((\phi_x)_i) \tag{2.10c}$$

Lemma 2.2. *For any $\sigma \in (\Sigma')^*$ such that $|\sigma| \leq w$, all $x \in \Lambda$, and all $(\phi_x)_i$ as in Equation 2.7:* $[\exists z_i \exists^{=z_i} p.(\phi_x)_i](\sigma) \leftrightarrow [\exists z_i . A((\phi_x)_i)](f(\sigma))$

Proof. First, note that $(\phi_x)_i$ is quantifier-free and has no free count variables (Chiang, Cholak, and Pillay, 2023, Theorem 1); therefore $(\phi_x)_i$ consists only of positional ($Q_a(p)$) and modular ($MOD_y^x(p)$) predicates—where the only bound position variable is p—and logical operators acting on them. $A((\phi_x)_i)$ is constructed from $(\phi_x)_i$ by replacing each instance of $Q_a(p)$ and

$MOD_y^x(p)$ with $Q'_a(p)$ and $MODC_y(x,p)$ (respectively), where p is a position variable in the first pair of terms, and a count variable in the second.

By Lemma 2.1, $[Q_a(p)](\sigma) \leftrightarrow [Q'_a(p)](f(\sigma))$ for all $1 \leq p \leq w$, where p is a position variable in the left-hand side of the equation and a count variable in the right-hand side. Similarly, for all p, x and all $1 \leq y \leq w$, $MOD_y^x(p) \leftrightarrow MODC_y(x,p)$ by construction (Equation 2.6), where again p is a position variable in the left-hand side of the equation and a count variable in the right-hand side.

Therefore, for all $1 \leq p \leq w$, $(\phi_x)_i$ holds with respect to σ if and only if $\alpha((\phi_x)_i)$ holds with respect to $f(\sigma)$.

By construction (Equation 2.9), $E(p,z,\phi)$ holds for any predicate ϕ with the count variable p free if and only if there are $\geq z$ unique values of p such that ϕ holds. By definition (Equation 2.10), $A((\phi_x)_i)$ holds if and only if there are exactly z_i values of p such that $\alpha((\phi_x)_i)$ holds. □

Now, for each ϕ_x in Equation 2.4, define $A(\phi_x)$ as in Equation 2.11.

$$A(\phi_x) := \exists z_1 \ldots \exists z_k [\bigwedge_{i=1}^{k} A((\phi_x)_i) \wedge \chi] \tag{2.11}$$

Lemma 2.3. *For all $x \in \Lambda$ and all $\sigma \in (\Sigma')^*$ such that $|\sigma| \leq w$: $[\phi_x](\sigma) \leftrightarrow [A(\phi_x)](f(\sigma))$*

Proof. By Lemma 2.2, each $A((\phi_x)_i)$ of Equation 2.11 holds for $f(\sigma)$ if and only if each $(\phi_x)_i$ holds for σ. As such, for each bound count variable z_i, the set of cardinality z_i of count values that make $A((\phi_x)_i)$ true with respect to $f(\sigma)$ is identical to the set of position values that make $(\phi_x)_i$ true with respect to σ. The predicate χ contains no position variables (Chiang, Cholak, and Pillay, 2023, Theorem 1), and is defined identically in Equation 2.11 as in Equation 2.7; therefore, χ (within $A(\phi_x)$) holds for $f(\sigma)$ if and only if χ (within ϕ_x) holds for σ. □

Now, for each external negation prefix $\eta \in N$, define $\psi_\eta(i)$ and $\psi'_\eta(i,j)$ as in Equation 2.12, where i, j are count variables and $Q'_{(-)}(-)$ is defined as in Equation 2.8.

$$\psi_\eta(i) := \bigwedge_{k=0}^{|\eta|-1} Q'_{\eta_k}(i+k) \tag{2.12a}$$

$$\psi'_\eta(i,j) := \psi_\eta(i) \wedge i + |\eta| - 1 = j \tag{2.12b}$$

Then define $\psi(i)$ and $\psi'(i,j)$ as in Equation 2.13, where i and j are count variables.

$$\psi(i) := \bigvee_{\eta \in N} \psi_\eta(i) \tag{2.13a}$$

$$\psi'(i,j) := \bigvee_{\eta \in N} \psi'_\eta(i,j) \tag{2.13b}$$

Now define $\rho(i,j)$ as in Equation 2.14, where i and j are again count variables.

$$\rho_1(k,a,b,i,j) := i \leq a \leq k \wedge k \leq b \leq j \wedge \psi'(a,b) \tag{2.14a}$$

$$\rho(i,j) := \forall k[i \leq k \leq j \rightarrow \exists a,b.\rho_1(k,a,b,i,j)] \tag{2.14b}$$

Lemma 2.4. *For any $\sigma \in (\Sigma')^*$ such that $|\sigma| \leq w$, and all $1 \leq i < j \leq w$: $[\rho(i,j)](f(\sigma)) \leftrightarrow \sigma_{i:j} \in N^*$—i.e. $\rho(i,j)$ holds for $f(\sigma)$ if and only if the span $i \rightarrow j$ in σ is a sequence of one or more external negation prefixes.*

Proof. I first prove the right-to-left direction: $\sigma_{i:j} \in N^* \rightarrow [\rho(i,j)](f(\sigma))$. The proof proceeds by induction. First, assume that σ is a single external negation prefix (i.e. $\sigma_{i:j} \in N$). Then by assumption and definition (Equation 2.12), $\psi'_{\sigma_{i:j}}(i,j)$ holds; by definition (Equation 2.13), this implies $\psi'(i,j)$. For all $i \leq k \leq j$, let $a = i$, $b = j$: by definition (Equation 2.14), $\rho_1(k,a,b,i,j)$ holds. This implies $\rho(i,j)$. This proves the base case.

Now suppose $\sigma_{i:j} = \eta \,||\, \eta'$, with $\eta \in N^*$ and $\eta' \in N$. By the inductive hypothesis, $\rho(i, i+|\eta|-1)$ holds. By the base case above, $\rho(i+|\eta|, j)$ holds. It now remains to prove

that $\rho(i, i+|\eta|-1) \wedge \rho(i+|\eta|, j) \rightarrow \rho(i, j)$. For all $1 \leq k \leq j$, if $k < i+|\eta|$, then there exist $a, b < i+|\eta|$ such that $\rho_1(k, a, b, i, j)$ (by the validity of $\rho(i, i+|\eta|-1)$), and if $k \geq i+|\eta|$, there exist $a, b \geq i+|\eta|$ such that $\rho_1(k, a, b, i, j)$ (by the validity of $\rho(i+|\eta|, j)$); therefore, $\rho(i, j)$ holds. This proves the induction step.

I now prove the right-to-left direction by contradiction: assume $\rho(i, j)$ and $\sigma_{i:j} \notin N^*$. By assumption, there exists $\eta \in N^* \cup \{\epsilon\}$ such that η is a substring of $\sigma_{i:j}$. For all $i \leq k \leq j$ such that σ_k is not contained within η: $\neg\exists a, b.\rho_1(k, a, b, i, j)$, by the assumption that external negation prefixes do not overlap (see Theorem 1). Therefore, $\rho(i, j)$ does not hold—this is a contradiction. □

Now define $\rho'(i, j)$ as in Equation 2.15.

$$\rho'_1(a, b, i, j) := (a \leq i \wedge b > j) \vee (a < i \wedge b \geq j) \tag{2.15a}$$

$$\rho'_2(a, b, i, j) := a > 1 \wedge \rho'_1(a, b, i, j) \tag{2.15b}$$

$$\rho'(i, j) := \rho(i, j) \wedge \neg\exists a, b[\rho'_2(a, b, i, j) \wedge \rho(a, b)] \tag{2.15c}$$

For all $x \in \Lambda$, define $F_1(x)$ as in Equation 2.16.

$$F_1(x) := \neg\exists i, j[j > i > 1 \wedge \rho'(i, j)] \wedge A(\phi_x) \tag{2.16}$$

$F_1(x)$ is intended to coincide with ϕ_x on any $(P_k, H_k, L_k) \in D$—i.e. an example in which the hypothesis is *not* externally negated. The term $j > i > 1$ in Equation 2.16 allows for the possibility that the premise P_k may be externally negated in the original dataset D.

Lemma 2.5. *For all $x \in \Lambda$ and all $\sigma \in (\Sigma')^*$ such that $|\sigma| \leq w$ and there does not exist $\eta \in N^*$ such that η is a subsequence of $\sigma_{2:}$: $[\phi_x](\sigma) \leftrightarrow [F_1(x)](f(\sigma))$*

Proof. By Lemma 2.3, $[\phi_x](\sigma) \leftrightarrow [A(\phi_x)](f(\sigma))$. By assumption, $\neg\exists i, j[j > i > 1 \wedge \rho'(i, j)]$ holds for all such $f(\sigma)$. □

Now, define $A'(\phi_x)$ by replacing each predicate $Q'_a(p)$ in $A(\phi_x)$ (Equation 2.11) with $\beta(Q'_a(p))$, as defined in Equation 2.17, where i and j are free count variables in $A'(\phi_x)$.

$$\beta_1(Q'_a(p)) := p < i \wedge Q'_a(p) \tag{2.17a}$$

$$\beta_2(Q'_a(p)) := p \geq i \wedge Q'_a(p + (j - i) + 1) \tag{2.17b}$$

$$\beta(Q'_a(p)) := \beta_1(Q'_a(p)) \vee \beta_2(Q'_a(p)) \tag{2.17c}$$

Lemma 2.6. *For all $(P_k, H_k, L_k) \in D$, all $x \in \Lambda$, and all $\eta \in N^*$ such that $|P_k \eta H_k| \leq w$: $[\phi_x](P_k H_k) \leftrightarrow [A'(\phi_x)](f(P_k \eta H_k))$ when the free variables $i = |P_k| + 1$, $j = |P_k \eta|$ in Equation 2.17.*

Proof. I first prove that $[A(\phi_x)](f(P_k H_k)) \leftrightarrow [A'(\phi_x)](f(P_k \eta H_k))$. Note that $A'(\phi_x)$ is constructed from $A(\phi_x)$ by replacing each instance of $Q'_a(p)$ with $\beta(Q'_a(p))$. It therefore suffices to prove that for all $a \in \Sigma'$ and all $1 \leq p \leq w$: $[Q'_a(p)](f(P_k H_k)) \leftrightarrow [\beta(Q'_a(p))](f(P_k \eta H_k))$.

If $p \leq |P_k|$, then $[Q'_a(p)](f(P_k H_k)) \leftrightarrow [\beta(Q'_a(p))](f(P_k \eta H_k))$ by definition (Equation 2.17). Otherwise, $[Q'_a(p)](f(P_k H_k)) \leftrightarrow [\beta(Q'_a(p))](f(P_k \eta H_k))$ if and only if $(P_k H_k)_p = (P_k \eta H_k)_{p+(j-i)+1}$. By assumption, $p + (j - i) + 1 = p + (|P_k \eta| - (|P_k| + 1)) + 1 = p + |\eta|$ and $(P_k H_k)_p = (P_k \eta H_k)_{p+|\eta|}$.

By Lemma 2.3 and the above result, we have: $[\phi_x](P_i H_i) \leftrightarrow [A(\phi_x)](f(P_i H_i)) \leftrightarrow [A'(\phi_x)](f(P_i \eta H_i))$. □

Now, define $F_2(x)$ as in Equation 2.18, where $G(\mathcal{E}) = \mathcal{C}$, $G(\mathcal{C}) = \mathcal{E}$, and $G(\mathcal{N}) = \mathcal{N}$.

$$\gamma_x^1(n) := MODC_2(1,n) \wedge A'(\phi_{G(x)}) \tag{2.18a}$$

$$\gamma_x^2(n) := MODC_2(0,n) \wedge A'(\phi_x) \tag{2.18b}$$

$$\gamma_x^3(k) := i \leq k \leq j \wedge \psi(k) \tag{2.18c}$$

$$\gamma_x^4(n) := E(k,n,\gamma_x^3(k)) \tag{2.18d}$$

$$\gamma_x^5(n) := \neg\exists y[y > n \wedge E(k',y,\gamma_x^3(k'))] \tag{2.18e}$$

$$\gamma_x := \exists n[\gamma_x^4(n) \wedge \gamma_x^5(n) \wedge (\gamma_x^1(n) \vee \gamma_x^2(n))] \tag{2.18f}$$

$$F_2(x) := \exists i,j[j > i > 1 \wedge \rho'(i,j) \wedge \gamma_x] \tag{2.18g}$$

Lemma 2.7. *Define $N_0, N_1 \subset N^*$ as the sets of even- and odd-length—in terms of number of prefixes, rather than characters—sequences of external negation prefixes, respectively. Then for all $x \in \Lambda$ and all $(P_k, H_k, L_k) \in D$:*

i. for all $\eta \in N_0$: $[\phi_x](P_kH_k) \leftrightarrow [F_2(x)](f(P_k\eta H_k))$

ii. for all $\eta' \in N_1$: $[\phi_{G(x)}](P_kH_k) \leftrightarrow [F_2(x)](f(P_k\eta' H_k))$

Proof. I first prove (i). By Lemma 2.4 and the definition of $\rho'(i,j)$ (Equation 2.15), the respective values of i,j that make the term $j > i > 1 \wedge \rho'(i,j)$ hold in Equation 2.18 are $i = |P_k| + 1$ and $j = |P_k\eta|$. By the definitions of $E(k,n,-)$, $\psi(-)$, and γ_x (Equations 2.9, 2.13, and 2.18, respectively)—and the assumption that $\eta \in N_0$—the value of n that makes $[\gamma_x](f(P_k\eta H_k))$ hold is even. Therefore, the term $MODC_2(0,n)$ in $\gamma_x^2(n)$ holds, and so $[A'(\phi_x)](f(P_k\eta H_k)) \leftrightarrow [F_2(x)](f(P_k\eta H_k))$.

By Lemma 2.6 and the above result, we have: $[\phi_x](P_kH_k) \leftrightarrow [A'(\phi_x)](f(P_k\eta H_k)) \leftrightarrow [F_2(x)](f(P_k\eta H_k))$.

I now prove (ii); the proof proceeds in a similar fashion as that of (i) above. But now n is odd, and so the term $MODC_2(1,n)$ in $\gamma_x^1(n)$ holds. Therefore, $[A'(\phi_{G(x)})](f(P_k\eta' H_k)) \leftrightarrow [F_2(x)](f(P_k\eta' H_k))$.

Again by Lemma 2.6 and the above result: $[\phi_{G(x)}](P_k H_k) \leftrightarrow [A'(\phi_{G(x)})](f(P_k \eta H_k)) \leftrightarrow [F_2(x)](f(P_k \eta H_k))$. □

For all $x \in \Lambda$, define $F(x)$ as follows (Equation 2.19).

$$F(x) := F_1(x) \vee F_2(x) \tag{2.19}$$

Now, define the formula $S_{T'}$ in Equation 2.20 below.

$$S_{T'} := \bigwedge_{x \in \Lambda} F(x) \leftrightarrow \exists^{=1} p.Q_x(p) \tag{2.20}$$

Lemma 2.8. *For all $(P_k, H_k, L_k) \in D$, all $\eta \in N_0$ such that $|P_k \eta H_k| \leq w$, and all $\eta' \in N_1$ such that $|P_k \eta' H_k| \leq w$:*

i. $[S_{T'}](f(P_k H_k) L_k) \leftrightarrow [S_T](P_k H_k L_k)$

ii. $[S_{T'}](f(P_k \eta H_k) L_k) \leftrightarrow [S_T](P_k H_k L_k)$

iii. $[S_{T'}](f(P_k \eta' H_k) G(L_k)) \leftrightarrow [S_T](P_k H_k L_k)$

Proof. By Lemma 2.5, $[F_1(L_k)](f(P_k H_k))$ holds if and only if $[\phi_{L_k}](P_k H_k)$ does as well, for all $(P_k, H_k, L_k) \in D$. $F_2(x)$ does not hold for any $x \in \Lambda$ by definition (Equation 2.18), and $[F_1(x)](f(P_k H_k)) \leftrightarrow [\phi_x](P_k H_k)$ for any $x \in \Lambda - \{L_k\}$ by Lemma 2.5. This proves (i).

For all $\eta \in N_0$ such that $|P_k \eta H_k| \leq w$, $[F_1(x)](f(P_k H_k))$ does not hold for any $x \in \Lambda$ by definition (Equation 2.16), and $[F_2(x)](f(P_k H_k)) \leftrightarrow [\phi_x](P_k H_k)$ for all $x \in \Lambda$ by Lemma 2.7(i). This proves (ii).

For all $\eta' \in N_1$ such that $|P_k \eta' H_k| \leq w$, $[F_1(x)](f(P_k H_k))$ does not hold for any $x \in \Lambda$ by definition, and $[F_2(x)](f(P_k H_k)) \leftrightarrow [\phi_{G(x)}](P_k H_k)$ for all $x \in \Lambda$ by Lemma 2.7(ii). This proves (iii). □

By Chiang, Cholak, and Pillay (2023) Theorem 5, there exists a transformer encoder T'' that recognizes the language defined by $S_{T'}$. By Lemma 2.8(i), $Acc(T'', f(D)) = Acc(T, D)$,

and $Acc(T'', \{f(P_i \eta H_i)\}_{i \in I}) = Acc(T, D)$ for any $\eta \in N^*$ such that $max_{i \in I} |P_i \eta H_i| \leq w$ by Lemma 2.8(ii-iii).

But T'' is an arbitrary-precision transformer. It remains to show that we can derive a *fixed*-precision transformer T' from T''. Note that by definition (Equation 2.5), for any $\sigma \in (\Sigma')^*$ such that $|\sigma| < w$: $|f(\sigma)| = w(|\sigma| + 1)$. By assumption (Theorem 1), no input example—challenge or otherwise—exceeds the fixed, finite $w > max_{i \in I} |P_i H_i|$ in length. Within the assumptions of Theorem 1, it follows that the upper bound on the length of possible inputs to T'' is $w^2 + w$.

By definition, the floating-point precision of an arbitrary-precision transformer varies as a function of input length. Let $\pi \colon \mathbb{N} \to \mathbb{N}$ be the function mapping input length to floating-point precision (in bits) of T''—presumably, π is monotone-increasing, but it need not be. Define T' as T'' with floating-point precision fixed at $max_{1 \leq n \leq w^2 + w} \pi(n)$.

This completes the proof of Theorem 1.

Chapter 3

Formal-Logical Distributional Semantics (FoLDS)[1]

In this chapter, I aim to motivate the use of language models over logical forms—in particular, the larger, graph-based neural model introduced in Chapter 4. In Section 3.1, I briefly overview the fields of formal and distributional semantics, and discuss related work in merging these two representational paradigms.

I introduce the simple, prototype *Formal-Logical Distributional Semantics* (FoLDS) model in Section 3.2: a non-neural, distributional model that generates complex-valued word vectors drawn from a fuzzy-logical model world imperatively constructed from logical-form representations. I argue that the use of logical-form inputs has a syntactic de-noising effect, as such representations equivalence-class the myriad surface realizations of a given proposition. This permits models over logical forms such as FoLDS to generate meaningful representations from less data than their superficial counterparts, which must learn to equate these various periphrastic realizations.

Additionally, I posit that the explicit representation (i.e. de-noising) yielded by the function-argument structure inherent in logical forms allows for greater sensitivity to critical

[1]An abridged version of this chapter has been published in Sullivan (2023).

logical operators such as negation. In the case of FoLDS, this sensitivity to negation is encoded via its complex-valued embeddings, which permit a complex-valued similarity metric, allowing the model to leverage two axes of similarity simultaneously: antonymy/synonymy and relatedness.

I then show in Section 3.3 that FoLDS outperforms several competing superficial approaches—that require significantly more data—on a downstream task. This result—in combination with the performance of the larger, neural model of Chapter 4 (see Chapter 5)—provides converging evidence from multiple methodologies in support of the Accelerated Learning Hypothesis of Chapter 1, and towards the efficacy and practical utility of language modeling over logical forms.

3.1 Background and Related Work

There are two main approaches to the problem of formally representing the meaning of natural language utterances (Boleda and Herbelot, 2016): *formal/symbolic* (logical formulas; Section 3.1.1) and *distributional* (vector encoding; Section 3.1.2). In this section, I compare the respective advantages and drawbacks of both methods, in order to motivate *formal-distributional* semantics (Section 3.1.3), a broadly-defined research program of models—including FoLDS—that aim to consolidate these two approaches.

3.1.1 Formal Semantics

Formal semantics is a broad and interdisciplinary field that incorporates concepts from—and finds applications in—linguistics, philosophy, computer science, mathematical logic, and cognitive psychology (Lewis, 1976). For the purposes of the discussion at hand, I limit this exposition to those aspects of formal semantics that pertain to computational linguistics and NLP.

While there are multiple logical systems that serve as an underlying basis for formal

semantic representations (first/higher order predicate calculus, intuitionistic logic, etc.; Fine, 2014), all variants of formal semantic representation are designed to model the state of affairs that is described by a given utterance by mapping that utterance to explicitly defined logical formulas with interpretable truth conditions (Winter, 2016). Because natural language expressions map to sentences in a symbolic logic in formal-semantic frameworks, it is rather (conceptually, if not computationally) straightforward to compute their truth-validity (Boleda and Herbelot, 2016). Such readily-computable truth values have obvious benefits for computational linguistics/NLP tasks such as question-answering: a binary (yes/no) question is true if its corresponding logical formula is as well, the answer to a wh-question is the set of all entities, propositions, events, or situations that make that logical formula true, and so on (Lopez et al., 2007). Additionally, logical formulas have the effect of equivalence-classing syntactic paraphrases (see Section 3.2.1): an active sentence and its passive or topicalized counterparts are mapped to the same symbolic representation, etc.

These representations of meaning, however, can be difficult to employ computationally at scale: they require semanticists to construct not only an entire grammar—in particular, mappings from syntactic structures to logical representations—and a lexicon that includes all possible senses of a given word, but also a reasonably accurate set- or type-theoretic model of the entire known universe (Bos, 2011). Furthermore, there is no known way to quantitatively compare the similarity of two logical formulas. While it is possible to verify that two formulas are logically equivalent, incompatible, unrelated, or that one implies the other, a finer-grained, continuous similarity metric—such as those that arise in distributional semantics—does not exist between logical formulas.

3.1.2 Distributional Semantics

Conversely, distributional representations are designed to essentially crowd-source—and thus facilitate—the encoding of meaning by data-mining a set of documents and producing vector representations of words, sentences, and documents derived, ultimately, from co-

occurrence statistics between words and the contexts in which they appear (Lenci, 2018). These representations allow us to compute the similarity between any two word, sentence, or document vectors using metrics such as the vector dot product (i.e. cosine similarity/distance) and Euclidean distance (Aerts, Kitto, and Sitbon, 2011).

However, these encodings are superficial—i.e. directly derived from surface text—and therefore reflect only syntactic properties of the words, sentences, and documents that they represent. While patterns in the surface text can reflect semantic properties, vector embeddings do not directly correspond to the underlying meanings of the linguistic units that they represent (see e.g. Withagen et al., 2012; Jones et al., 2022). A direct consequence of this fact is that the similarity metrics that accompany these representations purely measure *textual*—rather than truly semantic—similarity. Another potentially significant drawback to distributional approaches is that there is no known way to compute the truth value of a meaning vector (Boleda and Herbelot, 2016), which presents a potential impediment to distributional models' ability to perform inferencing and logical-reasoning tasks: for example, Borji (2023) shows that ChatGPT is prone to logical and factual errors, such as insisting that it is not possible to know what the gender of the first female President of the United States will be.

3.1.3 Formal-Distributional Semantics

The framework of Formal-Distributional Semantics (FDS) encompasses a broad range of approaches to fusing formal and distributional semantics, with the goal of addressing the respective drawbacks of formal and distributional approaches to natural language semantics, and unifying their advantages (Boleda and Herbelot, 2016). FDS can be divided into two subfields: *F-first* FDS, in which formal logic constitutes the basis of the semantic framework and distributional representations are incorporated by licensing inferences between distributionally similar terms, and *D-first* FDS, in which logical operators are conceived as (multi-)linear mappings that act directly on distributional representations (Boleda and

Herbelot, 2016).

Erk (2016). In one such implementation of F-first FDS, Erk (2016) uses distributional and formal semantics in order to model speakers' probabilistic information states. The central notion is that human beings have (at least) two categories of world knowledge: that which has been directly observed and/or stated, and that which has been inferred probabilistically from the first category, using our knowledge of the similarity of word usage in linguistic contexts.

Erk (2016) first uses the McRae et al. (2005) *feature norm database* to generate a model world G. A feature norm database (see Section 3.3.1) consists of a set of *concepts* (words) and a set of *features*, in which each concept w is assigned a feature vector $F(w) \in \mathbb{R}^n$, where n is the number of features in the database. The value of $F(w)_Q$ is the value of the feature Q for the word w. In the McRae et al. (2005) database, feature values are obtained from experiment participants' judgments, so that $F(w)_Q$ represents the proportion of participants (out of thirty) who said that the word w has the property Q. Note that in this database, properties are elicited: the participants were asked to list all of the properties that come to mind for each word (as opposed to choosing relevant properties from a predefined list).

Using the words and feature values from the McRae et al. (2005) database, Erk's (2016) model G is generated as follows: for each word w, there are thirty entities x such that $w(x)$ holds in G. For each property Q, there are $F(w)_Q \cdot 30$ entities x such that $w(x) \wedge Q(x)$ holds in G. For example, $F(\textit{knife})_{\textit{utensil}} = 0.634$ (19/30), so there are 30 entities x such that $\textit{knife}(x)$ holds in G and, of those 30 entities, there are 19 such that $\textit{utensil}(x)$ also holds in G.

Erk (2016) then uses linear regression to infer the best relationship between property overlap—i.e. overlap of extensions—in G and distributional similarity for any two words w and u. In this framework, a speaker's information state is modeled as a set of randomly-generated first-order logical (FOL) models U. For each model $M \in U$, Erk (2016) uses the above-mentioned linear regression model to perform a Bayesian update, which yields a

probability $P(M)$: the speaker's belief that M is the "real world". Given a proposition ϕ, the speaker's belief $\mathcal{P}(\phi)$ that ϕ is true in the "real world" is defined in Equation 3.1 below.

$$\mathcal{P}(\phi) = \sum_{M \in \{M' \in U \mid [[\phi]]_{M'}\}} P(M) \tag{3.1}$$

In words: $\mathcal{P}(\phi)$ is equal to the sum of the probabilities of all model worlds $M \in U$ such that ϕ is true in M.

However, due to the fact that all of the probabilities mentioned above are defined in terms of traditional distributional word vectors, they are still affected by the noise resulting from those embeddings' shortcomings: insensitivity to negation, word order, syntactic paraphrases, affordances, and so on (see e.g. Withagen et al., 2012; Ettinger, 2020; Chaves and Richter, 2021; Jones et al., 2022). Additionally, this framework can only generate inferences regarding universally-quantified statements involving unary predicates.

Herbelot and Copestake (2021). In another approach to F-first FDS, Herbelot and Copestake (2021) eschew traditional distributional embeddings generated from surface text in corpora, and instead employ vector representations generated directly from a logical model world M. In this framework, each word (predicate) w is represented by a vector $v(w) \in \mathbb{R}^n$, where n is the number of entities in the domain. For each $1 \leq i \leq n$, the value of $v(w)_i = 1$ if $w(e_i)$ is true in M of the i^{th} entity e_i, and $v(w)_i = 0$ otherwise. Basic logical operators (negation, conjunction, disjunction, etc.) can then be defined in terms of operations on these vector representations. Herbelot and Copestake (2021) find that these logical embeddings are highly effective at modeling elementary semantic relations such as hyponymy, synonymy, and antonymy. For example, let $v \odot w$ denote the Hadamard product (Kim et al., 2016) between two vectors v and w, where $(v \odot w)_i = v_i \cdot w_i$. Then v is a hyponym of w if and only if $v \odot w = v$.

One significant drawback to Herbelot and Copestake's (2021) approach is that it can only be implemented with respect to a so-called "ideal" model M. The binary (0/1) logical

vectors that this approach employs are incompatible with the noisy and contradictory nature of real-world corpus data. As such, the authors are only able to employ this method with a toy-example, hand-constructed model M.

Venhuizen et al. (2022). Similarly, Venhuizen et al. (2022) introduce an F-first FDS model in which *propositions*—rather than predicates—are represented as binary vectors. These authors employ a constraint-based generation process to generate a set of K model worlds $\{\mathcal{M}_i\}_{1\leq i\leq K}$: for each proposition ϕ with corresponding embedding $v(\vec{\phi})$, the i^{th} coordinate $v(\vec{\phi})_i = 1$ if and only if ϕ holds in $\mathcal{M}_i$ (otherwise, $v(\vec{\phi})_i = 0$). Venhuizen et al. (2022) then train a Simple Recurrent Neural Network (SRN; Elman, 1990) over these carefully-crafted embeddings, yielding a model that can generate probabilistic embeddings over unseen propositions.

However, Venhuizen et al.'s (2022) model still suffers from many of the same drawbacks as that of Herbelot and Copestake (2021). Although their use of an SRN yields a more flexible model, a set of hand-crafted constraints is still needed to generate the set of worlds $\mathcal{M}_i$: the SRN model can only generalize to unseen *propositions*, not to unseen predicates and entities—in order to incorporate new predicates and entities into this model, it is necessary to hand-craft additional relevant constraints for model-world generation.

Grefenstette et al. (2014). In Grefenstette and Sadrzadeh (2011), the authors construct a compositional model for distributional semantics (D-first FDS). These authors use traditional distributional embeddings to model word meanings, which are then composed together to form sentence vectors. In this paradigm, a categorial grammar (Steedman, 1993) provides the syntactic combinatorics that guide semantic composition in a parallel, step-by-step fashion. This is similar to the semantic combinatorics typically employed by categorial grammars such as Hybrid Type-Logical Categorial Grammar (Kubota, 2010; Kubota and Levine, 2020), the key difference being that distributional vectors, rather than classical logical terms, provide the underlying lexical semantics.

This framework draws heavily on concepts from quantum physics: for example, a sentence vector is modeled as a superposition of its constituent words, in the sense that each word vector is viewed as a quantum state, and the sentence is a mixture (i.e. tensor product; Marcus and Moyls, 1959) of these states.

While theoretically interesting, this model requires many of its components to be defined by hand, and the advantages that such compositional sentence vectors provide over those generated by a neural LM such as BERT (Devlin et al., 2019) remain unclear.

Emerson (2018). The D-first FDS framework of Emerson (2018), employs neural models that take Dependency Minimal Recursion Semantics (DMRS; Copestake et al., 2016) situation graphs as input, and return probability distributions over predicates in the domain.

Take, for example, the DMRS situation graph in Figure 3.1, where X and Z denote entities, and Y denotes an event variable. The individuals (i.e. entities and events) X, Y, and Z are represented by randomly-sampled, binary-valued individual embeddings. Each edge label (e.g. *ARG1*, *ARG2*) is represented by a Cardinality-Restricted Boltzmann Machine (Swersky et al., 2012) that learns a joint distribution over individuals: for example, the probability $P_{ARG1}(X, Y)$ that the entity X occurs as the first-place argument of the event Y.

The model then uses one-layer, logistic feed-forward networks over these entity embeddings to yield three probability distributions $T_{r,X}$, $T_{r,Y}$, and $T_{r,Z}$ for each predicate r, corresponding to the probabilities that $r(Y, –, Z)$, $r(–, X, Z)$, and $r(Y, X, –)$ are true of X, Y, and Z, respectively.

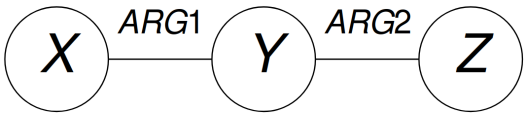


Figure 3.1: DMRS situation graph.

This method achieves competitive results on lexical and contextual similarity tasks, and uses significantly less training data than language models such as BERT and GPT-3. There are, however, a few shortcomings to this approach. In particular, it can only account for

simple sentences—i.e. those without relative clauses or adjuncts—involving intransitive or transitive verbs, and requires a "precisely annotated", hand-built corpus of DMRS structures for its training procedure.

3.2 The FoLDS Model

In this section, I discuss both the motivation for and architecture of the FoLDS model, which generates complex-valued word vectors—permitting a two-dimensional similarity metric (see Section 3.2.5)— drawn from a fuzzy-logical model world imperatively constructed from logical formulas via a recursive procedure (Section 3.2.4). To obtain these logical formulas, I first parsed a set of documents (discussed in Section 3.3.3) into Minimal Recursion Semantics (MRS; Copestake et al., 2005) representations (Section 3.2.2). However, due to their underspecified nature, MRS structures are not suitable for constructing a model world, necessitating their conversion into an intermediate representation, which I describe in Section 3.2.3.

The FoLDS model is primarily intended to be used for (quasi-)logical inference: the key idea behind this model is to consider *logical* distributional co-occurrence in order to capture and predict *logical* properties—as opposed to traditional distributional models, which consider *textual* distributional co-occurrence in order to capture and predict *textual* properties—while avoiding the brittle nature of similar models (e.g. Herbelot and Copestake, 2021; see Section 3.1.3) that prohibits their use with real-world data.

3.2.1 Distributional Semantics over Logical Forms

As discussed in Chapter 1, I argue that co-occurrence statistics should be equivariant with respect to syntactic paraphrases—non-canonical constructions such as passivization, topicalization, etc. (Colin and Gardent, 2018)—in order to more efficiently capture meaning via distributional methods. For example, the active sentence in Example 1a should belong to the same equivalence class as its passive counterpart in Example 1b.

Example 1.

(a) Tahitian natives feasted on __

(b) __ *was feasted on by Tahitian natives*

Clearly, Examples 1a-b are *not* the same context for a superficial distributional model. On the other hand, converting both examples to an FOL-typed λ-calculus representation yields the exact same formula: namely, $\lambda x.\textit{feast-on}(\textit{tahitian-natives}, x)$.

Intuitively, treating logical formulas as distributional contexts should decrease the amount of training data required to obtain accurate embeddings, as the larger amounts of data required to learn to equate passive constructions and their active counterparts are no longer necessary (c.f. the Accelerated Learning Hypothesis of Chapter 1). Additionally, Herbelot and Copestake (2021) demonstrate that distributional embeddings obtained from logical-form descriptions of model-theoretic representations of events and situations are highly effective at modeling elementary semantic relations such as hyponymy, synonymy, and antonymy (as discussed in Section 3.1.3).

FoLDS therefore constructs embeddings from logical-form representations of sentences, rather than directly from surface text—as in Emerson (2018) and Herbelot and Copestake (2021). Unlike the models introduced by those authors, however, FoLDS utilizes an automated process to generate these logical representations, which greatly increases its potential for scalability.

I additionally argue that negation carries a unique function in natural language semantics: as seen in Chapter 2, superficial distributional models struggle to model this function. To illustrate this point, consider the two sentences in Examples 2a-b: this pair of sentences *increases* the superficial distributional similarity between *herbivore* and *carnivore*—many of the words in those two sentences are the same—when in fact they are antonymous. Superficial LMs are known to be insensitive to negation (see Chapter 2) and word order (Ettinger, 2020), and so these models are inherently less capable of detecting the evidence of antonymy between

herbivore and *carnivore* that Examples 2a-b contribute than a negation-sensitive architecture such as FoLDS.

Example 2.

(a) Alligators are not ever considered herbivores, even when food is scarce

(b) Alligators are always considered carnivores, even when food is scarce

While Examples 2a-b suggest that *herbivore* and *carnivore* are antonymous, they do not indicate that the terms are *unrelated*. Inferences can still be drawn from antonymous but related categories: for example, given that *carnivore* has a property such as *eats-meat*, a model should be able to leverage the antonymy between *carnivore* and *herbivore* to hypothesize that *herbivore* does not have that property.

There is a crucial distinction to be drawn between synonymy/antonymy and relatedness. Unrelated categories should not contribute to the inference of properties: for example, properties of *duct tape* should have no bearing on the estimation of properties of *alligator*.

Examples 2a-b demonstrate that negation is crucial for detecting antonymy, and the above two points suggest the utility of a two-dimensional similarity metric that measures both synonymy/antonymy and relatedness. The de-noising effect of the function-argument structure of logical forms provides the means to systematically detect negation and incorporate it into the model's embeddings: unlike other logical operators such as conjunction, disjunction, implication, and quantifiers, I do not remove negation from the logical structures described in Sections 3.2.2 and 3.2.3 below, in order to better capture antonymy and facilitate the two-dimensional FoLDS similarity metric.

3.2.2 From Textual to MRS Representations

The first step towards generating the logical formulas that serve as distributional contexts in the FoLDS model is to parse the raw text into Minimal Recursion Semantics (MRS; Copestake et al., 2005) representations, and augment these MRS structures with additional

information such as coreference data (3.2.2.2). In Section 3.2.2.1, I provide a brief overview of the structure of MRS representations in order to facilitate understanding of the remainder of Section 3.2: I refer interested readers to Copestake et al. (2005) for a more in-depth description of MRS and its applications.

3.2.2.1 Minimal Recursion Semantics (MRS)

MRS is a framework for representing natural language semantics in a computationally-tractable and underspecified manner. In this framework, elementary predications (EPs)—i.e. predicates and quantifiers—are organized into implicitly-conjoined lists that are referred to as *handles*. For example, the handle h_1: $big(x), white(x), horse(x)$ has the label h_1, and is interpreted in FOL as $big(x) \wedge white(x) \wedge horse(x)$. Quantifier scope in MRS is underspecified, meaning that, for example, the MRS representation corresponding to "*every student read a book*" in Equation 3.2 can be translated to FOL as either $\forall x[student(x) \rightarrow \exists y[book(y) \wedge read(x, y)]]$ or $\exists y[book(y) \wedge \forall x[student(x) \rightarrow read(x, y)]]$.

$$\begin{aligned} &h_1\colon every(x, h_2, h_3) \\ &h_4\colon a(y, h_5, h_6) \\ &h_7\colon student(x) \\ &h_8\colon read(x, y) \\ &h_9\colon book(x) \\ &h_2 =_q h_7, h_5 =_q h_9 \end{aligned} \tag{3.2}$$

In the MRS structure in Equation 3.2, $every(x, h_2, h_3)$ indicates that x is the bound variable, h_2 is the restriction, and h_3 is the nuclear scope of the quantifier *every*. Note the flat structure of this representation: no EP or handle directly scopes over another, and the entire structure is simply a list of handles and their corresponding EPs.

Although h_2 is the restriction of the quantifier EP *every*, there are no EPs that belong to the handle h_2—clearly, h_7 should be the restriction of *every*. To account for this, h_2 and h_7 are linked via the equality modulo quantifiers (*qeq*) relation: $h_2 =_q h_7$. This relation indicates that h_2 should be treated as equivalent to h_7 for the purpose of determining quantifier scope, but that other quantifiers may intervene between h_2 and h_7.

To motivate the use of the *qeq* relation in MRS, take, for example, the sentence "*every nephew of some politician runs*". There are two possible representations of this sentence in FOL: $\forall x[\exists y[politician(y) \wedge nephew(x,y)] \rightarrow run(x)]$ or $\exists y[politician(y) \wedge \forall x[nephew(x,y) \rightarrow run(x)]]$. In MRS, this sentence is represented as in Equation 3.3.

$$\begin{aligned} &h_1 \colon every(x, h_2, h_3) \\ &h_4 \colon some(y, h_5, h_6) \\ &h_7 \colon politician(y), nephew(x,y) \\ &h_8 \colon run(x) \\ &h_2 =_q h_7, h_5 =_q h_7 \end{aligned} \tag{3.3}$$

Note that $politician(y)$ and $nephew(x,y)$ both belong to the handle h_7, and that $h_2 =_q h_7$ and $h_5 =_q h_7$: *every* and *some* both scope over h_7, but either one can scopally intervene between h_7 and the other.

In MRS, all logical operators are represented as EPs. For example, "*Sally sees Mary or John*" is (roughly) represented in MRS with the EPs $see(sally, x)$ and $or\_c(x, mary, john)$—i.e. x is the variable corresponding to either *mary* or *john*. The sentence "*Sally sees Mary and John*" is represented in a similar fashion, but with the EP *and_ c* in place of *or_ c*. To account for the scopal ambiguity inherent in negation constructions[2], negation (expressed by the EP $neg(-)$) scope is left underspecified, in the same manner as for quantifiers.

[2]For example, the sentence "*all of the students didn't read Harry Potter*" has two possible interpretations: $\neg\forall x[student(x) \rightarrow read(x, harry\text{-}potter)]$ or $\forall x[student(x) \rightarrow \neg read(x, harry\text{-}potter)]$.

3.2.2.2 Parsing and Coreference Alignment

Turning to the discussion of the FoLDS pipeline, I first passed each document in the corpus (see Section 3.3.3) through the Spacy *NeuralCoref*[3] coreference resolution module, which returns a set of *instances*: 4-tuples (c, s, e, m) denoting referring expressions, where c is the *coreference cluster* index—all referring expressions that refer to the same discourse entity belong to the same coreference cluster. The middle two values, s and e, are the *start* and *end* character indices (respectively), and indicate the span of text corresponding to the referring expression in question. The last element, m, is a binary (true/false) value that indicates whether the instance is a *main instance*: usually the most specific mention of the entity in the coreference cluster—e.g. a proper name, NP modified by a relative clause, etc.— that serves as the anchor that all other instances in the cluster refer to.

Following coreference resolution, I then passed the documents through the Spacy *SentenceRecognizer*[4] sentence-segmentation pipeline and applied the ACE ERG (Copestake and Flickinger, 2000) parser[5] to each sentence to obtain its MRS representation. Each quantifier-type EP in the MRS structure was then assigned a coreference cluster, as determined by NeuralCoref.

I used a heuristic procedure to identify the most likely MRS quantifier that each referring expression in a coreference cluster corresponds to: as a neural model, NeuralCoref is somewhat noisy, and so the referring expressions that it identifies do not necessarily align one-to-one with the NPs identified by the ERG parser. Each MRS EP contains a *lnk* value: a pair (s, e) denoting the *start* and *end* character indices of the span of text corresponding to that EP. I assigned to each quantifier Q an *expanded lnk* value (s_Q, e_Q), denoting the left-most start and right-most end indices (respectively) of all EPs contained within the restriction of Q. The resulting expanded pair (s_Q, e_Q) roughly corresponds to the entire NP that constitutes the restriction of the quantifier Q.

[3]https://spacy.io/universe/project/neuralcoref
[4]https://spacy.io/api/sentencerecognizer
[5]ERG-1214 release: https://github.com/delph-in/erg

To align NeuralCoref instances with MRS quantifiers, I developed and employed the *pseudo-Euclidean distance* ($Dist_{PE}$) metric defined in Equation 3.4.

$$Dist_{PE}((s_1, e_1), (s_2, e_2)) = \sqrt{(s_1 - s_2)^2 + |e_1 - e_2|} \tag{3.4}$$

Pseudo-Euclidean distance is defined similarly to the standard Euclidean distance, but places more weight on the left-hand side (the start indices s_1, s_2): this is because English noun phrases (NPs) tend to have their heads closer to the left-hand edge. Given a referent identified by NeuralCoref that starts at character s_1 and ends at character e_1, and a quantifier span identified by ERG that starts at character s_2 and ends at character e_2, the primary factor in determining the pseudo-Euclidean distance between the two is the difference between s_1 and s_2. The difference between e_1 and e_2 essentially serves as a tie-breaker between referring expressions that were determined by NeuralCoref to have similar start indices.

To illustrate this process, consider the sentence in Example 3, where the brackets denote NP referents, and the underlined portions indicate spans of text identified by NeuralCoref as belonging to some coreference cluster c.

Example 3.

[the dog <u>who loved [the boy] and</u> [the girl]] chased <u>his</u> tail

In Example 3, the underlined span $S =$ "*who loved the boy and*" and "*his*" both belong to the same coreference cluster, but NeuralCoref has erroneously truncated the span. Note that the span $D =$ "*the dog who loved the boy and the girl*" starts at character 0 and ends at character 38, the span $B =$ "*the boy*" starts at character 18 and ends at character 25, and S starts at character 8 and ends at character 29. Using Euclidean distance, $Dist_E(S, D) = 12.04$ and $Dist_E(S, B) = 10.77$: the boy (B) is closer to S than the dog (D) and will therefore be (incorrectly) assigned to the coreference cluster c. However, with *pseudo*-Euclidean distance, $Dist_{PE}(S, D) = 8.54$ and $Dist_{PE}(S, B) = 10.2$, correctly identifying S with D.

I then used the Hungarian Matching algorithm (Kuhn, 1955) to find the optimal pairing of

NeuralCoref referring expressions with MRS quantifier spans, with respect to pseudo-Euclidean distance.

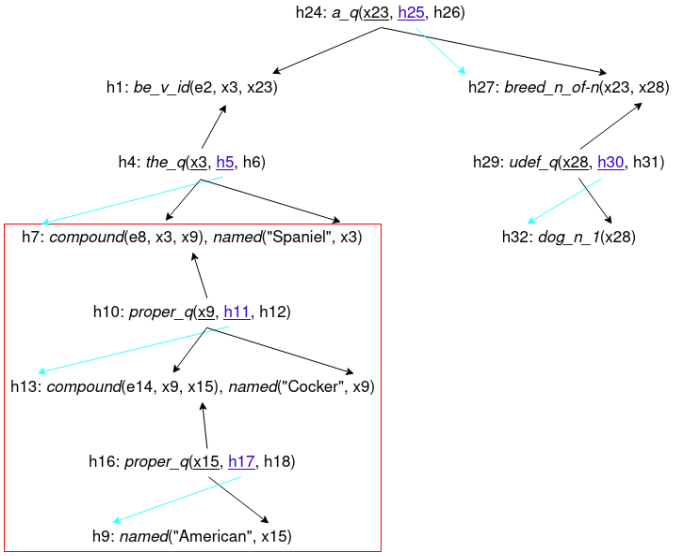


Figure 3.2a: MRS structure corresponding to *"the American Cocker Spaniel is a breed of dog"* before compound resolution. The *compound structure* corresponding to the compound noun *"American Cocker Spaniel"* is highlighted with a red box. The blue arrows indicate handle *qeq* relations, and the black arrows serve as a visual aid to indicate instances of the same variable.

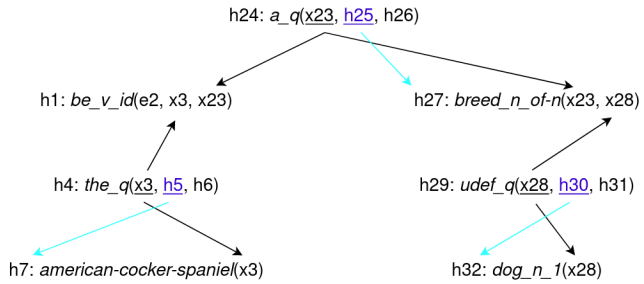


Figure 3.2b: MRS structure corresponding to *"the American Cocker Spaniel is a breed of dog"* after compound resolution.

After aligning the coreference data, I removed compound structures—MRS sub-structures denoting compound words (see Figure 3.2a)—from the MRS representations and replaced them with hyphenated concatenations of the predicate labels of their constituent parts (see Figure 3.2b).

3.2.3 From MRS to Pseudo-MRS

In order to obtain logical representations that can serve as distributional contexts, FoLDS removes all quantifiers and other logical operators from the MRS formulas, while preserving as much information from these operators as possible. To accomplish this, it was first necessary to convert MRS structures to *pseudo*-MRS (PMRS), a representation that is more similar to FOL. The primary motivation for the conversion to PMRS is the fact that scope is left underspecified in MRS: without first determining their scopal ordering, it is not feasible to remove quantifiers and other logical operators from the formulas while preserving meaningful information.

3.2.3.1 MRS Preprocessing

To derive PMRS structures from MRS, I first removed event-variable arguments from the EPs and discourse-related EPs (e.g. *parg_ d*) from the MRS structures. I then removed copular EPs: for example, a sentence such as "*John is a student*" corresponds to the FOL formula $student(john)$, but its MRS representation is along the lines of $proper\_q(x) \wedge named(x, "john") \wedge be\_v\_id(x,y) \wedge a\_q(y) \wedge student(y)$. To remove the copular $be\_v\_id(x,y)$ EP from the MRS structure, I replaced all instances of y with x and removed the quantifier binding y (along with $be\_v\_id(x,y)$) from the formula.

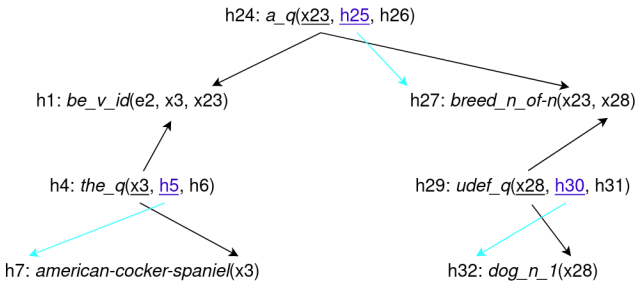


Figure 3.3a: MRS structure corresponding to *"the American Cocker Spaniel is a breed of dog"* before copula resolution (duplicated from Figure 3.2b). The blue arrows indicate handle *qeq* relations, and the black arrows serve as a visual aid to indicate instances of the same variable.

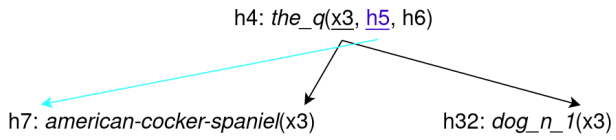


Figure 3.3b: MRS structure corresponding to *"the American Cocker Spaniel is a breed of dog"* after copula resolution.

I additionally treated certain EPs with a copular function, such as *breed_of*, *kind_of*, *type_of*, *species_of*, etc. as instances of *be_v_id*, and removed them in an identical fashion (see Figure 3.3). For example, if there is a sentence such as "*the Cocker Spaniel is a breed of dog*", then—ignoring quantification—the corresponding MRS structure is roughly expressed as: $\textit{cocker-spaniel}(x_1) \wedge \textit{be\_v\_id}(x_1, x_2) \wedge \textit{breed\_of}(x_2, x_3) \wedge \textit{dog}(x_3)$. Without treating *breed_of* as an instance of *be_v_id*, FoLDS would instantiate an entity e_n such that $\textit{dog}(e_n)$ and $\textit{breed\_of}(x, e_n)$ holds for all x such that $\textit{cocker-spaniel}(x)$ (see Section 3.2.4). Given another sentence such as "*the Golden Retriever is a breed of dog*", the same procedure will apply with another entity $e_m \neq e_n$ such that $\textit{dog}(e_m)$ and $\textit{breed\_of}(x, e_m)$ holds for all x such that $\textit{golden-retriever}(x)$. In this scenario, the fact that Golden Retrievers and Cocker Spaniels are both breeds of dog does not lead to meaningful overlap between the two predicates: there is no entity instantiated that is both a dog and a Golden Retriever (or a dog and a Cocker Spaniel). By treating *breed_of* as an instance of *be_v_id*, FoLDS does admittedly lose semantic data encoded in the MRS structure, but all Cocker Spaniels and Golden Retrievers x will directly receive the property $\textit{dog}(x)$, leading to more meaningful overlap of properties between these predicates.

I then removed conjunction and disjunction from the MRS structures, by simply duplicating each MRS formula ϕ containing an *and_c* or *or_c* EP, and instantiating two new formulas ϕ_L and ϕ_R: copies of ϕ with the left- and right-hand conjuncts removed, respectively. For example, let $MRS(S)$ denote the MRS representation of an English sentence S, and let $\phi = MRS(\textit{"everyone likes cats and dogs"})$. Then $\phi_L = MRS(\textit{"everyone likes dogs"})$ and $\phi_R = MRS(\textit{"everyone likes cats"})$. This process continued recursively until all *and_c* and

or_ c EPs were removed from each MRS structure.

Although FoLDS does ignore (i.e. lose) semantic data by treating *and_ c* and *or_ c* in the same manner, it is not apparent that a more effective approach is possible within the limitations of this model. There are two obvious alternatives to the treatment of logical disjunction laid out above: first, given a proposition $\phi(x) \vee \psi(x)$, we could randomly select one of the two conjuncts to assign to x (ϕ or ψ)—this approach would still result in a loss of semantic data, due the deletion of one of the disjuncts. We could also weigh each disjunct with a weight of $1/2$—i.e. treat $\phi(x) \vee \psi(x)$ as half of an occurrence of $\phi(x)$ and half of an occurrence of $\psi(x)$. Arguably, there is not much of a difference between this method and that laid out above (i.e. treating *and_ c* and *or_ c* in the same manner), and it is difficult to draw parallels between this method and intuitive notions about the natural-language semantics of the word "or".

3.2.3.2 Pseudo-MRS (PMRS)

In order to convert MRS structures into the more FOL-like PMRS structures, I sorted all quantifier EPs into one of three categories: *universal* ($\forall$; e.g. *all_ q*, *every_ q*), *existential* ($\exists$; e.g. *some_ q*, *a_ q*), and *coreferential* (Π). All MRS quantifiers that are not the main instance of a coreference cluster are mapped to Π. For example, take the span of text in Example 4, where the subscript c indicates that *all men*$_c$ and *they*$_c$ belong to the same coreference cluster c.

Example 4.

All men$_c$ *are mortal. For this reason,* *they*$_c$ *are unhappy.*

In Example 4, *all men*$_c$ is the main referent and *they*$_c$ is an instance of c. The sentence "*all men*$_c$ *are mortal*" is mapped to a PMRS structure resembling $\forall^c x[man(x) \rightarrow mortal(x)]$, which carries the same interpretation as the PMRS structure $\forall x[man(x) \rightarrow mortal(x)]$, but contains additional information indicating that the restriction of the quantifier (in this case, $man(x)$) corresponds to the span of text identified by NeuralCoref as the main referent of

the coreference cluster c. The clause "$\underline{they_c}$ *are unhappy*" is mapped to a PMRS structure resembling $\Pi^c y[unhappy(y)]$: when the anaphora is resolved (see Step 5 of Section 3.2.4), $\Pi^c y[unhappy(y)]$ will be converted to $\forall x[man(x) \rightarrow unhappy(x)]$.

Each PMRS structure ϕ consists of a list ϕ_Q of quantifiers, a list ϕ_P of (implicitly conjoined) predicates, and a weight $\phi_W \in (0, 1]$ that encodes the degree of evidence towards the validity of ϕ: in most cases, ϕ_W will be equal to one—ϕ_W is only less than one in the case that ϕ arose from a $\forall$-type PMRS structure ψ (see Step 4 of Section 3.2.4). Each quantifier contains indices denoting its coreference cluster, bound variable, and the list of predicates constituting its restriction. The ordering of ϕ_Q determines the scopal relations between the quantifiers in the PMRS structure: a quantifier q outscopes a quantifier q' if and only if q precedes q' in the list ϕ_Q (see Figure 3.4b).

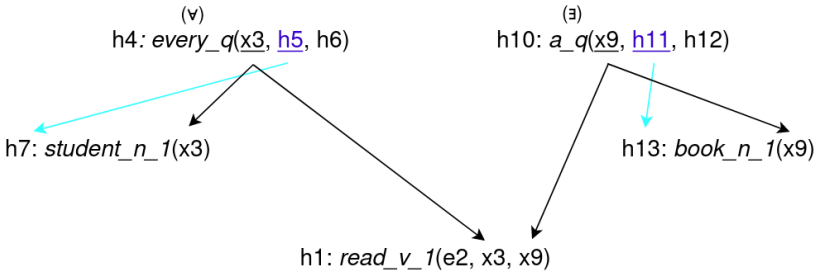


Figure 3.4a: MRS structure corresponding to *"every student reads a book"*. The symbols in parentheses above the MRS quantifiers (*every_q* and *a_q*) indicate the PMRS quantifier categories that they will be sorted into.

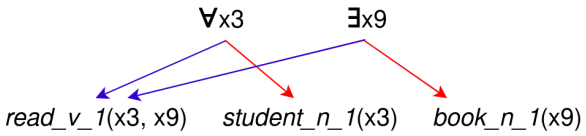


Figure 3.4b: PMRS structure corresponding to the $\forall > \exists$ interpretation of *"every student reads a book"*. The red arrows indicate quantifier restriction, and the blue arrows nuclear scope. Note that the restriction of each quantifier corresponds to its restriction handle in the MRS structure (see Figure 3.4a), and its nuclear scope consists of all predicates not in its restriction that contain an instance of its bound variable.

Each predicate consists of a list of arguments, a label (the name of the predicate, e.g. *dog_n_1*, *run_v_1*, *see_v_1*, etc.), and a boolean polarity value that indicates whether the

predicate is negated. I placed the PMRS structures into negation normal form (Darwiche, 2001): negation operators were recursively percolated downward until they scoped over individual predicates, using the rules of logical replacement in Equation 3.5 below.

$$\neg\forall x[P] = \exists x[\neg P] \tag{3.5a}$$

$$\neg\exists x[P] = \forall x[\neg P] \tag{3.5b}$$

In the following section (3.2.4), I often denote PMRS structures using their rough FOL equivalents for the sake of expositional simplicity. In those cases, I indicate this notational abuse with the "$\approx$" symbol: for example, (PMRS) $\phi \approx \forall x[student(x) \rightarrow \exists y[book(y) \wedge read(x, y)]]$ to denote the PMRS structure in Figure 3.4b.

3.2.4 From Pseudo-MRS to a Fuzzy-Logical Model World

I then sorted each PMRS structure ϕ into one of four categories: $\exists$-type, in which the highest-scoping quantifier $\phi_Q[0]$ is existential; $\forall$-type, in which $\phi_Q[0]$ is universal; Π-type, in which $\phi_Q[0]$ is coreferential; and *non-quantified*, in which the quantifier list ϕ_Q is empty. I initialized the weight ϕ_W all of PMRS structures ϕ to 1.0, and used the following procedure (Steps 1-6 below) to remove all logical operators from the PMRS structures and obtain a list of $(p, c(p)_+, c(p)_-)$ triples, where p is an *atomic proposition* (a predicate and its arguments), $c(p)_+$ is the number of times that p occurs in the data (the *positive* count), and $c(p)_-$ the number of times that $\neg p$ occurs in the data (the *negative* count).

Step 1. The integer-valued count $d = 0$, which is used to index variables inserted by existential quantifiers (see Step 3), and the *coreference store*—a list of pairs (c, x), where c is the index of a coreference cluster, and x is either an entity or a PMRS structure (see Step 5)—are intialized. For each named entity e belonging to a coreference cluster c, the pair (c, e)

is added to the coreference store—no other pairs are added to the coreference store at this point.

Now take, for example, the following pair of ∀-type PMRS structures: $\phi \approx \forall x[man(x) \rightarrow mortal(x)]$ and $\psi \approx \forall y[mortal(y) \rightarrow \neg happy(y)]$. If ψ were to be processed before ϕ, then every entity e such that $man(e)$ is true would not receive the property $\neg happy(e)$, because e has not yet received the property $mortal(e)$: structure ϕ must be processed before structure ψ, so that each entity e such that $man(e)$ is true receives the property $mortal(e)$, and can then receive the property $\neg happy(e)$.

For this reason, the ∀-type PMRS structures are inserted into a partial order, such that $\forall x[P(x) \rightarrow R(x)] \leq \forall y[R(y) \rightarrow Q(y)]$—i.e. given two ∀-type PMRS structures ϕ and ψ, $\phi \leq \psi$ if and only if the scope of $\phi_Q[0]$ is equal[6] to the restriction of $\psi_Q[0]$. The terminal nodes—∀-type PMRS structures ϕ such that there does not exist a ∀-type PMRS ψ with $\phi \leq \psi$—will be processed first in the remainder (Steps 2-6) of this procedure.

Step 2. For each non-quantified PMRS ϕ, the procedure iterates over each predicate p in ϕ_P, and updates the triple $(p, c(p)_+, c(p)_-)$ in the atomic proposition list as in Equation 3.6.

$$(p, c(p)_+, c(p)_-) \leftarrow \begin{cases} (p, c(p)_+ + \phi_W, c(p)_-) & \text{if } polarity(p) = \top \\ (p, c(p)_+, c(p)_- + \phi_W) & \text{otherwise} \end{cases} \tag{3.6}$$

This is to say that the weight ϕ_W (degree of attestation; see Step 4) of the PMRS ϕ is added to the *negative* count if p is negated, and ϕ_W is added to the *positive* count if p is not negated.

For example, let $\phi_P = [\neg see(john, mary)]$ and $\phi_W = 1$, and suppose that the tuple $(see(john, mary), 1, 0)$ is already in the atomic proposition list. Then following Step 2, $(see(john, mary), 1, 0) \leftarrow (see(john, mary), 1, 1)$: the existing positive count remains unchanged, and ϕ_W is added to the negative count.

[6]Although strict equality may be an excessively rigid requirement from a theoretical perspective, it was necessary for reasons of computational feasibility.

Step 3. For each ∃-type PMRS ϕ, the bound variable of $\phi_Q[0]$ is replaced with the dummy entity e_d, and $d \leftarrow d+1$. If $\phi_Q[0]$ belongs to a coreference cluster c, then the pair (c, e_d) is added to the coreference store, $\phi_Q[0]$ is removed from ϕ_Q, and ϕ_Q is re-sorted into the ∃-type, ∀-type, Π-type, or *non-quantified* categories, as determined by the type of the new $\phi_Q[0]$.

For example, let $\phi \approx \exists x \forall y[love(x,y)]$. Then following Step 3, $\phi \leftarrow \forall y[love(e_d,y)]$, $d \leftarrow d+1$, and ϕ is recategorized as a ∀-type PMRS, as ∀ is now the highest-scoping quantifier.

As another example, let $\phi \approx \exists x \exists y[love(x,y)]$. Following Step 3, $\phi \leftarrow \exists y[love(e_d,y)]$ and $d \leftarrow d+1$. After another application of Step 3, $\phi \leftarrow love(e_d, e_{d+1})$ and ϕ is recategorized as a *non-quantified* PMRS.

Step 4. For each terminal node, ∀-type PMRS ϕ, the procedure first iterates over every entity e in the domain and evaluates the truth value $w = [[rstr(\phi_Q[0])(e)]]$ (Equation 3.7) of $rstr(\phi_Q[0])(e)$, where $rstr(\phi_Q[0])(e)$ denotes the restriction of the quantifier $\phi_Q[0]$ with the bound variable of $\phi_Q[0]$ replaced by e—i.e. $rstr(\phi_Q[0])(e) = rstr(\phi_Q[0])[bvar(\phi_Q[0]) \Rightarrow e]$.

$$[[\phi]] = \begin{cases} \min\limits_{e} [[\phi[bvar(\phi_Q[0] \Rightarrow e)]]] & \text{if } \phi \text{ is ∀-type} \\ \max\limits_{e} [[\phi[bvar(\phi_Q[0] \Rightarrow e)]]] & \text{if } \phi \text{ is ∃-type} \\ \min\limits_{e \in \phi_P} [[p]] & \text{if } \phi \text{ is } \textit{non-quantified} \\ \begin{cases} 1 - [[\phi]] & \text{if } \phi \text{ is negated} \\ c(\phi)_+ / (c(\phi)_+ + c(\phi)_-) & \text{otherwise} \end{cases} & \text{if } \phi \text{ is atomic formula} \end{cases} \tag{3.7}$$

As shown in Equation 3.7, the truth value of any PMRS ψ is calculated via a recursive procedure. If ψ is a ∀-type PMRS such as $\psi \approx \forall x[dog(x) \rightarrow \exists y[tail(y) \wedge have(x,y)]]$, we iterate over all entities e in the domain and calculate $1 - min([[dog(e)]], 1 - [[\exists y[tail(y) \wedge have(e,y)]]])$: in many treatments of fuzzy logic, $\neg P$ is defined as $1-P$, and $P \wedge R$ is defined as $min(P,R)$ (Zadeh, 1965), and so $1 - min(P, 1-R) = \neg(P \wedge \neg R) = P \rightarrow R$,

therefore $1 - min([[dog(e)]], 1 - [[\exists y[tail(y) \wedge have(e, y)]]])$ is the fuzzy-logical equivalent of $\text{dog}(e) \rightarrow \exists y[tail(y) \wedge have(e, y)]$.

Under a finite domain, $\forall x[P(x)]$ is equivalent to the conjunction $\bigwedge_x P(x)$ over all entities x in the domain (Barklund, 1994). As such, I defined the truth value of $\psi \approx \forall x[dog(x) \rightarrow \exists y[tail(y) \wedge have(x, y)]]$ as $\min_e\{1 - min([[dog(e)]], 1 - [[\exists y[tail(y) \wedge have(e, y)]]]\}$: the fuzzy-logical equivalent of $\bigwedge_e dog(e) \rightarrow \exists y[tail(y) \wedge have(e, y)]$).

On the other hand, for each $\exists$-type PMRS such as $\psi \approx \exists x[dog(x) \wedge run(x)]$, we iterate over all entities e in the domain and calculate the fuzzy-logical equivalent of $dog(e) \wedge run(e)$: $min([[dog(e)]], [[run(e)]])$. Under a finite domain, $\exists x[P(x)]$ is equivalent to the disjunction $\bigvee_x P(x)$ over all entities x in the domain (Barklund, 1994). As $P \vee R$ is defined as $max(P, R)$ in many treatments of fuzzy logic (Zadeh, 1965), I defined the truth value of $\psi \approx \exists x[dog(x) \wedge run(x)]$ to be $\max_e\{min([[dog(e)]], [[run(e)]])\}$: the fuzzy-logical equivalent of $\bigvee_e dog(e) \wedge run(e)$.

Each non-quantified PMRS ψ consists of a list of implicitly-conjoined atomic propositions: for example, $\psi \approx happy(john) \wedge sleep(john)$. In this case, I defined the truth value $[[\psi]]$ to be $min\{[[happy(john)]], [[sleep(john)]]\}$: the fuzzy logical equivalent of $happy(john) \wedge sleep(john)$. For each of the atomic formulas such as $happy(john)$, we compute the truth value $[[happy(john)]]$ by first retrieving the corresponding triple $(happy(john), p, n)$ from the atomic proposition list. Then $[[happy(john)]] = p/(p + n)$: the positive occurrence count of $happy(john)$ divided by the total occurrence count of $happy(john)$. This step terminates the recursion in Equation 3.7.

Now, returning to the discussion of Step 4, recall that for each terminal node $\forall$-type PMRS ϕ, we begin by iterating over each entity e and evaluating the truth value $w = [[rstr(\phi_Q[0])(e)]]$ of $rstr(\phi_Q[0])(e)$. For example, let $\phi \approx \forall x[person(x) \rightarrow mortal(x)]$ be a terminal $\forall$-type node with $\phi_W = 1$, and suppose that the triples $(person(socrates), 1, 0)$, $(person(plato), 1, 1)$, and $(person(clifford), 0, 1)$ are already in the atomic proposition list, and that there are no other propositions involving the predicate *person* in the list. Then

$rstr(\phi_Q[0])(e) = person(e)$, so we have: $[[rstr(\phi_Q[0])(socrates)]] = 1.0$, $[[rstr(\phi_Q[0])(plato)]] = 0.5$, and $[[rstr(\phi_Q[0])(clifford)]] = 0.0$.

For all entities e such that $[[rstr(\phi_Q[0])(e)]] \neq 0.0$, a new PMRS $\phi(e)$ is generated, consisting of the scope of $\phi_Q[0]$ with all instances of the bound variable of $\phi_Q[0]$ replaced by e. The weight $\phi(e)_W$ is then set to $[[rstr(\phi_Q[0])(e)]]$, and $\phi(e)$ is sorted into the $\exists$-type, $\forall$-type, Π-type, or *non-quantified* categories according to the (type of the) new $\phi(e)_Q[0]$. If $\phi_Q[0]$ belongs to a coreference cluster c and is not a Π-type, the pair $(c, rstr(\phi_Q[0]))$ is added to the coreference store. If there are no entities e that satisfy the restriction of ϕ (i.e. such that $[[rstr(\phi_Q[0])(e)]] \neq 0$), we ensure co-occurrence between the restriction and scope of ϕ by converting the universal quantifier $\phi_Q[0]$ to an existential quantifier and re-sorting ϕ into the $\exists$-type category (see Step 3).

In our example above, $\phi(socrates) = mortal(socrates)$ with $\phi(socrates)_W = 1.0$, and $\phi(plato) = mortal(plato)$ with $\phi(plato)_W = 0.5$. As $[[person(clifford)]] = 0$, no corresponding PMRS $\phi(clifford)$ is generated.

The idea here is that we have some evidence that Plato is a person, but an equal amount of evidence that he is not. As such, the assertion that Plato is mortal ($mortal(plato)$) should receive less weight than the assertion that Socrates is mortal: we only have evidence that Socrates is a person, and none that he is not. After $\phi(plato)$ passes through Step 2, then $[[mortal(plato)]] = 0.5/0.5 = 1$: there is no evidence against Plato being mortal, and so the truth value is still 1. However, if another PMRS $\psi \approx \neg mortal(plato)$, with $\psi_W = 1$, is encountered, then $[[mortal(plato)]] = 0.5/(1 + 0.5) = 1/3$: the evidence for Plato being mortal came from a less certain inference (a universal statement whose restriction we are only half certain applies to Plato) than the evidence against Plato being mortal, and so we are more certain that he is not mortal. On the other hand, if $\phi(plato)_W$ were to equal 1 in the above example, and $\psi \approx \neg mortal(plato)$ with $\psi_W = 1$, were encountered, then $[[mortal(plato)]] = 1/(1 + 1) = 1/2$: the evidence for Plato being mortal is equally certain as the evidence against Plato being mortal.

As another example, suppose again that $\phi \approx \forall x[person(x) \rightarrow mortal(x)]$, but that no propositions involving the predicate *person* are in the atomic proposition list. Then, as discussed above, $\phi \leftarrow \exists x[person(x) \wedge mortal(x)]$ and ϕ is recategorized as an $\exists$-type PMRS, in order to ensure co-occurrence between $person(x)$ and $mortal(x)$.

At the end of Step 4, each terminal node is removed from the $\forall$-type partial order, and the new set of terminal structures is identified.

Step 5. For each Π-type PMRS ϕ belonging to a coreference cluster c, the procedure searches through the coreference store to find the corresponding (c, x) pair. If x is an entity, the bound variable of $\phi_Q[0]$ is replaced with x, $\phi_Q[0]$ is removed from ϕ_Q, and ϕ is re-sorted into the $\exists$-type, $\forall$-type, Π-type, or *non-quantified* categories according to the type of the new $\phi_Q[0]$. If x is a PMRS structure—i.e. x is the restriction of a $\forall$-type PMRS (see Step 4)—then a new PMRS ϕ' is instantiated by replacing the free variable of x with the bound variable of $\phi_Q[0]$, and converting $\phi_Q[0]$ to a universal quantifier with restriction x and nuclear scope ϕ. The new PMRS ϕ' is then inserted into the $\forall$-type partial order.

If there is no such (c, x) pair in the coreference store, ϕ is returned to the Π-type category to wait for the next pass through steps 2-6.

For example, suppose that $\phi \approx \Pi^c x[run(x)]$, and $(c, mary)$ is in the coreference store. Then after applying Step 5, $\phi \leftarrow run(mary)$ and ϕ is recategorized as a non-quantified PMRS. On the other hand, suppose again that $\phi \approx \Pi^c x[run(x)]$, but that $(c, alligator(-))$ is in the coreference store (there cannot be more than one referent paired with a given cluster c). This occurs only when there is a PMRS $\psi \approx \forall^c y[alligator(y) \rightarrow P(y)]$, and $\psi_Q[0]$ (i.e. the universal quantifier binding y) is the main instance of the cluster c. In this case, $\phi \leftarrow \forall x[alligator(x) \rightarrow run(x)]$ and ϕ is recategorized as a $\forall$-type PMRS. This is intended to reflect instances of anaphora along the lines of "*all alligators*$_i$ *are ...* *they*$_i$ *run frequently.*"

Step 6. If the $\exists$-type, $\forall$-type, Π-type, and *non-quantified* categories are all empty, the procedure is terminated, returning the set of $(p, c(p)_+, c(p)_-)$ atomic proposition, positive

count, negative count triples. Otherwise, we return to Step 2.

Following the procedure outlined in Steps 1-6 above, the set of triples $(p, c(p)_+, c(p)_-)$ in the atomic proposition list is then broken down further into a set of distributional contexts, generating the set of nearly all possible combinations of λ-abstracted arguments and predicates for each atomic proposition p (see Figure 3.5). This *atomic decomposition* procedure is partially intended to mitigate the sparsity inherent in the formal-logical distributional data generated by the procedure laid out in Steps 1-6 above. For example, suppose that we have the following two sentences: "*the man saw the house*" and "*the dog saw the squirrel*". FoLDS will generate multiple logical formulas from these two sentences, but there are two which are relevant to this example: $see(e_{man}, e_{house})$ and $see(e_{dog}, e_{squirrel})$. Without atomic decomposition, e_{man} and e_{dog} would only appear in the contexts $\lambda x.see(x, e_{house})$ and $\lambda x.see(x, e_{squirrel})$, respectively, meaning that e_{man} and e_{dog} would gain no overlap in properties from this pair of sentences.

However, using the atomic decomposition procedure, e_{man} and e_{dog} both appear in the context $\lambda x \lambda y.see(x, y)$, where the outermost λ-abstraction operator indicates the argument position in which a term occurs within the context[7]. This leads to an increase in similarity between the predicates *man* and *dog* due to the fact that they are both capable of sight.

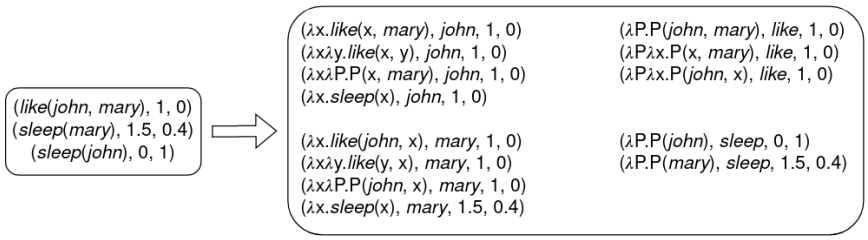


Figure 3.5: An example of the conversion of atomic propositions into distributional contexts. Note that each atomic proposition is broken down into nearly all possible combinations of λ-abstracted predicates and arguments—the only exception being those combinations in which *all* arguments and predicates have been λ-abstracted.

In Figure 3.5, the 4-tuple $(\lambda x.like(x, mary), john, 1, 0)$ encodes the fact that there is one

[7]This is to say that, for example, e_{dog} occurring in the context $\lambda x \lambda y.see(x, y)$ indicates that there is an instance of $see(e_{dog}, z)$ (for some entity z) in the data.

positive occurrence—and zero *negative* occurrences—of *john* liking *mary*. Similarly, the 4-tuple $(\lambda x \lambda P.P(x, mary), john, 1, 0)$ encodes the fact that there is one positive occurrence of *john* doing *something* to *mary*, $(\lambda x \lambda y.like(x, y), john, 1, 0)$ encodes the fact that there is one positive occurrence of *john* liking *something*, and so on. On the other hand, the tuple $(\lambda x.sleep(x), john, 0, 1)$ encodes the fact that there is one negative occurrence—and zero positive occurrences—of *john* sleeping: i.e. one count of him *not* sleeping.

3.2.5 Similarity Metric

The 4-tuples generated by the process laid out in Section 3.2.4 are then used to construct complex-valued count vectors for each entity and predicate in the domain: contexts—λ-abstracted formulas—index the dimensions, whose values are complex numbers of the form $a + bi$, where a is the positive count and b the negative count. For example, in the atomic decomposition demonstrated in Figure 3.5, the entity *john* is mapped to a 13-dimensional complex-valued vector: the embedding dimension is equal to the number of contexts (i.e. the number of 4-tuples in the right-hand side of the figure). Each coordinate is indexed by a context—the first element of the 4-tuples in Figure 3.5—so that, for example, the $\lambda x.like(x, mary)^{th}$ coordinate of the vector *john* ($john_{\lambda x.like(x,mary)}$) has a value of $1 + 0i$. Similarly, $john_{\lambda x \lambda y.like(x,y)} = 1 + 0i$, $john_{\lambda x \lambda P.P(x,mary)} = 1 + 0i$, and $john_{\lambda x.sleep(x)} = 0 + 1i$—for all of the remaining nine contexts ϕ in Figure 3.5, $john_{\phi} = 0 + 0i$.

FoLDS calculates the similarity between any two such entity vectors using the formula defined in Equation 3.8 below.

$$sim(x, y) = \tau(x, y) \cdot \Omega(x, y) \tag{3.8}$$

The term $\tau(x, y)$ is a length-one complex number that lies in the positive quadrant of the complex plane—in other words, $\tau(x, y)$ lies between $1 + 0i$ and $0 + 1i$ (inclusive). This complex number is a measure of the synonymy/antonymy between x and y: given two entities

x, y that are completely synonymous, the value of $\tau(x, y)$ will be $1 + 0i$. Conversely, if x and y are completely antonymous, the value of $\tau(x, y)$ will be $0 + 1i$. Values lying between these two extrema reflect graded degrees of synonymy/antonymy.

In the context of this discussion, I define synonymy/antonymy to be a function of the differences between the *truth values* of x and y with respect to the contexts ϕ in which they both appear—i.e. those contexts ϕ such that $x_\phi \neq 0 + 0i$ and $y_\phi \neq 0 + 0i$, where x_ϕ denotes the ϕ^{th} coordinate of the vector x. Equation 3.9 defines the fuzzy truth value of x_ϕ, $r(x_\phi) \in [0, 1]$: a real number that reflects the degree to which $\phi(x)$ holds in the fuzzy-logical model defined in Section 3.2.4.

$$r(a + bi) = \frac{a}{a + b} \tag{3.9}$$

Given a context ϕ and an entity x, the ϕ^{th} coordinate of x, $x_\phi = a + bi$, is a complex number, where a represents the positive occurrence count of $\phi(x)$, and b represents the negative count. The truth value $r(x_\phi)$ is then simply the positive occurrence count of $\phi(x)$ divided by the total occurrence count of $\phi(x)$.

Returning to the discussion of the term $\tau(x, y)$ of Equation 3.8, any two entities x and y are considered synonymous if for each context ϕ, the truth value of x_ϕ equals the truth value of y_ϕ, or $r(x_\phi) = 1$. On the other hand, x and y are considered *antonymous* if, for each context ϕ, $r(x_\phi) = 1 - r(y_\phi)$. This is designed to model logical implication in a fuzzy setting: in classical logic, $\phi(y) \rightarrow \phi(x)$ holds if $\phi(y) = \phi(x)$ or $\phi(x) = \top$, and does not hold only if $\phi(y) = \top$ and $\phi(x) = \bot$. This is modeled by the formula $\sigma(\phi, x, y)$ *in Equation 3.10*, which reflects the fuzzy degree to which $\phi(y) \rightarrow \phi(x)$ holds with respect to some context ϕ.

$$\sigma(\phi, x, y) = max(r(x_\phi), 1 - |r(x_\phi) - r(y_\phi)|) \tag{3.10}$$

Note that $\sigma(\phi, x, y) = 1$ if $r(x_\phi) = r(y_\phi)$ or $r(x_\phi) = 1$. In Equation 3.8, the term $\tau(x, y) = \tau'(x, y)/|\tau'(x, y)|$ is simply the length-one normalization of $\tau'(x, y)$: the term $\tau'(x, y)$ defined

in Equation 3.11 represents the the raw aggregation of the degree of synonymy/antonymy of x and y with respect to each context ϕ.

$$\tau'(x,y) = \sum_{\phi}(\sigma(\phi,x,y)\cdot\iota(\phi)) + (1-\sigma(\phi,x,y))i \tag{3.11}$$

In words: $\tau'(x,y)$ is calculated by summing over all contexts ϕ, first computing the value of $\sigma(\phi,x,y)$. The real part of $\tau'(x,y)$ is simply the sum of the values of $\sigma(\phi,x,y)$ for each context ϕ (i.e. the fuzzy degree to which $\phi(y) \rightarrow \phi(x)$ holds), while the imaginary part is the sum of $1-\sigma(\phi,x,y)$—the fuzzy degree to which $\neg(\phi(y) \rightarrow \phi(x))$ holds.

Note that the real part of $\tau'(x,y)$ is scaled by the positive real number $\iota(\phi)$. Unfortunately, there is no theoretical motivation for this scalar factor: I simply found empirically that it yields better results on inferencing tasks. For that reason, I postpone any further discussion of $\iota(\phi)$ to a later part of this section (see Equation 3.13).

Returning to Equation 3.8, the length-one complex number $\tau(x,y)$ is scaled by the real scalar $\Omega(x,y) \in [0,1]$, as defined in Equation 3.12. The term $\Omega(x,y)$ is an asymmetric measure of the overlap between the contexts in which x and y appear, and is designed to measure the fuzzy degree to which x is a hyponym of y. For example, $\Omega(\textit{apple},\textit{fruit})$ should be close to 1, but $\Omega(\textit{fruit},\textit{apple})$ should be closer to 0—every property of *fruit* is also a property of *apple*, but not vice versa.

$$\Omega(x,y) = \frac{\sum_{\phi} min(|x_{\phi}|,|y_{\phi}|)\cdot\iota(\phi)}{\sum_{\phi}|x_{\phi}|\cdot\iota(\phi)} \tag{3.12}$$

The scalar $\Omega(x,y)$ is a function of the ratio of the magnitudes $|x_{\phi}|$ and $|y_{\phi}|$ of x_{ϕ} and y_{ϕ} (respectively), for each context ϕ. The magnitude of a complex number is simply its length: as the magnitude of x_{ϕ} increases, so does the degree of attestation of $\phi(x)$—i.e. $\phi(x)$ occurs more often in the data. If $|x_{\phi}| = 0$, then neither $\phi(x)$ nor $\neg\phi(x)$ occurs in the data. As such, if $|x_{\phi}| \leq |y_{\phi}|$ for each context ϕ, then it is very likely that x is a hyponym of y (modulo negation).

To view this conceptually, let $E(x) = \{\phi \mid x_\phi \neq 0 + 0i\}$ be the set of all contexts ϕ whose value for x is nonzero (i.e. known), and assume that the magnitude of each nonzero coordinate of x and y is equal to 1. Then as $|E(x) \cap E(y)| \to 0$, $\Omega(x, y) \to 0$, and as $|E(x) \cap E(y)| \to |E(x)|$, $\Omega(x, y) \to 1$. The idea is that the closer that $E(x)$ is to a subset of $E(y)$, the more confident we can be that properties of y apply to x (again, modulo negation), and the further $E(x)$ is from a subset of $E(y)$, the less confident we are that properties of y apply to x.

Note that for each context ϕ, if $|x_\phi| \leq |y_\phi|$, then $|x_\phi| = min(|x_\phi|, |y_\phi|)$, and therefore $min(|x_\phi|, |y_\phi|) \,/\, |x_\phi| = 1$. If $|x_\phi| > |y_\phi|$, then we have $|x_\phi| > min(|x_\phi|, |y_\phi|)$, and therefore $min(|x_\phi|, |y_\phi|) \,/\, |x_\phi| < 1$. In words: if y occurs in the context ϕ more often than x, then $min(|x_\phi|, |y_\phi|) \,/\, |x_\phi| = 1$, and if x occurs more often than y in the context ϕ, then $min(|x_\phi|, |y_\phi|) \,/\, |x_\phi| < 1$.

Each summand $min(|x_\phi|, |y_\phi|)$ and $|x_\phi|$ in the numerator and denominator (respectively) is scaled by $\iota(\phi)$, which is defined in Equation 3.13.

$$\iota(\phi) = log_2 \frac{\mu_{max}}{\mu(\phi)} \tag{3.13a}$$

$$\mu(\phi) = \sum_x |x_\phi| \tag{3.13b}$$

$$\mu_{max} = \max_\phi \mu(\phi) \tag{3.13c}$$

The value of $\iota(\phi)$ is intended to mimic *inverse document frequency* (Sparck Jones, 1972) weighting, where the notion of document frequency has been replaced with the sum of the magnitudes of each coordinate of the context vector ϕ (Equation 3.13b). The value of $\iota(\phi)$ *decreases* as the frequency with which the context ϕ appears *increases*. The weights $\iota(\phi)$ ensure that less frequent contexts are more important to the value of $\Omega(x, y)$: if x and y both have known values for a very frequent context—i.e. a context/property that many entities have—that alone does not lend very much evidence to the transferability of properties from

y to x.

3.3 Experiment: Property Inference

In order to evaluate the viability of the FoLDS model and the effectiveness of its logical-form-derived embeddings, I evaluated FoLDS on a property inference task (described in Section 3.3.1), and compared its performance to that of the models evaluated in Rosenfeld and Erk (2022) (Section 3.3.2). I describe the methods that I employed to leverage the two-dimensional FoDLS similarity metric for property inference in Section 3.3.3, and report the results of this experiment in Section 3.3.4.

3.3.1 Task Description

Property inference refers to language users' ability to infer properties of the denotations of words purely from their linguistic distributions. For example, given the passage "many well-read adults know that Buddha sat long under a banyan tree [...] and Tahitian natives lived idyllically on breadfruit and poi" (Levy and Nelson, 1994), observing the terms *banyan tree*, *breadfruit*, and *poi* in this single linguistic context suffices to infer some of their properties—even if these words are entirely unfamiliar to the reader: a banyan tree must be somewhat large (as Buddha was able to sit under one), and breadfruit and poi must be foods. Language users are able to make such inferences without having any knowledge grounding these terms to real-world concepts.

Property inference is a logic-oriented task (Patalano, Wengrovitz, and Sharpes, 2009): given that an aardwolf is a type of animal, we assume that aardwolves have all of the properties that animals have: *alive*, *breathes*, etc. Here, we are reasoning from hyponymy (Herbelot and Vecchi, 2015): this is comparable to knowledge-representation frameworks such as KL-ONE (Brachman and Schmolze, 1989), in which *subclasses* (∼hyponyms) inherit values (∼properties) from their parent *superclasses* (∼hypernyms). However, property-inferential

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 3.1: McRae et al. (2005) feature norms for the concept *knife*. For all other features Q, $F(knife)_Q = 0$.

reasoning arguably goes beyond hyponymy: for example, given that alligators have the property *is-dangerous*, we assume that rabbits do not have that property—here, we are reasoning from antonymy. Property inference tasks therefore provide an ideal proving ground to evaluate the performance of a logic-oriented distributional model such as FoLDS.

As an NLP benchmark, property inference is typically performed over a feature norm database, such as the Vinson and Vigliocco (2002) and McRae et al. (2005) databases. As discussed in Section 3.1.3, a feature norm database consists of a set of *concepts* (words) and a set of *features*, in which each concept w is assigned a feature vector $F(w) \in \mathbb{R}^n$, where n is the number of features in the database: the value of $F(w)_Q$ is the value of the feature Q for the word w (see Table 3.1).

The McRae et al. (2005) database, which I used to evaluate FoLDS, consists of 541 concepts and 2,526 features: feature values are obtained from experiment participants' judgments. Rosenfeld and Erk (2022) create ten random folds consisting of 50 concepts each from the dataset. On each fold, the concepts within the fold represent the set U of *unknown* words—words which have been observed in text but are not grounded to real-world concepts—and the concepts outside of the fold represent the set K of *known* words. For each unknown word $u \in U$, the feature vector $F(u)$ is withheld: the task is to reconstruct $F(u)$ given the features of the known words in K and the similarity between u and each word in K.

3.3.2 Previous Work

In their analysis, Rosenfeld and Erk (2022) evaluated a wide variety of property inference methods. Note that all of the property inference methods evaluated by those authors share the same distributional word embeddings: LSA vectors drawn with a context window of two from four different corpora, comprising a total of ∼10.3 billion words: ukWaC (Ferraresi et al., 2008), Google Gigaword (Graff and Cieri, 2003), English Wikipedia[8], and the BNC (BNC, 2007). What varies across methods is how they use these embeddings to estimate properties.

A variant of *Modified Adsorption* (ModAds; Talukdar and Crammer, 2009)—a label-propagation algorithm—achieved SoTA results in Rosenfeld and Erk's (2022) analysis. I refer interested readers to Talukdar and Crammer (2009) and Rosenfeld and Erk (2022) for an in-depth explanation of ModAds and its application to property inference tasks.

Many of the property inference methods that Rosenfeld and Erk (2022) evaluated include a *shifted* variant. This does not indicate a difference in the models' architectures, but rather the feature vectors $F(w)$. In the shifted trials, Rosenfeld and Erk (2022) decrease the values of those properties Q such that $F(w)_Q = 0$ to negative values, in order to increase the separation between irrelevant and relevant properties.

3.3.3 Experiment

For this experiment, I used Simple English Wikipedia[9] (SEW) to generate FoLDS embeddings. There were two primary motivations for this choice of dataset: first, SEW is intended for second-language learners of English and people with learning disabilities. As such, the sentences in SEW are simpler, shorter, and use less complex grammatical constructions than those in the standard English Wikipedia, which facilitates the use of the ERG rule-based parser. Second, SEW is significantly smaller than English Wikipedia—24.5 million vs. 4.2 billion words, respectively—making the natural-language-to-logic translation process

[8]https://en.wikipedia.org/
[9]https://simple.wikipedia.org/

described in Section 3.2 more computationally feasible.

Recall that the procedure described in Section 3.2 only yields vectors for each *entity*, not each *word*. For each of the concept words w in the McRae et al. (2005) database, I obtained an embedding by summing together the entity vectors for each entity in $\{e \mid \exists z \in \mathbb{R}^+[e_w = z + 0i]\}$—the set of all entities that have a nonzero positive count, and a zero negative count, for the context $\lambda x.w(x)$.

In order to interface with the FoLDS similarity metric (Equation 3.8) in a meaningful way, it was necessary to convert the real-valued feature vectors $F(w)$ of the McRae et al. (2005) database into complex-valued vectors $C(F(w))$. For each term (word) w and property Q in the dataset, I set $C(F(w))_Q$ to $0 + 1i$ (90°) if $F(w)_Q = 0$. Otherwise, I placed $C(F(w))_Q$ on the positive quadrant of the unit circle with its angle between 0° and 45°, inversely proportional to the value of $F(w)_Q$—mimicking Rosenfeld and Erk's (2022) shifting procedure (see Section 3.3.2)—as defined in Equation 3.14.

$$C(F(w))_Q = \begin{cases} 0 + 1i & \text{if } F(w)_Q = 0 \\ \beta(F(w)_Q) & \text{otherwise} \end{cases} \tag{3.14a}$$

$$\beta(z) = \frac{(1+z) + (1-z)i}{|(1+z) + (1-z)i|} \tag{3.14b}$$

To estimate properties, I replicated the Johns and Jones (2012) method—but with complex, rather than real, numbers—which estimates property values for unknown words as the sum of all of the property values of the known words, weighted by their cosine similarity with the unknown word in question. Letting K denote the set of known words, u denote a given unknown word, and Q denote a given property, the Johns and Jones (2012) method calculates the estimated value of Q for u, $P(u)_Q$, as in Equation 3.15.

$$P(u)_Q = \sum_{w \in K} F(w)_Q \cdot cos(u, w)^\lambda \tag{3.15}$$

Where λ is a hyperparameter—higher values of λ reduce the influence of less similar words.

In this experiment, I found empirically that only considering the set $K_n(u)$ of the top n most related known words w to a given unknown word u—as determined by the magnitude $\Omega(u, w)$—yielded the best performance for the FoLDS model. Via grid search, I found $n = 25$ to be the optimal value for this task. For a given unknown word u and property Q, the estimated complex value of Q for u, $P(u)_Q$, is calculated as in Equation 3.16.

$$P(u)_Q = \sum_{w \in K_n(u)} abs(C(F(w))_Q \cdot sim(u, w)) \tag{3.16}$$

Where $abs(a + bi) = |a| + |b|i$. This function forces the resulting value of $C(F(w))_Q \cdot sim(u, w)$ to the positive quadrant of the complex plane while preserving its distance from the real axis.

Recall that $sim(u, w)$ (Equation 3.8) is the product of the length-one complex number $\tau(u, w)$ (Equation 3.11) and the real scalar $\Omega(u, w)$ (Equation 3.12), which reflect synonymy/antonymy and relatedness, respectively. If u and w are synonymous, then $\tau(u, w) = 1 + 0i$. Given some property of w—represented by a positive-quadrant complex number $a + bi$—whose value is unknown for u, $(a + bi) \cdot \tau(u, w) = a + bi$: synonymous words are predicted to have the same values for each property. On the other hand, suppose that u and w are antonymous, so that $\tau(u, w) = 0 + 1i$, which corresponds to a 90° rotation: antonymous words are predicted to have opposite property values. The real number $\Omega(u, w)$ scales $\tau(u, w)$: the known words w with higher values of $\Omega(u, w)$—i.e. those which are more related to u—will contribute more to the overall inference than those with lower values (see Figure 3.6).

In order to compare the predicted values $P(u)$ to the ground-truth values $F(u)$, each $P(u)_Q$ must be converted to a real number $r(P(u))_Q$. Recall that the values $C(F(u))_Q$ are shifted (Equation 3.14), which I accounted for by placing a floor ξ on the values of $r(P(u))_Q$.

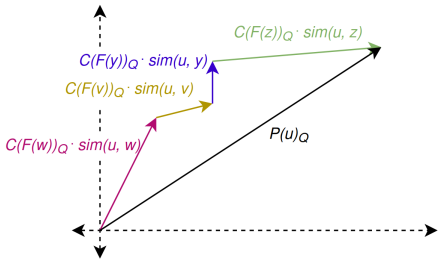


Figure 3.6: Equation 3.16 can be viewed as the average of the predicted values of Q for u for each known word x (in this example, $x \in \{w, v, y, z\}$), weighted by their overlap $\Omega(u, x)$ (Equation 3.12).

In this experiment, I found via grid search that $\xi = 0.15$ yielded the best results. This floor was implemented via the formula in Equation 3.17, where $r(-)$ is as defined in Equation 3.9.

$$R(P(u))_Q = \begin{cases} r(P(u)_Q) & \text{if } r(P(u)_Q) \geq \xi \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

As $P(u)_Q$ approaches the real axis, $R(P(u))_Q \to 1$, and as $P(u)_Q$ approaches the imaginary axis, $R(P(u))_Q \to 0$.

Note that, formally, $R(P(u))_Q$ is the probability that, for any x such that $u(x)$ holds, $Q(x)$ holds as well. Under the definition of a language model over logical forms given in Chapter 1 (Definition 3): $R(P(u))_Q = p([\text{MASK}] = Q \mid \forall x[u(x) \to [\text{MASK}](x)])$.

3.3.4 Evaluation and Results

Rosenfeld and Erk (2022) used two separate evaluation metrics in their analysis. The first, *Mean Average Precision* (MAP), was originally designed as an evaluation metric for information retrieval systems (Zhu, 2004). Applied to this task, MAP (which ranges from 0 to 1) measures a given property inference method's ability to rank relevant features above irrelevant features. In the McRae et al. (2005) database, relevant features for a given word $u \in U$ are those features Q whose value $F(u)_Q > 0$ (i.e. those with non-zero values). For example, a system would receive a perfect *Average Precision* (AP) score—MAP is simply

the mean over the AP scores for each unknown word—for the concept *knife* if the top six predicted features for that word were *a-utensil*, *found-in-kitchens*, *used-with-forks*, *a-cutlery*, *is-dangerous*, and *a-weapon* (see Table 3.1). The order (ranking) of these predicted features does not impact the AP score: any permutation of the above list would still result in a perfect score, as long as those features remain the top six returned.

Additionally, it does not matter what value is predicted for the ground-truth irrelevant features, as long as the ground-truth relevant features are ranked higher than the irrelevant ones. As an illustration, suppose that *is-dangerous*, *a-utensil*, *used-with-forks*, *found-in-kitchens*, *a-cutlery*, and *a-weapon* are the top six predicted features for the concept *knife*, in that order (refer to Table 3.1). Suppose further that the predicted value $P(knife)_{a\text{-}weapon}a\text{-}weapon = 0.81$, so that the predicted values for the other relevant features are all higher than 0.81. In this case, the system could predict a value $P(knife)_{requires\text{-}pilots} = 0.805$ for some irrelevant feature such as *requires-pilots*, and still receive a perfect AP score. Again, all that matters for the MAP evaluation is that relevant features are ranked above irrelevant features.

For this reason, I argue that MAP is not an ideal metric for evaluating property inference methods. If the system has a perfect MAP score, that by no means excludes the possibility that it predicts high values for irrelevant features. One could object that, given a property inference system that achieves a perfect MAP score, we need only assign non-zero values to the top k features—for example, $k = 6$—and zero out the rest, in order to recover the ground-truth list of properties for each unknown word. However, the number of non-zero features varies across words in the McRae et al. (2005) database: some words have as few as six, and some as many as 26. In this hypothetical scenario, any choice of k will inevitably lead to some words receiving fewer relevant predicted features than the ground-truth, and some receiving irrelevant predicted features.

Many of the problematic aspects of the MAP evaluation do not pertain to the second evaluation metric employed in Rosenfeld and Erk's (2022) analysis, Spearman's Rank Correlation Coefficient (Spearman ρ; Spearman, 1904). Rosenfeld and Erk (2022) calculate the

Spearman ρ for each unknown word against its ground-truth values in the McRae et al. (2005) database, then average over all Spearman ρ scores for each unknown word in each fold to yield an overall score for each model.

Under the Spearman ρ evaluation metric, the raw predicted values for each feature are not necessarily relevant: what is important is the *ranking* between the predicted features. While this is arguably a drawback of this metric, it is also a drawback suffered by the MAP score. However, as mentioned above, many deficiencies of MAP do not pertain to Spearman ρ: for example, *a-utensil*,*found-in-kitchens*, *used-with-forks*, *a-cutlery*, *is-dangerous*, and *a-weapon* must be the top six predicted features for the concept *knife*—in that order—to obtain a perfect Spearman ρ score. Furthermore, no irrelevant features can be ranked above one another: they all must receive the exact same predicted value.

Suppose, for the sake of example, that a property inference model returns *a-utensil*, *found-in-kitchens*, *used-with-forks*, *a-cutlery*, *is-dangerous*, and *a-weapon*—in that order—as the top six predicted features for the concept *knife*, and that the predicted value $P(knife)_{a\text{-}weapon} = 0.81$ (so that the predicted values for the other relevant features are all higher than 0.81). In this case, such a system could predict a value $P(knife)_{requires\text{-}pilots} = 0.805$ for some irrelevant feature such as *requires-pilots* and still receive a perfect Spearman ρ score, but only if $P(knife)_Q = 0.805$ for all other irrelevant features Q. If any irrelevant features—i.e. those with a ground-truth value of 0 for the concept *knife* in the McRae et al. (2005) database—are ranked differently, the system will be penalized under the Spearman ρ metric.

Suppose that we have a property inference method which achieves a perfect Spearman ρ score. As with the parallel example for the MAP score discussed above, such a system very well could assign a non-zero value to irrelevant features. However, in contrast to MAP, all irrelevant features *must* receive the same predicted value, which must be lower than the lowest predicted value for any given relevant feature. In such a case, we need only zero out the lowest predicted value (by assumption assigned to all irrelevant features), and re-normalize the remaining values—those assigned to relevant features—to fall between zero and one. This

Method	ρ	**Method**	ρ
Property frequency	0.049	Property sum	0.042
JJ (1 step)	0.114	JJ (2 step)	0.107
Linear SVM	0.077	Linear SVM shifted	0.089
Cosine SVM	0.082	Cosine SVM shifted	0.082
Linear PLS	0.077	Linear PLS shifted	0.075
Cosine PLS	0.082	Cosine PLS shifted	0.083
ModAds equal	0.161	ModAds equal shifted	0.244
ModAds decay	0.161	ModAds decay shifted	0.243
ModAds NN	0.244	ModAds NN shifted	**0.281**
FoLDS	0.253		

Table 3.2: Comparison of methods in Rosenfeld and Erk (2022) against FoLDS on the McRae et al. database.

procedure will recover the correct ranking of relevant features for each unknown word, and will assign a value of zero to all irrelevant features.

For the above reasons, I concluded that Spearman ρ is far more effective as an evaluation metric for property inference tasks than MAP, and so do not report MAP scores in my analysis. Following Rosenfeld and Erk (2022), I averaged over all Spearman ρ scores for each unknown word in each fold for evaluation. FoLDS achieves a Spearman ρ of 0.253: the second-best out of the 18 methods in Rosenfeld and Erk's (2022) analysis (see Table 3.2).

Crucially, all of these methods use LSA vectors generated from a PPMI-transformed co-occurrence matrix (Roller, Erk, and Boleda, 2014) obtained from a lemmatized and POS-tagged 10.3 billion word corpus, while FoLDS uses count vectors obtained from a 24.5 million word corpus ($\sim$420 times smaller). These results provide strong support towards the Accelerated Learning Hypothesis of Chapter 1, demonstrating that a language model over logical forms is capable of competitive performance against a superficial model that uses significantly more data.

3.4 Discussion

In Section 3.2.1 (and Chapter 1), I argued that translating surface text to logical form has a de-noising effect: syntactic paraphrases are effectively equivalence-classed, so that, for example, an active sentence and its passive counterpart are mapped to the same representation. I posited that this de-noising effect should allow language models over logical forms to obtain meaningful embeddings from less training data, because—unlike a superficial model—they do not need to learn to equate the various periphrastic realizations of a given proposition. This conjecture is intrinsically linked to the Accelerated Learning Hypothesis of Chapter 1: the semantic equivalence between syntactic paraphrase constructions is itself elementary linguistic knowledge that is inherent to the architecture of a language model over logical forms, by virtue of its logical form inputs.

The de-noising effect afforded by the function-argument structure of logical forms additionally permits the sensitivity to negation required to generate the complex-valued word embeddings that allow FoLDS to leverage two axes of similarity simultaneously: logical forms reduce the various surface realizations of negation (see e.g. Chapter 2) to a single symbol ($\neg$), and explicitly delineate its scope, thereby facilitating the incorporation of negation into the model's embeddings.

Unlike similar models (e.g. Emerson, 2018; Herbelot and Copestake, 2021), the FoLDS model introduced in Section 3.2 is sufficiently robust to be able to be applied to noisy, real-world text data, demonstrating that the fallibility of a rule-based parser such as the ACE ERG does not irreparably impede the use of logical forms as distributional contexts. In Section 3.3, I showed that FoLDS can achieve near-SoTA results on a property inference task—the second-best out of 18 methods, and within 10% of the SoTA— using over 400 times less training data than competing approaches, a result which constitutes strong evidence towards the syntactic de-noising conjecture of Section 3.2.1, and therefore towards the Accelerated Learning Hypothesis.

As a prototype designed to demonstrate the potential of language models over logical

forms, FoLDS constitutes a substantial step towards accomplishing the objectives of this dissertation. However, as a general-purpose language model, FoLDS still leaves much to be desired.

While the complex-valued word embeddings generated by FoLDS permit the model to leverage two axes of similarity simultaneously (see Section 3.2.5), the use of complex-valued embeddings is fairly non-standard in the field of NLP (and machine learning in general). Although complex-valued neural networks do exist, these architectures are in their infancy and typically rely on converting the complex values to real numbers (e.g. as polar coordinates) to be passed through a real-valued model, as complex derivatives are not conducive to standard optimization algorithms such as gradient descent (Bassey, Qian, and Li, 2021). The few extant, truly complex-valued neural network architectures do not enjoy a high degree of infrastructural support in common machine learning libraries such as PyTorch[10] and TensorFlow[11].

Further compounding this problem is the fact that FoLDS only generates count vectors: the resulting embeddings are very sparse ($\sim$200,000-dimensional) and therefore will fail to capture latent features to the same extent as reduced-dimensionality representations. The scarcity of complex-valued machine learning architectures effectively precludes the application of any practical dimensionality-reduction techniques to FoLDS embeddings, and severely limits the model's potential range of downstream tasks, restricting its applications to those contexts in which we are able to explicitly hard-code prediction algorithms, such as in Section 3.3.3.

Equally problematic is the fact that FoLDS can only generate static word vectors: there is no straightforward way to obtain dynamic—or phrase- and sentence-level—embeddings from these representations. The model's inability to distributionally represent linguistic units beyond the level of the word further limits its utility, prohibiting its application to more advanced NLP tasks. For these reasons, it will be necessary to employ a more powerful, neural

[10]https://pytorch.org/
[11]https://www.tensorflow.org/

architecture in order to accomplish the main objective of this dissertation: demonstrating the viability of language modeling over logical forms.

Chapter 4

Graph-based FoLDS (GFoLDS)

In Chapter 3, I introduced and evaluated the FoLDS model, which generates complex-valued word vectors drawn from a fuzzy-logical model world imperatively constructed from logical-form representations of sentences in a corpus. I showed that FoLDS outperforms all but one label-propagation algorithm—and other machine learning architectures—on a property inference task while using a pretraining corpus roughly 420 times smaller than these competing approaches, thereby demonstrating the feasibility of language modeling over logical forms. However, the deficiencies of this architecture—namely its sparse, complex-valued static word embeddings—present a severe limitation to the model's application to a wider range of downstream tasks.

As discussed in Chapter 3, the above-mentioned limitations of FoLDS suggest two fundamental desiderata to consider when constructing a language model over logical forms: (i) the embeddings generated by such a model should be *real*-valued, in order to permit the model to more readily interface with downstream neural architectures; and (ii) the model should be capable of generating *dynamic* (i.e. context-sensitive) embeddings to facilitate its employment in NLP tasks necessitating awareness of linguistic units beyond the level of the word.

To address these desiderata, I introduce in this chapter the *Graph-based Formal-Logical*

Distributional Semantics (GFoLDS) model: a modified transformer (Vaswani et al., 2017) encoder architecture—specifically, GFoLDS is a variant of the *graph transformer* paradigm introduced in Wu et al. (2021) (see Section 4.1.2)—that takes as input directed graph representations derived from Dependency MRS (DMRS; see Copestake, 2009 and Section 4.1.1) structures. As an encoder transformer, GFoLDS by default generates real-valued, contextual embeddings: as discussed in Chapters 2 and 3, I do not claim that transformers in and of themselves are problematic, merely that *superficial* transformers are. Transformers in general are a "tried and true" architecture that have achieved SoTA results in applications ranging from NLP to computer vision (e.g. Dosovitskiy et al., 2020; Carion et al., 2020; Wang, Yeshwanth, and Nießner, 2021) and computational biology (e.g. Rives et al., 2021; Abramson et al., 2024).

While more recent LLMs have been trending towards encoder-decoder (e.g. Lewis et al., 2020; Raffel et al., 2020, etc.) or decoder-only (e.g. Brown et al., 2020; Touvron et al., 2023; OpenAI, 2023, etc.) transformers for sequence-to-sequence or generative models (respectively), such architectures are not applicable to the objectives of this dissertation. In the context of graph-based NLP models, encoder-decoder architectures are typically applied to graph-to-sequence (i.e. graph-to-text) tasks: for example, decoding semantic representation graphs (such as DMRS) into the text to which they correspond (e.g. Hajdik et al., 2019; Wang, Wan, and Jin, 2020; Wang, Wan, and Yao, 2020; Beck, Haffari, and Cohn, 2018; Guo et al., 2019; Zhang et al., 2020b, etc.; see Section 4.1.3). On the other hand, while a graph-to-graph decoder architecture for generating logical representations—for example, for chain-of-thought reasoning (see e.g. Wei et al., 2022)—is certainly an interesting avenue of research, this falls decidedly outside of the scope of this dissertation and is therefore left to future work: a tentative path towards constructing such a model is outlined in Chapter 7.

As a *graph* (encoder) transformer, GFoLDS is more well-suited to the task of language modeling over logical forms than *sequential* encoder transformers—i.e. those that take a linearly-ordered sequence of tokens as input, e.g. (Devlin et al., 2019; Liu et al., 2019; He

et al., 2021, etc.); graph representations are not necessarily *required* to construct a transformer LM over logical forms, but this format has two desirable properties in particular that motivate their employment in this role.

First, graph representations permit permutation-invariance: a graph representation of a formula such as (for example) $\phi \wedge \psi$ is equivalent (i.e. isomorphic) to that of the formula $\psi \wedge \phi$. If the logical formulas were encoded as strings, any model over those string representations would need to incorporate linear positional encodings in order to read the formulas in a meaningful way. Such an approach renders the model vulnerable to learning spurious correlations with respect to specific orders of operands of commutative operators such as conjunction and disjunction. Conversely, graph representations are also sufficiently flexible to encode linear order when necessary—e.g. for logical implication $\phi \rightarrow \psi$—via (for example) edge labeling or Levi encoding (Levi, 1942).

Second, graph representations facilitate the encoding of rich structures such as the scope of logical operators: for example, the restriction of a quantifier can be represented as the set of nodes dominated by (the node corresponding to) that quantifier in the graph. While such structures can be encoded in string representations of logical formulas using (for example) parentheses, this approach once again runs the risk of the model learning spurious correlations with respect to sequences of purely structural symbols—for example, three parentheses in a row.

These desirable properties of graphs with respect to logical-form representations have led to their employment in several semantic frameworks. In Section 4.1, I provide an overview of DMRS—the specific graph-based framework that I employ with the GFoLDS model—and discuss the current literature on the use of graphs in deep learning in general and their applications to NLP in particular.

In Section 4.2, I define the architecture of the GFoLDS model, and then discuss the preprocessing steps that I take to convert DMRS graphs into GFoLDS inputs in Section 4.3. Finally, in Section 4.4, I describe the details of GFoLDS' pretraining procedure—which

allows the model to be fine-tuned on a variety of downstream tasks (see Chapter 5)—and its *masked node modeling* (MNM) pretraining objective, which is analogous to the masked *language* modeling objective used to pretrain many encoder transformer LMs (e.g. Devlin et al., 2019; Liu et al., 2019; He et al., 2021, etc.).

4.1 Background and Related Work

In this section, I describe the DMRS framework from which the GFoLDS model's input graphs are derived (Section 4.1.1). I then provide an overview of current methods in deep learning for modeling graph data (Section 4.1.2), and survey existing applications of these methods within the domain of NLP (Section 4.1.3).

4.1.1 Dependency MRS (DMRS)

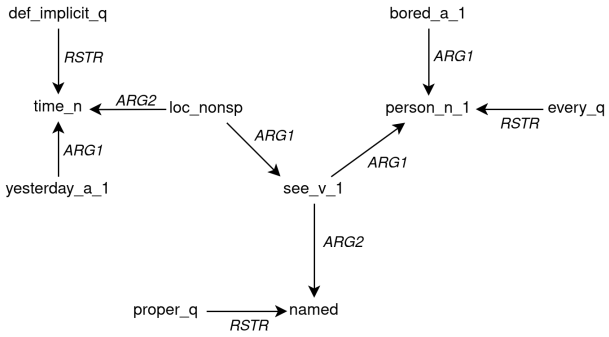


Figure 4.1: DMRS representation of the sentence "*Every bored person saw Mary yesterday.*"

DMRS is a semantic framework designed to simplify MRS (see Chapter 3 and Copestake et al., 2005) by encoding MRS structures as labeled, directed acyclic dependency graphs (Copestake, 2009). Note that this graphical format is *equivalent* to MRS: the process of representational simplification to DMRS does not result in a loss of information.

Given an MRS structure M, let $G_M = (V, E)$ be its corresponding DMRS graph, where V denotes the set of nodes and E the set of directed edges $x \xrightarrow{\ell} y$ (with labels ℓ) from $x \in V$ to $y \in V$. For each elementary predication (hereafter *predicate*) of M, there is a corresponding

node with the same label in V: there is a one-to-one correspondence between the nodes V of G_M and the predicates of M.

Note that, by construction, every MRS variable x—regardless of type: entity, event, etc.—is the zeroth-place argument (i.e. *ARG0*) of a *unique* predicate ϕ—such an x is referred to as the *intrinsic* (or characteristic) variable[1] of ϕ. The intrinsic variables of predicates corresponding to nouns are typically entity-type, while those of predicates corresponding to verbs, adjectives, and adverbs are typically event-type.

These intrinsic variables of MRS permit DMRS to omit variables entirely (see Figure 4.1). For each predicate ϕ, let $A(\phi)$ denote the set of argument label, variable pairs (ℓ, x) of ϕ, where $\ell \in \{ARG1, ARG2, RSTR, \dots\}$, excluding *ARG0*: for example, $(ARG2, x) \in A(\phi)$ indicates that x is the second-place argument of ϕ. For each $(\ell, x) \in A(\phi)$, there is an edge[2] $\phi \xrightarrow{\ell} \mathcal{I}(x)$ in G_M, where $\mathcal{I}(x)$ is the unique predicate in M such that x is the *ARG0* of $\mathcal{I}(x)$—i.e. x is the intrinsic variable of $\mathcal{I}(x)$.

As an example, consider the MRS structure corresponding to the sentence "*she read the book*" in Equation 4.1.

$$\begin{aligned}
&h_1\colon \mathit{read_v_1}(e_1, x_1, x_2)\\
&h_4\colon \mathit{pron}(x_1)\\
&h_5\colon \mathit{pronoun_q}(x_1, h_6, h_7)\\
&h_9\colon \mathit{the_q}(x_2, h_{10}, h_{11})\\
&h_{12}\colon \mathit{book_n_of}(x_2, i_1)\\
&h_6 =_q h_4, h_{10} =_q h_{12}
\end{aligned} \tag{4.1}$$

[1]The converse does not necessarily hold: all variables are the intrinsic variable of some predicate, but not all predicates have an intrinsic variable (e.g. quantifiers).

[2]DMRS also includes *secondary* link labels (*/QEQ* and */NEQ*) attached to the primary link labels *ARG1*, *ARG2*, etc., in order to encode underspecified scopal constraints. Scopal constraints are not relevant to the GFoLDS model, and I omit these secondary labels when constructing GFoLDS inputs from DMRS graphs. I refer interested readers to Copestake (2009) for a discussion of secondary link labels and the scopal constraints that they represent.

Here, the first-place argument of each non-quantifier predicate is its intrinsic variable (quantifiers do not have intrinsic variables): $\mathcal{I}(e_1) = \textit{read\_v\_1}$, $\mathcal{I}(x_1) = \textit{pron}$, and $\mathcal{I}(x_2) = \textit{book\_n\_of}$.

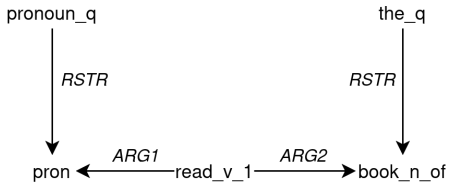


Figure 4.2: The DMRS graph derived from the MRS structure in Equation 4.1.

The MRS structure in Equation 4.1 yields the DMRS graph G in Figure 4.2: the first-place argument ($ARG1$) of *read_v_1* is the intrinsic variable of *pron* (x_1), so there is an edge $\textit{read\_v\_1} \xrightarrow{ARG1} \textit{pron}$ in G. As the second-place argument ($ARG2$) of *read_v_1* is the intrinsic variable of *book_n_of* (x_2), there is an edge $\textit{read\_v\_1} \xrightarrow{ARG2} \textit{book\_n\_of}$, and so on.

In MRS, features/properties[3] are associated with *variables*. For example, a singular noun or past-tense verb is expressed by assigning the *NUM:SG* or *TENSE:PAST* feature to the intrinsic variables of their respectively corresponding predicates. DMRS, on the other hand, encodes features as attributes of predicates: each feature of a given variable x in M is assigned to $\mathcal{I}(x)$ in G_M (see Figure 4.3).

The use of these simpler DMRS structures results in streamlined graphs with significantly fewer nodes when compared to those derived from standard MRS, thereby facilitating the use of semantic graph representations as input to the GFoLDS model discussed in Section 4.2 below. Graph data, however, represents a drastically different domain than those typically modeled with neural architectures (e.g. text and images), requiring specialized architectures to adapt to such structures.

[3]Roughly equivalent to the Chomskyian notion of phi-features (person, number, gender, tense, etc.); see e.g. Adger and Harbour (2008).

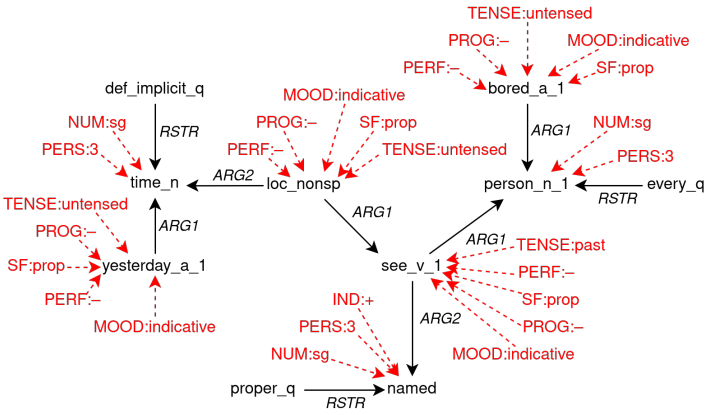


Figure 4.3: DMRS graph from Figure 4.1, with node features included. Features are highlighted in red; a dashed arrow $\phi \dashrightarrow x$ indicates that ϕ is a feature of the node x.

4.1.2 Graph Neural Networks (GNNs)

In comparison to graphs, modalities such as images and text have considerably more regular structure: images are always rectangular arrays of pixel values, and text data is invariably arranged as a linear sequence of characters or tokens. This facilitates machine learning over such data, as architectures can (relatively) easily be designed with appropriate structural priors due to such regularity. For example, linear positional encodings in superficial transformers (see Vaswani et al., 2017) and the implicit linearity encoded via recurrence in LSTMs (Hochreiter and Schmidhuber, 1997) yield architecural awareness of the sequentiality of textual data, while convolution kernels in convolutional neural networks (CNNs; LeCun et al., 1989) directly account for the translation-invariance inherent in image data.

Constructing effective structural priors with respect to graph data is considerably more difficult, due to its comparative irregularity. We cannot employ transformer-like linear positional encodings, as nodes in a graph are not typically totally-ordered (and graph data would not be of interest if they were). Nor can we define graph convolution kernels in an identical manner to those employed by CNNs: while every pixel in an image has the same local neighborhood structure[4]—one above, one below, one to the left, and one to the right—local neighborhoods in a graph vary across nodes.

[4]Although pixels at the edges of a raw image do not share this exact neighborhood structure, images are typically padded around the edges during preprocessing in order to uniformize pixel neighborhood structures.

There are various approaches to applying the techniques of deep learning to graph structures; I will ignore random-walk-based and transductive[5] methods (e.g. Perozzi, Al-Rfou, and Skiena, 2014; Yang, Cohen, and Salakhudinov, 2016) in this section, and focus on inductive, *message-passing* Graph Neural Networks (GNNs): models which learn node representations by iteratively passing information between nodes along graph edges (Chami et al., 2022).

4.1.2.1 Graph Convolutional Networks

One such GNN architecture, the Graph Convolutional Network (GCN; Kipf and Welling, 2017), generalizes the notion of CNNs to graph data: the kernels convolve over a given node's local neighborhood (i.e. the set of nodes to which it is directly connected via graph edges), regardless of its structure. Given a graph $G = (V, E)$, each node $v \in V$ is assigned an initial embedding $H_v^{(0)} \in \mathbb{R}^{C_0}$, which is typically based on the label/features of v. The GCN itself consists of L GCN layers, where for each $1 \leq i \leq L$, the i^{th} layer contains C_i convolution kernels. The output of the final layer is then typically passed to a feed-forward network for node classification.

The output $H^{(i)} \in \mathbb{R}^{|V| \times C_i}$ of the i^{th} layer is given in Equation 4.2 below, where σ is a non-linear activation function (ReLU is used in Kipf and Welling, 2017), $H^{(i-1)} \in \mathbb{R}^{|V| \times C_{i-1}}$ is the output of the previous layer (or the initial embeddings, if $i = 1$), and $\Theta^{(i)} \in \mathbb{R}^{C_{i-1} \times C_i}$ is the layer's learnable weight matrix: each column of $\Theta^{(i)}$ is a convolution kernel. $\hat{A} \in \mathbb{R}^{|V| \times |V|}$ is the degree-normalized graph adjacency matrix with added self-connections: $\hat{A}_{kk} = (d_k)^{-1}$, $\hat{A}_{jk} = (\sqrt{d_j d_k})^{-1}$ if $k \to j \in E$ (i.e. the j^{th} and k^{th} nodes are edge-connected in G), and $\hat{A}_{jk} = 0$ otherwise, where d_k is the degree—the number of nodes to which it is edge-connected—of the k^{th} node plus one.

$$H^{(i)} = \sigma\left(\hat{A} H^{(i-1)} \Theta^{(i)}\right) \tag{4.2}$$

[5] Learning representations on a *fixed* graph, in comparison to *inductive* approaches, which are designed to generalize representations to unseen graphs.

Letting $\hat{H}^{(i)} = H^{(i-1)}\Theta^{(i)}$, note that $\hat{H}^{(i)}_{jk}$ is the dot product of the j^{th} row vector of $H^{(i-1)}$ with the k^{th} column vector of $\Theta^{(i)}$—i.e. the result of applying the k^{th} convolution kernel to the embedding of the j^{th} node. The j^{th} row of $\hat{H}^{(i)}$ is then the vector of outputs of the convolution kernels in $\Theta^{(i)}$ with respect to the embedding of the j^{th} node.

The j^{th} row vector $(\hat{A}\hat{H}^{(i)})_j \in \mathbb{R}^{C_i}$ of $\hat{A}H^{(i-1)}\Theta^{(i)}$ is the value of the j^{th} node embedding before the application of σ, and is equivalent to the formula given in Equation 4.3 below, where $\mathcal{N}(j) = \{k \in V \mid k \to j \in E\}$ denotes the local neighborhood of the j^{th} node.

$$(\hat{A}\hat{H}^{(i)})_j = \sum_{k \in \mathcal{N}(j) \cup \{j\}} \hat{A}_{jk} \cdot \hat{H}^{(i)}_k = \left(\frac{1}{d_j} \cdot \hat{H}^{(i)}_j\right) + \sum_{k \in \mathcal{N}(j)} \frac{1}{\sqrt{d_j d_k}} \cdot \hat{H}^{(i)}_k \tag{4.3}$$

In words: $1/d_j \cdot \hat{H}^{(i)}_j$ is the vector of outputs of the i^{th} layer's convolution kernels with respect to the output of the previous layer's representation for the j^{th} node, scalar multiplied by the inverse of the degree of the j^{th} node ($1/d_j$). This is then summed together with the vector of outputs of the i^{th} layer's convolution kernels with respect to the k^{th} node ($\hat{H}^{(i)}_k$)—scalar multiplied by $1/\sqrt{d_j d_k}$—for each node k to which j is edge-connected in G. The term $1/\sqrt{d_j d_k}$ in Equation 4.3 ensures that pairs of low-degree nodes pass more information to each other than high-degree/low-degree pairs, which in turn pass more information to each other than high-degree/high-degree pairs. The idea here is that high-degree nodes have many connections, and so the information that they pass is less important: in the extreme case, a node n that is connected to *every* other node in the graph would cause all nodes to converge towards the same representation—which would then be useless for classification—without inverse-degree weighting.

4.1.2.2 GCNs Aggregate Local Neighborhoods

Equation 4.3 demonstrates that, within a single GCN layer, each node aggregates information from its local neighborhood (weighted by degree): each layer only permits message-passing between pairs of edge-connected nodes. However, by stacking multiple GCN layers together,

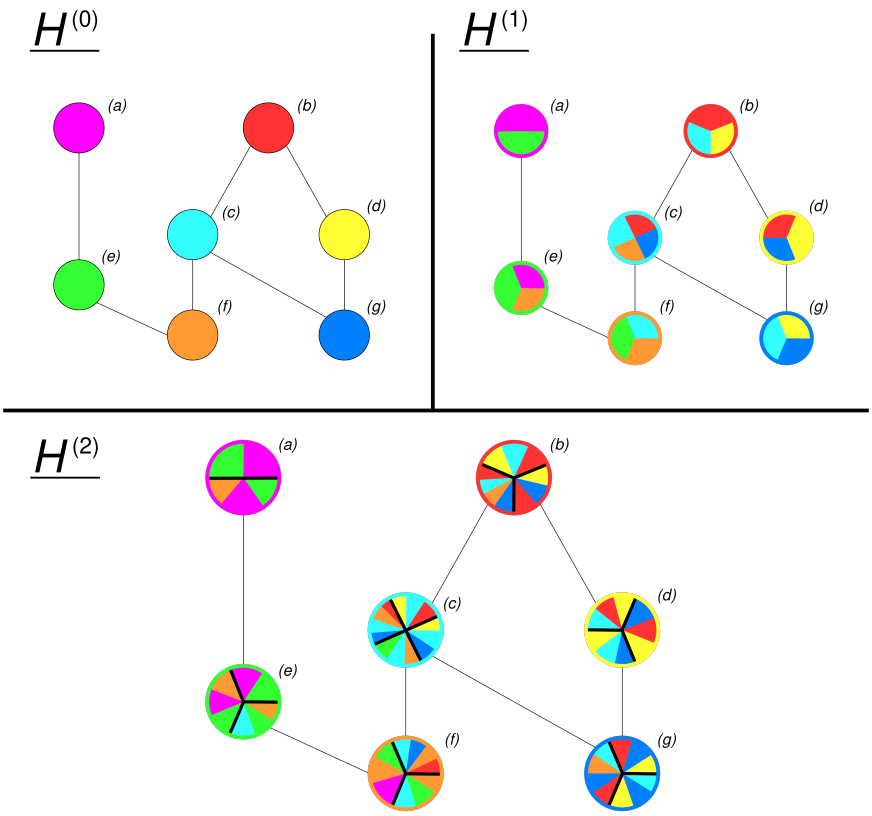


Figure 4.4: An illustration of L^{th}-order neighborhood aggregation in an L-layer GCN ($L = 2$), where node information (i.e. embeddings) is represented by colors. The initial node embeddings $H^{(0)}$ are illustrated in the top left figure, the output of the first GCN layer $H^{(1)}$ in the top right figure, and the output of the second layer $H^{(2)}$ in the bottom figure. Inverse degree-weighting is not illustrated: any proportional difference in size between colored regions within nodes is not intended to be meaningful.

a node can pass information to (and receive information from) nodes that are further away in the graph structure. Concretely, a GCN with L layers allows a given node x to exchange information with all nodes in its L^{th}-order neighborhood: the set of nodes y connected to x via a path p: $y \Rightarrow x$ such that the length of p is less that or equal to L.

This is illustrated in Figure 4.4. Nodes are first assigned initial embeddings $H^{(0)}$ (top left); in the output $H^{(1)}$ of the first GCN layer (top right), each node aggregates information from its immediate neighborhood. For example, node c (teal) aggregates information from nodes b (red), f (orange), and g (blue).

In the second GCN layer, each node again only aggregates information from its immediate

neighborhood. However, as each node's neighbor has aggregated from their respective neighborhoods in the first layer, that information is then included in the node's representation in the output of the second layer $H^{(2)}$ (bottom of Figure 4.4).

For example, node c again aggregates information from b, f, and g; however, in the first layer, b and g incorporated information from d (yellow), and f from e (green). Therefore, when c aggregates from $H_b^{(1)}$, $H_g^{(1)}$, and $H_f^{(1)}$, its final representation $H_c^{(2)}$ includes $H_d^{(0)}$ and $H_e^{(0)}$ as well (illustrated by the yellow and green slices—respectively—in the node coloring of $H_c^{(2)}$ in Figure 4.4). Observe that $H_c^{(2)}$ does *not* include any information from a (purple) in Figure 4.4, as a is three hops away from c in the graph structure.

Note that this L-layer aggregation permits structural information about a given node's L^{th} order neighborhood beyond a simple "bag of nodes". For example, $H_c^{(2)}$ includes $H_b^{(0)}$ *and* $H_b^{(1)}$, indicating that b is an immediate neighbor of c. On the other hand, $H_c^{(2)}$ only includes $H_d^{(0)}$, which signals that d lies two hops away from c in the graph[6].

Additionally, $H_c^{(2)}$ includes $H_d^{(0)}$ *twice* in Figure 4.4—i.e. there are two yellow slices in the node coloring of $H_c^{(2)}$—which indicates that two nodes in the immediate neighborhood of c are edge-connected to d (b and g): this effectively corresponds to a double-magnitude $H_d^{(0)}$ vector in the second-layer representation $H_c^{(2)}$ of c (see Equation 4.3). In contrast, $H_c^{(2)}$ only includes $H_e^{(0)}$ once—i.e. there is one green slice in its node coloring in Figure 4.4—indicating that only one immediate neighbor of c is edge-connected to e.

This layer-constrained information-exchange depth is a property of all such message-passing GNNs, including Gated GNNs (Li et al., 2016), GraphSAGE (Hamilton, Ying, and Leskovec, 2017), Graph Attention Networks (GATs; Veličković et al., 2018), and so on. The performance of these GNNs decreases considerably as the number of layers, and therefore the message-passing distance, increases (Wu et al., 2021). This is in fact desirable for learning on large graphs such as citation networks and knowledge graphs (Chami et al., 2022), where local neighborhoods are more meaningful than global structure. However, I argue in Section

[6]More generally, for an L-layer GCN, a node x, and another node y with (shortest) distance $\ell \leq L$ from x in the graph: $H_x^{(L)}$ includes $H_y^{(i)}$ for all $0 \leq i \leq L - \ell$.

4.1.2.3 that an architecturally-determined, fixed message passing depth is not beneficial to language modeling over semantic representation graphs such as DMRS structures.

4.1.2.3 Graph Transformers

While a language model over semantic representation graphs such as GFoLDS must (obviously) be sensitive to graph structures in order to capture co-occurrence relations between predicate nodes, the model must also allow for the possibility of *disconnected* co-occurrence: co-occurrence relations that occur between nodes that are not necessarily path-connected (or are simply located far away from one another) in a given input graph.

To illustrate this point, consider the sentence "*the woman crossed the street because she needed to get to the bus stop*." Clearly, there is a relationship between *the woman* and *she* (aside from the obvious co-indexicality): the presence of *the woman* as the subject of the first clause increases the likelihood of *she* appearing as the subject of the second clause, and vice-versa. However, the nodes corresponding to *woman* and *she* in any (D)MRS graph representation of this sentence are *not* connected by any directed path.

Therefore, the model architecture should be *sensitive* to the graph structure, but not entirely reliant upon it. This is to say that the architecture must permit *global* message-passing—message-passing between nodes that are not connected in the graph structure—but such message-passing should still be informed by the graph structure.

Wu et al.'s (2021) *graph transformer* paradigm presents an elegant solution to the problem of permitting global message-passing in graph structures, by coupling a message-passing GNN to an encoder transformer (Vaswani et al., 2017). The GNN assigns a local-neighborhood-aware embedding to each node, and these node embeddings are then passed to the transformer, which enables global attention (i.e. message passing). In contrast to other adaptations of transformer architectures to graph data (e.g. Wang, Wan, and Jin, 2020; Zhang et al., 2020a; Dwivedi and Bresson, 2020, etc.; see Section 4.1.3) that constrain the transformer's attention mechanism such that nodes can only attend to their respective local neighborhoods—thereby

restricting the model to local message passing—the transformer encoder in Wu et al.'s (2021) paradigm attends over all nodes in the graph: none of the nodes are attention-masked from one another in the encoder, so the degree to which they attend to each other is instead merely parameterized by their respective locations in the graph structure, by virtue of the preceding GNN. This grants the architectural flexibility required for the model to learn (for example) disconnected co-occurrence relations between tokens that reliably predict the presence of one another in certain structural configurations, despite not actually being path-connected within any DMRS graph.

Furthermore, Wu et al. (2021) omit linear positional embeddings—such as those employed in Vaswani et al. (2017) and other superficial NLP transformer architectures—in the transformer encoder: the GNN component of the architecture generates a representation of each node that encodes its position relative to that of its neighbors in the graph. By omitting positional encodings, the transformer encoder is invariant with respect to the linear order of the node embeddings—an appropriate design choice to model the set (rather than sequence) of nodes in a graph.

Wu et al. (2021) employ graph transformers for biological and chemical (i.e. molecule graph) classification tasks, utilizing a [CLS] token (as in BERT's next sentence prediction task; Devlin et al., 2019) to pool over the entire graph to predict the corresponding molecule's biochemical properties. Critically, however, the graph transformer is a *paradigm*, rather than a specific architecture: a graph transformer is simply a GNN whose node embeddings are passed to a transformer encoder for global attention. The particular GNN[7] to be employed is left open to best fit the task at hand, permitting the use of specialized GNN architectures adapted to (for example) labeled, directed graphs such as the DMRS structures discussed in Section 4.1.1 above.

I adopt the graph transformer paradigm for GFoLDS due its ability to permit global

[7]In fact, Wu et al. (2021) leave the choice of *transformer* open as well, allowing for the use of more compute- and memory-efficient transformers (e.g. Wu et al., 2020; Kitaev, Kaiser, and Levskaya, 2020; Choromanski et al., 2021, etc.) with larger graphs.

message-passing between nodes—a property that is necessary for language modeling over graph representations of logical forms (as argued above). Furthermore, this paradigm's above-mentioned flexibility with respect to the choice of GNN facilitates its adaptation to DMRS-derived graphs.

In Section 4.1.3 below, I review prior applications of graph representations and GNNs to NLP, with the goal of further motivating the particular implementation of the graph transformer paradigm that I employ in the GFoLDS model (discussed in Section 4.2)

4.1.3 GNNs for NLP

This subsection surveys the use of graph structures and GNNs to model language data. Section 4.1.3.1 overviews what I refer to as *task-specific* architectures: specialized, graph-based models that are designed to be employed for a single task (or family of related tasks), and cannot be applied more generally to a wider range of NLP domains. In Section 4.1.3.2, I discuss NLP models that incorporate knowledge graphs to improve performance on a broader variety of classification and generation tasks.

Section 4.1.3.3 then describes architectures that infuse linguistic structures—in particular, syntactic dependency graphs and semantic representations—into superficial models, while Section 4.1.3.4 surveys encoder-decoder, graph-to-text models that are employed for syntax-based neural machine translation and decoding semantic representations back into natural language.

Finally, I dedicate Section 4.1.3.5 to a description of Functional Distributional Semantics at Scale (FDSAS; Lo et al., 2023)—an extension of the FDS (Emerson, 2018) model described in Chapter 3—which represents the graph-based NLP model that is most directly comparable to GFoLDS.

4.1.3.1 Task-Specific Models

A large proportion of the task-specific models are developed for *relation extraction* tasks, where the objective is to determine the relationships[8] (if any) that exist between pairs of entities in a sentence (or body of text in general). All such models take as input syntactic dependency graphs, typically generated by an off-the-shelf parser such as the *SpaCy DependenceyParser*[9].

Guo, Zhang, and Lu (2019), Hu et al. (2021), and Zong et al. (2021) employ (variants of) the GCN architecture for relation extraction. In an attempt to overcome the global message passing problem described in Sections 4.1.2.2 and 4.1.2.3, Hu et al. (2021) and Zong et al. (2021) use the outputs of an LSTM and BERT (respectively) over the surface text as the initial embeddings for the GCN model (see Section 4.1.2.1). However, the use of the LSTM/BERT *before* the GCN means that these models can only use superficial cues to parameterize unbounded message-passing, unlike the graph transformer paradigm (see Wu et al., 2021 and Section 4.1.2.3), in which local graph neighborhood structure directly impacts global attention.

Although Guo, Zhang, and Lu's (2019) *Attention-Guided* GCN (AGGCN) employs global attention in a similar manner to the graph transformer, global attention occurs *before* graph structure is incorporated into the node embeddings—in other words, the global attention mechanism in the AGGCN is effectively over a bag-of-words representation of the input sentence.

Zhang, Ning, and Huang (2022) develop the *Syntax-guided Graph Transformer* (SGT) model for temporal event relation classification (i.e. predicting the pairwise temporal order of events), which takes as input a syntactic dependency parse graph of the input sentence, along with the output of a pretrained BERT model to initialize the node embeddings. While many other graph-adapted transformer variants (e.g. Wang, Wan, and Jin, 2020; Zhang et al.,

[8]These are roughly defined in terms of semantic roles (see e.g. Van Valin, 1999): for example, the relation *entity-destination* exists between the agent and goal of a predicate, such as between *boy* and *bed* in "*the boy went to bed*"

[9]https://spacy.io/api/dependencyparser

2020a; Dwivedi and Bresson, 2020) incorporate graph structure by constraining attention such that each node can only attend to other nodes within its respective local neighborhood, the SGT model permits unbounded message-passing by permitting each node to attend to each node to which it is *path-connected*, using a scalar value to encode distance—i.e. the length of the path. However, although it is *unbounded*, message-passing in the SGT model is not *global*: nodes cannot attend to other nodes that are not located within the same connected subgraph. As discussed in Section 4.1.2.3, such a limitation is undesirable for the task of language modeling over logical forms.

For document-level entity relation classification tasks, Nan et al. (2020) employ a GCN over *document graphs*, a method which outperforms competing superficial approaches. The edges in these graphs represent coreference relations across sentences, and verb/preposition-argument dependencies within sentences. Similarly, Khot, Sabharwal, and Clark (2018) create a weakly-linguistically informed, graph-based model in which the nodes of the graph structures are sequences of text, and the edges between them denote syntactic roles: specifically, subject/object and prepositions. Notably, this model outperforms (at the time) SoTA superficial models on NLI and multiple-choice question-answering tasks.

The fact that Nan et al. (2020) and Khot, Sabharwal, and Clark (2018) outperform superficial models with linguistically-informed, graph-based approaches supports the general thrust of this dissertation, and the Accelerated Learning Hypothesis of Chapter 1 in particular. However, as the graphs utilized by these authors are merely weakly linguistically-informed and do not directly incorporate linguistic structures such as semantic representations or syntactic parse graphs, the architectural details of these models are not of particular relevance to the discussion at hand.

Plenz and Frank (2024) adapt *pretrained* language models (the authors specifically employ BERT, although any similar LM could be used) for relation classification over knowledge graph (KG) triples. The authors first verbalize KG relations, then tokenize the verbalization and convert the tokenized representation back into a graph according to linear order: for

example, the KG relation *poodle* $\xrightarrow{is\text{-}a}$ *dog* is verbalized as "*poodle is a dog*", tokenized, then converted back into a graph as *poodle* $\rightarrow$ *is* $\rightarrow$ *a* $\rightarrow$ *dog*. This process preserves node identity, so (for example) the KG subgraph *poodle* $\xrightarrow{is\text{-}a}$ *dog* $\xleftarrow{is\text{-}a}$ *labrador* is converted to a graph of the form *poodle* $\rightarrow$ *is* $\rightarrow$ *a* $\rightarrow$ *dog* $\leftarrow$ *a* $\leftarrow$ *is* $\leftarrow$ *labrador*. Plenz and Frank (2024) then use a combination of attention masking and the model's existing positional embeddings to represent these graph structures within the pretrained LM's architecture. The authors find that, due to the knowledge already incorporated into the LM via pretraining, this approach in fact outperforms specialized graph architectures trained from scratch on the KG relation-classification task: this highlights the benefit that pretraining yields for downstream applications, even in the graph domain.

4.1.3.2 Knowledge Graph Incorporation

While Plenz and Frank (2024) employ a superficial language model for classification over knowledge graphs, many other architectures (e.g. Zhou et al., 2018; Lin et al., 2019; Yasunaga et al., 2022) work in the other direction, and use knowledge graphs to improve performance on natural language tasks by giving the model access to an external database. As KGs are typically quite large—for example, ConceptNet 5.5 (Speer, Chin, and Havasi, 2017) contains ∼34 million edges—these architectures retrieve relevant KG subgraphs by detecting concept mentions within input sequences.

Lin et al. (2019) employ BERT and a GCN+LSTM GNN architecture over a knowledge graph to achieve SoTA performance on commonsense reasoning classification tasks. On the other hand, Zhou et al. (2018) use a GAT to incorporate KGs into an encoder-decoder GRU (Cho, 2014), in order to generate more informative responses in a conversational model.

The *Deep Bidirectional Language-Knowledge Graph Pretraining* (DRAGON; Yasunaga et al., 2022) model is of particular relevance to the current discussion. This model is an implementation of the GreaseLM (Zhang et al., 2022) architecture, which retrieves KG subgraphs from entity mentions in text, then fuses GAT-derived graph representations with

transformer encoder text representations at each layer of the transformer, which enriches the LM's input text representations with relevant external knowledge from the KG. DRAGON extends GreaseLM by *pretraining* the LM/GAT hybrid (Yasunaga et al., 2022 use RoBERTa as the language model) over text and KGs simultaneously. The model is pretrained for masked language modeling and knowledge graph link prediction, in order to learn to reason jointly over text and the KG.

As far as I am aware, DRAGON is the only existing graph-based NLP model—aside from GFoLDS and FDS/FDSAS—that is pretrained on graph representations[10]. This model demonstrates both the possibility and the utility of pretraining over graph representations: Yasunaga et al. (2022) show that DRAGON outperforms the baseline RoBERTa model on a variety of question-answering and commonsense reasoning tasks.

4.1.3.3 Linguistic Structure Infusion

The models discussed in Section 4.1.3.1 above demonstrate that linguistically-informed graph representations can be employed with specialized architectures to improve the SoTA on specific tasks, but are not designed to generalize to a wider range of downstream tasks. On the other hand, while the KG-aware models of Section 4.1.3.2—and DRAGON in particular—show that fusing graph representations in general into superficial models can improve model performance on a greater variety of tasks, they do not demonstrate that *linguistic structure* can yield task-agnostic model improvement.

To that end, Xu et al. (2021) fuse dependency parse graphs into pretrainined transformer encoders (e.g. BERT and RoBERTa) using a learnable scalar gating mechanism that is initialized near zero, thereby preventing the graph representations from interfering with the pretrained model weights. The authors employ *Syntax-Aware Attention* (SAA) over the dependency graphs: in a similar manner to the SGT (Zhang, Ning, and Huang, 2022) model discussed in Section 4.1.3.1, SAA permits each node to attend to each node to which it is

[10]However, unlike GFoLDS and FDS/FDSAS, DRAGON is pretrained on both (knowledge) graphs *and* text, simultaneously.

path-connected (scaled by distance). Unfortunately, SAA suffers from the same drawback as SGT—namely, this is not truly global attention. However, this model is still able to achieve SoTA results on relation classification, entity typing, and question answering tasks, demonstrating the general utility of linguistically-informed LMs. Similarly, Wu, Peng, and Smith (2021) fuse syntactic dependecy parse graphs into a pretrained BERT model with a standard GAT, yielding SoTA results on semantic role labeling and relation extraction tasks.

While the results of Xu et al. (2021) and Wu, Peng, and Smith (2021) demonstrate that linguistically-informed LMs can outperform their purely superficial counterparts, both of these models employ syntactic parse graphs, and therefore do not directly demonstrate the value of language modeling over *semantic* representations.

However, Prange, Schneider, and Kong (2022) show that linguistic structures can be incorporated into the input of GPT-2 (Radford et al., 2018) to improve the model's accuracy and entropy on next word prediction—critically, the authors experiment with both syntactic parse graphs and semantic representations. These graph representations are injected into the GPT-2 model as follows: a subgraph of the structure in question—i.e. either a syntactic parse or semantic representation—corresponding to the portion of the sentence that GPT-2 has already generated is embedded into the model's embedding space via GNN, then included into the input sequence. Of particular interest to the discussion at hand is Prange, Schneider, and Kong's (2022) finding that Elementary Dependency Structures (Oepen and Lønning, 2006)—an MRS-derived semantic framework that is related to DMRS—yield greater performance improvements than syntactic (or other semantic) representations.

In a similar vein, Wu, Peng, and Smith (2021) compare the performance of syntactic and semantic representations with respect to a linguistically-informed LM over graph representations. This model employs a Relational GCN (Schlichtkrull et al., 2018) over syntactic and semantic graphs whose node representations are initialized from a pretrained RoBERTa model over the surface text. The author's findings corroborate those of Prange, Schneider, and Kong (2022)—and extend their results to encoder transformers—by demonstrating that

semantic representations are more beneficial than syntactic structures for pretrained LMs. However, as with the architectures of Hu et al. (2021) and Zong et al. (2021), this model's use of RoBERTa before the GCN means that global attention can only be parameterized by the surface text, rather than the graph structure.

More generally, all of the models described thus far in this subsection are effectively hybrid architectures that merge superficial and graph representations. Furthermore, in the vast majority of these approaches, the superficial component of the model is initialized from a pretrained LM such as BERT, RoBERTa, or GPT-2. This contrasts with the GFoLDS model (see Section 4.2), which takes only graph representations as input and is pretrained from scratch.

4.1.3.4 Graph-to-Text Models

Most graph-to-text models, on the other hand, are trained from scratch, and by definition only take graphs as input. These models are typically encoder-decoder architectures, where the encoder encodes the input graph into an embedding space, and the decoder decodes that representation into text. For obvious reasons, the encoder portions of these architectures are of particular relevance to the topic at hand, and will be the primary focus of discussion.

While Hajdik et al.'s (2019) DMRS-to-text model is of interest due to their employment of the same semantic representation framework as GFoLDS, these authors linearize the DMRS structures in PENMAN notation (Goodman, 2020) in order to convert them into textual inputs that are recognizable by their sequential (LSTM) model. As discussed in Section 4.1.2.3 above, textual representations are disadvantageous for language modeling over logical forms.

On the other hand, Guo et al. (2019) use a Densely-Connected GCN encoder with an LSTM decoder for AMR-to-text (Abstract Meaning Representation—a graph-based semantic representation format; Banarescu et al., 2013) and syntax-based machine translation tasks. Similarly, Zhang et al. (2020b) use a Lightweight Dynamic GCN encoder with an LSTM

decoder for AMR-to-text generation. While both of these models outperform superficial SoTA models, their use of GCN encoders precludes global message-passing.

Wang, Wan, and Jin (2020) and Wang, Wan, and Yao (2020) employ an encoder-decoder transformer architecture for AMR-to-text generation, and adapt the transformer encoder to graph data by constraining the attention mechanism to prevent nodes from attending to other nodes outside of their respective local neighborhoods (as discussed in Section 4.1.2.3). As with other, local-message-passing GNNs, this approach limits a given node's message-passing radius to its L^{th}-order neighborhood, where L is the number of encoder layers. While they outperform superficial models over linearized AMR graphs, Wang, Wan, and Jin (2020) note that their model's performance degrades as the depth (i.e. the length of the longest path) of the input graph increases beyond the number of encoder layers; it is clear that unbounded message passing is necessary for language models over graph representations of logical forms.

In their approach to AMR-to-text generation and syntax-based machine translation, Beck, Haffari, and Cohn (2018) attempt to overcome the unbounded message-passing problem with the use of a Gated GNN (Li et al., 2016): a recurrent architecture that allows the network to dynamically adapt its message-passing radius to the depth of the input graph. Unfortunately, recurrent neural networks are known to suffer from exploding and vanishing gradients (Hanin, 2018), which results in greater training instability in comparison to non-recurrent architectures such as transformers.

In a similar vein, Nguyen et al. (2020) attempt to overcome the unbounded message-passing problem with respect to syntax-based machine translation via the attention-based *hierarchical accumulation* mechanism. However, these authors apply hierarchical accumulation to syntactic constituency parses (i.e. trees) obtained from the *Stanford CoreNLP* (Manning et al., 2014) parser—hierarchical accumulation is *only* applicable to tree data, and not to graphs in general—in particular, DMRS graphs are not trees.

4.1.3.5 Functional Distributional Semantics at Scale

As mentioned earlier in this section, Functional Distributional Semantics at Scale (FDSAS; Lo et al., 2023) extends the Functional Distributional Semantics model (FDS; Emerson, 2018) discussed in Chapter 3. Like FDS (and GFoLDS), FDSAS is a distributional model over DMRS structures, but replaces the binary-valued individual (i.e. entity and event) embeddings and Cardinality-Restricted Boltzmann Machine (Swersky et al., 2012) of FDS with real-valued embeddings and a variational autoencoder (VAE; Higgins et al., 2017).

As with FDS, FDSAS begins with a DMRS-derived probabilistic graphical model M (see Figure 4.5). Each variable x_i in M—represented by the predicate $\mathcal{I}(x_i)$ of which x_i is the intrinsic variable—is assigned a contextual embedding $\vec{e_i}$ as a function of the set of predicates that take x_i as an argument (including $\mathcal{I}(x_i)$), and the set of predicates $\mathcal{I}(x_k)$ such that $\mathcal{I}(x_i)$ takes x_k as an argument. The VAE encoder then produces mean and variance vectors μ_i and σ_i^2 from $\vec{e_i}$, which parameterize a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. The idea here is that rather than learning to embed predicates as points in space based on their distribution, FDSAS instead learns to embed predicates as probabilistic *regions*.

The model draws a representation $\vec{z_i} \sim \mathcal{N}(\mu_i, \sigma_i^2)$ of x_i that is then passed to the VAE decoder, which consists of a set of unary and binary logistic classifiers $t^{(r)}(-)$ and $t^{(r,a)}(-,-)$ (respectively; see Figure 4.5). As FDSAS takes a Neo-Davidsonian (event semantics; Dowty, 1989) perspective[11], the unary classifiers $t^{(r)}(-)$ correspond to predicates in the DMRS structure, while the binary classifiers $t^{(r,a)}(-,-)$ correspond to argument roles.

For example, in Figure 4.5, x_1 is the intrinsic variable of the predicate *postman*, so (assuming that the model has learned correctly) the value of $t^{(postman)}(\vec{z_1})$ should be higher than $t^{(r)}(\vec{z_1})$ for any other predicate r. As x_1 is the first-place argument of the predicate *deliver*, the value of $t^{(deliver,1)}(\vec{z_1}, \vec{z_2})$ should be higher than $t^{(r,a)}(\vec{z_1}, \vec{z_2})$ for any other predicate r and argument label a—where the event-type variable x_2 is the intrinsic variable of *deliver*

[11] i.e. predicates that are traditionally considered to be ≥2-ary are represented to be unary predicates taking a single event-type argument, while binary predicates over entities and events express argument labels. For example, the sentence "*Jane sees Mary*" is represented as: $\exists e[see(e) \wedge arg_1(e, jane) \wedge arg_2(e, mary)]$.

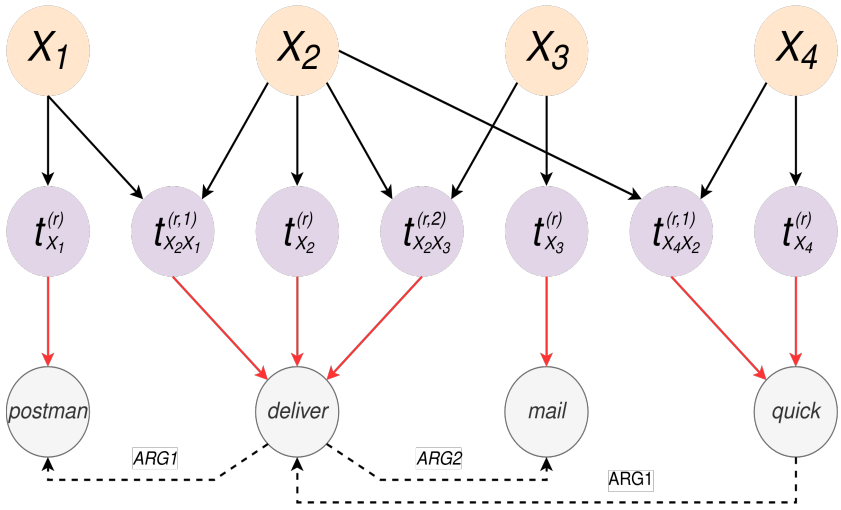


Figure 4.5: An FDSAS probabilistic graphical model corresponding to the sentence "*a postman delivers mail quickly*". Solid, black arrows $x_i \rightarrow t^{\rho}_{\chi}$ indicate that the classifier t^{ρ}_{χ} takes the (sampled representation $\vec{z}_i$ of the) variable x_i as an argument, and red arrows $t^{\rho}_{\chi} \rightarrow R$ indicate that the DMRS predicate R is the target classifier for t^{ρ}_{χ}. Dashed, labeled arrows correspond to the underlying DMRS argument structure (expressed via the binary classifiers $t^{(r,a)}$): $R \xrightarrow{\ell} P$ indicates that (the intrinsic variable of) the predicate P is the ℓ^{th}-place argument of R.

(see Figure 4.5).

The use of these binary, Neo-Davidsonian-esque argument role classifiers in FDSAS lends flexibility to this model, allowing it to take as input a wider range of DMRS structures than FDS, including those containing instances of adjectives, adverbs, and conjunction. Furthermore, despite using the exact same training dataset as FDS, FDSAS massively outperforms the former model, more than doubling the performance of FDS on the RELPRON (Rimell et al., 2016) and GS2011 (Grefenstette and Sadrzadeh, 2011) datasets.

However, this model still faces serious limitations: namely, it is unable to account for quantifiers, higher-order verbs[12], and disjunction—such structures and predicates are simply removed from its training dataset during preprocessing. GFoLDS, on the other hand, can take effectively any DMRS structure (subject to certain constraints; see Section 4.3.1) as input, including those containing quantifiers, higher-order verbs, and coordination structures.

[12]i.e. sentential-complement verbs such as *want*, *need*, *hope*, etc.

4.2 GFoLDS Architecture

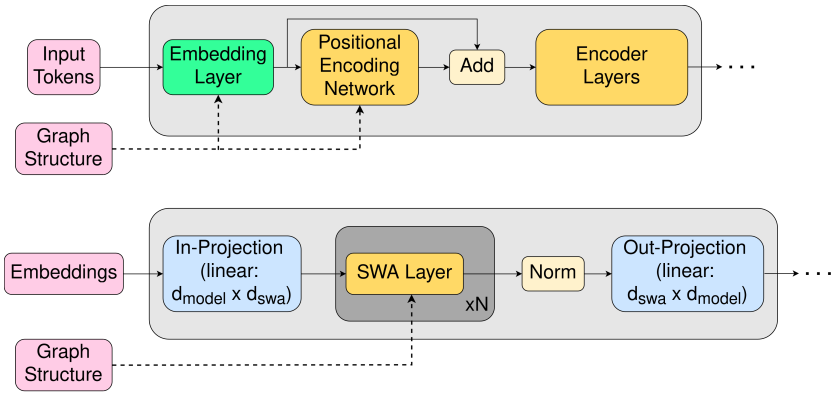


Figure 4.6: Top-level architecture of the GFoLDS model (top) and positional encoding network (bottom). The GNN component of GFoLDS consists of the embedding layer, positional encoding network, and skip connection (the "Add" block in the top figure).

As discussed in Section 4.1.2.3, the GFoLDS model is an implementation of the graph transformer paradigm of Wu et al. (2021): a GNN that encodes local node neighborhood information, whose output is then fed to a permutation-invariant (i.e. without linear positional embeddings) transformer encoder for global message-passing (attention). The GNN component of GFoLDS consists of an embedding layer and *positional encoding network* (see Figure 4.6). The output of the embedding layer is fed into the positional encoding network, which is intended to provide each node with a representation of its local neighborhood in the DMRS graph structure.

The input to the encoder stack is the sum of the respective outputs of the embedding layer and positional encoding network. This is analogous to (and inspired by) the approach taken by most superficial transformer LLMs (e.g. Devlin et al., 2019; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Dubey et al., 2024, etc.), in which the input to the encoder stack for a given token t at position p is $\vec{t_{con}} + \vec{p_{pos}}$: the sum of the content embedding for t ($\vec{t_{con}}$) with the positional embedding for p ($\vec{p_{pos}}$). This architectural design choice has the further benefit of creating a residual/skip connection—which are known to significantly boost the performance of deep neural networks (He, Liu, and Tao, 2020; Godbole et al.,

2023)—between the embedding layer and the output of the positional encoding network.

As discussed in Section 4.1.2.3, the positional encoding network generates a representation of each node that encodes its location relative to that of its neighbors in the graph. As mentioned above, these position-aware node representations are then sent to the transformer encoder: formally, the input to the encoder stack is $E(X, G) + P(E(X, G), G)$, where E denotes the embedding layer, P the positional encoding network, X the input tokens (i.e. node labels), and G the graph structure. The encoder is *not* directly exposed to the graph structure: all nodes are able to attend to any other node(s)—the degree to which they attend to one another is merely informed by their graph-aware positional encodings (again, as discussed in Section 4.1.2.3).

4.2.1 Embedding Layer

For the i^{th} node n_i, the output of the embedding layer $\vec{e_i} = E(X, G)_i$ is defined in Equation 4.4 below, where $F(n_i)$ denotes the set of properties/features for the node n_i, $\mathcal{E}_T$ the token/node embedding layer[13], and $\mathcal{E}_F$ the feature embedding layer (which maps each feature to a d_{model}-dimensional vector).

$$\vec{e_i} = \mathcal{E}_T(n_i) + Norm\left(\sum_{\phi \in F(n_i)} \mathcal{E}_F(\phi)\right) \tag{4.4}$$

In words: for a node n_i with features $F(n_i)$, the embedding layer outputs the sum of the embedding of the node's label (i.e. predicate) with the layer-normalized sum of the embeddings of each feature in $F(n_i)$. I included the layer-normalization in Equation 4.4 in order to prevent the feature embeddings—which can number up to six—from "drowning out" the single predicate embedding.

For example, the output of the embedding layer for the node *see_ v_ 1* in Figure 4.3 is the sum of $\mathcal{E}_T(see\_\ v\_\ 1)$ with the normalized sum of the embeddings of *SF:prop*[14], *TENSE:past*,

[13]A standard PyTorch embedding layer that maps each node label/predicate to a d_{model}-dimensional vector.
[14]Indicates that the event is propositional/declarative, as opposed to e.g. interrogative (*SF:ques*).

MOOD:indicative, *PERF:−* (not perfective), and *PROG:−* (not progressive).

4.2.2 Positional Encoding Network

These summed feature and node/predicate embeddings are then passed to the positional encoding network (see Figure 4.6). This module consists of a linear layer that projects the embeddings from d_{model} to d_{SWA}, followed by a stack of *step-wise aggregation* (SWA) layers (see Figure 4.7)—in which the output of each SWA layer is fed to the subsequent layer—that is in turn followed by a second linear projection that projects from d_{SWA} back to d_{model}.

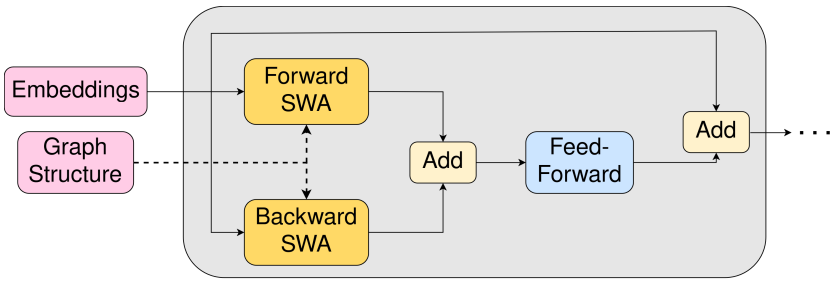


Figure 4.7: Architecture of an SWA layer in the positional encoding network.

In their respective adaptations of the GraphSAGE (Hamilton, Ying, and Leskovec, 2017) and 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 contains a forward (Equation 4.5a) and a backward (Equation 4.5b) SWA block, which encode the nodes (i.e. predicates)—and the semantic roles thereof—mapping into and out of a given node.

$$\vec{f_i} = Norm\left(\sum_{n_k \xrightarrow{\ell} n_i \in E} W_\ell^{(f)} \vec{x_k}\right) \tag{4.5a}$$

$$\vec{b_i} = Norm\left(\sum_{n_i \xrightarrow{\ell} n_k \in E} W_\ell^{(b)} \vec{x_k}\right) \tag{4.5b}$$

For a given node n_i, its forward representation $\vec{f_i}$—i.e. the output of the forward SWA block—is the layer-normalized sum of $W_\ell^{(f)}\vec{x_k}$ for each node n_k with an edge $n_k \xrightarrow{\ell} n_i$ in the graph structure, where $W_\ell^{(f)}$ 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, Haffari, and Cohn'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 layers $W_\ell^{(b)}$ and operates on the transpose of the graph (i.e. with all edges reversed). Letting L denote the number of DMRS edge labels[15]—corresponding to semantic roles e.g. *ARG1*, *ARG2*, *RSTR*, etc.—each SWA layer contains $2L$ edge projection layers of dimension $d_{SWA} \times d_{SWA}$: one forward and one backward projection for each label. The positional encoding network therefore constitutes a large percentage of the overall model's parameters: ∼38% of parameters of the GFoLDS model used in this work belong to the positional encoding network.

For a concrete example of the application of an SWA layer to an individual node, consider the DMRS graph in Figure 4.1/4.3. In this case, the node *see_v_1* receives the forward representation $\vec{f_{see\_v\_1}}$ (Equation 4.6a), which encodes the fact that *loc_nonsp* takes *see_v_1* as its first-place argument. Its backward representation (Equation 4.6b), $\vec{b_{see\_v\_1}}$, encodes the fact that *see_v_1* takes *person_n_1* and *named* (corresponding to the entity *Mary*) as its first- and second-place arguments, respectively.

$$\vec{f_{see\_v\_1}} = Norm(W_{ARG1}^{(f)}\vec{loc\_nonsp}) \tag{4.6a}$$

$$\vec{b_{see\_v\_1}} = Norm(W_{ARG1}^{(b)}\vec{person\_n\_1} + W_{ARG2}^{(b)}\vec{named}) \tag{4.6b}$$

[15] $L = 8$ in the model used in this work—see Section 4.3.2 and Table 4.1.

The respective forward ($\vec{f_i}$) and backward ($\vec{b_i}$) representations of each node n_i are then summed together and passed through a feed-forward module: this is inspired by the use of feed-forward modules after the attention block in transformer encoder layers (see Figure 4.8), as introduced in Vaswani et al. (2017). The feed-forward modules in each SWA layer are identical to those in BERT's (Devlin et al., 2019) encoder layers: they consist of a $d_{SWA} \times 2d_{SWA}$ linear layer, followed by GELU activation (Hendrycks and Gimpel, 2016) and a $2d_{SWA} \times d_{SWA}$ linear layer.

The feed-forward layer is then followed by a skip connection: the output of the SWA layer for the node n_i is $FF(\vec{f_i} + \vec{b_i}) + \vec{x_i}$, where $\vec{x_i}$ denotes the input to the SWA layer for n_i—i.e. the output of the previous SWA layer. I included this skip connection in each SWA layer because—as discussed above—it is well-known that residual connections drastically improve the performance of deep neural networks (He, Liu, and Tao, 2020; Godbole et al., 2023).

As in GCNs and similar message-passing GNNs, each SWA layer aggregates local node neighborhoods: as discussed in Section 4.1.2, stacking L SWA layers together allows each node to aggregate information from nodes within its L^{th}-order neighborhood. For example, if the model has two SWA layers, the forward representation of *see_v_1* in the second SWA layer encodes the fact that the *loc_nonsp* node that takes *time_n* as its first place argument takes *see_v_1* as its second-place argument. Similarly, its backward representation in the second layer encodes the fact that the *person_n_1* node that *see_v_1* takes as its first place argument is *also* taken by *bored_a_1* as its first-place argument, and is in the restriction of the *every_q* quantifier (and that the *named* second-place argument of *see_v_1* is in the restriction of *proper_q*).

This is to say that more SWA layers permit more widely-scoped representations (i.e. that aggregate larger neighborhoods) of each node's location in the graph—at the cost of a larger model, of course—which allows the downstream transformer encoder layers to more easily distinguish between individual nodes' locations in the structure.

4.2.3 Encoder Stack

As discussed above, 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.

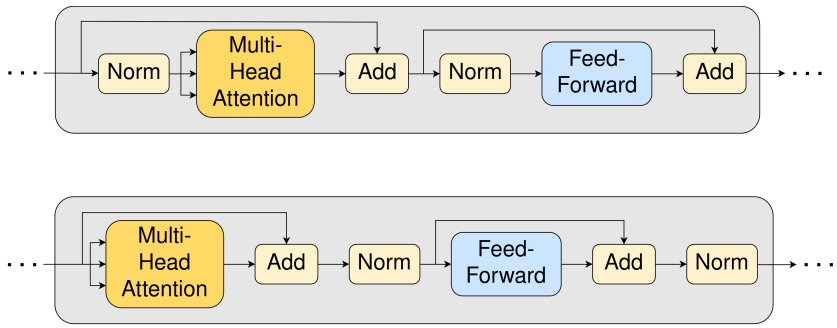


Figure 4.8: Architecture of a GFoLDS (top) and BERT (bottom) encoder layer.

The encoder layers in the GFoLDS architecture are similar to those in BERT and Vaswani et al. (2017)[16], but contain a few key differences—in particular with respect to the residual connections and interrelated layer normalization. As shown in Figure 4.8, 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—that is identical to a BERT encoder feed-forward layer— and a second skip connection; this skip connection is *not* followed by layer normalization.

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

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

residual (i.e. $x + f(Norm(x))$), which I implement in GFoLDS.

4.3 Data Preprocessing

The procedure that I used to convert a textual corpus (see Section 4.4 for a discussion of the particular corpus employed in these experiments) into a collection of DMRS-derived graph inputs for the GFoLDS model is similar to that of Chapter 3: I first used Spacy's *SentenceRecognizer*[17] sentence-segmentation pipeline to extract individual sentences from the text. I then used the *PyDelphin* (Goodman, 2019) library with the *ACE/ERG* parser/grammar[18] (Copestake and Flickinger, 2000) to obtain a DMRS representation of each sentence, before preprocessing the resulting DMRS structures to yield GFoLDS input graphs.

4.3.1 CARGs and OOV Items

There are two major (related) preprocessing steps that I took regarding the DMRS graphs. First, I replaced all OOV (out-of-vocabulary) terms—as a rule-based parser, ACE/ERG has a fixed vocabulary—with the [MASK] token. These [MASK] tokens are *not* targets for prediction during the MNM pre-training procedure, as they are OOV and so there is no possible target token: the goal is instead to have the model try to represent the OOV item with the closest in-vocabulary token, based on the context in which the OOV item appears.

The second preprocessing step involves the CARG-bearing ("constant argument") predicates: those predicates corresponding to seasons, numbers, dates, named entities—e.g. *named* in Figure 4.1/4.3—etc. Each such constant argument is represented in DMRS by a CARG-bearing predicate, which has an additional argument slot (*CARG*) that is filled by the string denoting the constant in question: for example, in Figure 4.1/4.3, the *CARG* of the predicate *named* is *"Mary"* (which is removed in Figure 4.1, due to the preprocessing described in this section).

[17] https://spacy.io/api/sentencerecognizer
[18] ERG-1214 release: https://github.com/delph-in/erg

Because named entities (CARGs) and OOV items do not constitute a fixed vocabulary, it is not feasible to tokenize them in the same manner as for the DMRS predicates—i.e. by simply assigning an integer to each unique predicate string in the vocabulary. I therefore attempted to incorporate two different approaches (described below) to tokenizing these items in order to include them into the graph structure: given that BERT's WordPiece (Wu et al., 2016) tokenizer is designed to handle any unicode input string, both attempts to include CARGs and OOV items into the graph structure involved tokenizing these strings with the BERT tokenizer.

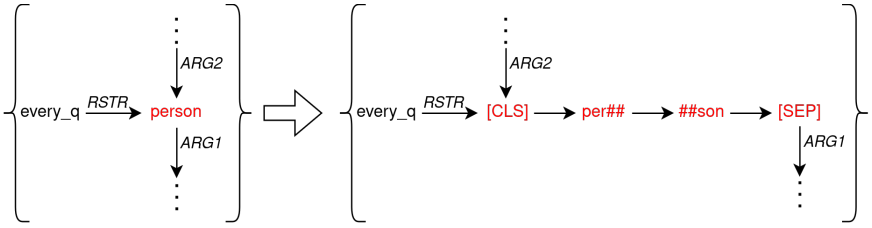


Figure 4.9: Illustration of my first attempt at incorporating CARGs and OOV items into the graph structure (assuming that *person* is OOV for the sake of example). The left-hand side represents what the graph *would* look like if *person* were in-vocabulary, while the right-hand side illustrates the attempted solution.

In the first such attempt (illustrated in Figure 4.9), given an OOV item or CARG X, I first used the BERT tokenizer to yield a sequence of tokens [CLS] $t_1 \dots t_n$ [SEP]. I then included that token sequence into the graph, connected by a sequence of edges according to the tokens' linear order, using a special edge label not used for DMRS semantic roles: [CLS] $\rightarrow t_1 \rightarrow \cdots \rightarrow t_n \rightarrow$ [SEP]. For any semantic role ℓ_1 "going into" X from some other predicate Y—i.e. indicating that X is taken as an argument of Y—I included an edge $Y \xrightarrow{\ell_1}$ [CLS] in the graph. Conversely, for any semantic role ℓ_2 "going out of" X to a predicate Z—i.e. X takes Z as an argument—I included an edge [SEP] $\xrightarrow{\ell_2} Z$ in the graph.

This attempt resulted in unreasonably high loss during pretraining and poor performance on downstream tasks. I hypothesized that it might have been too difficult for the model to follow a path $Y \xrightarrow{\ell_1}$ [CLS] $\rightarrow t_1 \cdots \rightarrow t_n \rightarrow$ [SEP] $\xrightarrow{\ell_2} Z$ and comprehend that the object that Y takes as its ℓ_1 argument is the same object that takes Z as its ℓ_2 argument.

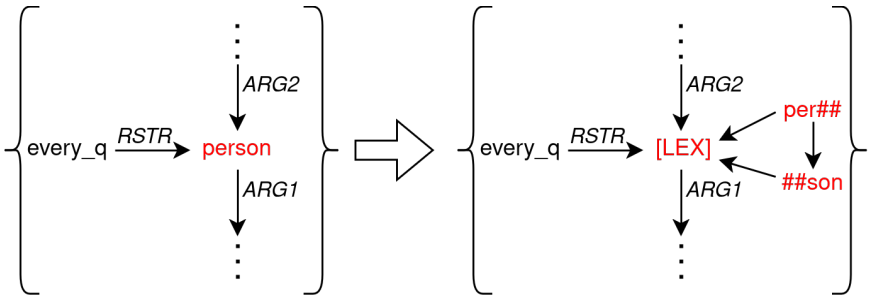


Figure 4.10: Illustration of my second attempt at incorporating CARGs and OOV items into the graph structure (again assuming that *person* is OOV for the sake of example). The left-hand side represents what the graph *would* look like if *person* were in-vocabulary, while the right-hand side illustrates the attempted solution.

Therefore, in the second attempt, I introduced a special token ([LEX]) that was intended to anchor the individual tokens of CARGs and OOV items (see Figure 4.10). Under this approach, I applied the BERT tokenizer to a given OOV item or CARG X, and discarded the leading and trailing (respectively) [CLS] and [SEP] tokens, yielding a sequence of tokens $t_1 \dots t_n$. I then incorporated this sequence into the graph structure by first connecting the tokens via a series of edges in the same manner as in the first approach (illustrated in Figure 4.9): $t_1 \rightarrow \cdots \rightarrow t_n$. This sequence was connected to the [LEX] token by including edges $t_i \rightarrow$ [LEX] $(1 \leq i \leq n)$ from each token node t_i to [LEX]. All semantic roles ℓ_1 "going into" X from Y were incorporated as edges $Y \xrightarrow{\ell_1}$ [LEX], and semantic roles ℓ_2 "going out of" X to Z as edges [LEX] $\xrightarrow{\ell_2} Z$.

As with the approach illustrated in Figure 4.9 above, this approach was not particularly successful, and resulted in poor performance on downstream tasks. I hypothesize that such approaches—i.e. including a single word as a sequence of tokens in the graph structure—fail because they result in two competing roles for the graph edges[19]: in some cases, graph edges denote semantic roles between predicates (e.g. *ARG1*, *ARG2*, etc.), while in other situations, the edges indicate that two or more tokens together comprise the same predicate label.

[19]For both approaches (i.e. those illustrated in Figures 4.9 and 4.10), I also attempted to treat *every* predicate as "OOV": tokenizing not only the strings of the CARGs/OOV items, but also the *in-vocabulary* predicates as well. I had hypothesized that a potential source of confusion for the model was arising from the mismatch between the two types of structures in the graph representations: the actual predicate nodes, and the OOV/CARG subgraphs that were effectively intended to be treated as individual nodes. Unfortunately, this approach did not yield significant improvement.

To avoid representing CARGs and OOV items as collections of multiple tokens within the graph structure itself, I also considered having a separate OOV/CARG embedding layer. The idea being that those items' individual tokens would be passed through that layer, then summed together to yield a mean-pooled representation which would then be included in the graph as a single node embedding. Due to GFoLDS' masked-node modeling pretraining objective (see Section 4.4.2), however, such an approach is not feasible: it is unclear how one would separate the individual tokens out of the OOV/CARG node embeddings outputted by the encoder stack in order to yield predictions when those items are masked.

As neither of the attempts to incorporate CARGs and OOV items into the graph structures were particularly successful—and as these items cannot be retained without causing the size of the vocabulary to explode—I ultimately decided to simply remove the CARGs themselves and keep only the CARG-bearing predicates: 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* [SEASON] *of* [YEAR]." As discussed in the beginning of this section, all OOV items are simply replaced with the [MASK] token—during both pretraining *and* downstream fine-tuning and inference.

The removal of CARGs and OOV items are stop-gap measures, and remain an open problem and barrier to the performance of the GFoLDS model. This being said, the research carried out in this dissertation is intended purely as a proof-of-concept of the viability of language modeling over logical forms (as discussed in Chapter 1): in order to retain this work's focus on that bigger picture, it was occasionally necessary to bypass smaller issues such as this, and leave their solution to future work. I therefore defer consideration of potential future avenues for the incorporation of CARGs and OOV items into DMRS graph structures to the discussion of future directions in Chapter 7.

4.3.2 Additional Preprocessing Steps

After removing CARGs and OOV items from the graph structures, there remain a few (relatively less significant) preprocessing steps that I took in order to transform DMRS

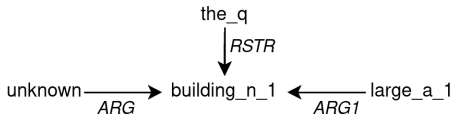


Figure 4.11: DMRS representation of the noun phrase "*the large building*".

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*[20] and an edge ARG: *unknown* $\rightarrow$ *building_n_1* (see Figure 4.11), 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*.

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 4.2.2). 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

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

parameters to its representation.

Therefore, during preprocessing, I converted each *ARG* label to the *MOD* label (see Table 4.1): 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, I equivalence-classed argument labels involved in coordination structures—i.e. those 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, I replaced the edge labels *L-HNDL*/*R-HNDL* and *L-INDEX*/*R-INDEX* with *HNDL* and *INDEX* (respectively; see Table 4.1), ignoring the surface order of the conjuncts. This preprocessing step had the added benefit of reducing the overall size of the GFoLDS model by $4n(d_{SWA})^2$ parameters, as discussed above.

Finally, I 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 (i.e. fronted; Szűcs, 2014) or the subject of a passivized verb (respectively). The inclusion of either of these predicates would run contrary to the discussion of the Accelerated Learning Hypothesis in Chapter 1 and the related arguments in Chapter 3: namely, that one of the primary advantages of language modeling over logical forms is the syntactic de-noising effect arising from the fact that such representations equivalence-class syntactic paraphrases such as topicalization and passivization.

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 4.1: 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).

4.4 Pretraining GFoLDS

In this section, I detail the procedure that I employed to pretrain the GFoLDS model for later application to downstream tasks in Chapter 5, beginning with an overview of the model's pretraining corpus in Section 4.4.1. I then describe the procedure and rationale behind the masked node modeling (MNM) objective (Section 4.4.2), before discussing the pretraining hyperparameters and architectural configuration (i.e. number of layers, etc.) in Section 4.4.3.

4.4.1 Corpus

To assemble the corpus, I randomly selected ∼17.5 million sentences from the November 1, 2023 English Wikipedia dump[21], which constitutes a total of ∼508 million words: ∼6.5 times smaller than BERT's pre-training corpus. The ACE/ERG parser was able to parse

[21]https://huggingface.co/datasets/wikimedia/wikipedia/viewer/20231101.en

∼84% of the data, for a total of ∼14.6 million DMRS-derived graphs to serve as the model's pretraining corpus.

After preprocessing (see Section 4.3), we are left with a total vocabulary size of 22,077 predicates (for comparison, BERT-uncased has a vocabulary of 30,522 tokens), 8 edge (i.e. argument; see Table 4.1) labels, and 31 feature-value pairs (number, person, tense, etc.; see Table 4.2), with ∼413 million nodes across the ∼14.6 million parsed DMRS graphs (∼28.3 nodes per graph on average).

Feature	**Value**
PERS	1, 2, 3
NUM	*PL*, *SG*
GEND	*F*, *M*, *M-or-F*, *N*
PT (pronoun type)	*REFL*, *STD*, *ZERO*
SF (sentence force)	*COMM*, *PROP*, *QUES*, *PROP-or-QUES*
TENSE	*FUT*, *PAST*, *PRES*, *TENSED*, *UNTENSED*
MOOD	*INDICATIVE*, *SUBJUNCTIVE*
PROG	+, −
PERF	+, −
IND (individuability)	+, −

Table 4.2: DMRS predicate features (left) and their corresponding values (right).

4.4.2 Masked Node Modeling

As an encoder transformer, GFoLDS is limited in terms of pretraining objectives to those tasks which involve de-noising a corrupted input sequence. Therefore, as mentioned at the beginning of this chapter, the pretraining objective for this model is *masked-node modeling* (MNM), which is analogous to the masked language modeling (MLM) objective used to pretrain encoder transformer LMs such as BERT (Devlin et al., 2019); although some encoder transformers such as ELECTRA (Clark et al., 2021) are pretrained with more complex corruption procedures reminiscent of generative adversarial networks (Goodfellow et al., 2014), I chose to employ the simpler masking method and leave exploration of more advanced approaches to future work.

The MLM procedure as introduced in Devlin et al. (2019) model proceeds as follows: given an input sequence S comprised of tokens $s_1 \dots s_n$, ∼15% of the tokens are randomly chosen as *targets*—the remaining ∼85% are ignored by the loss (i.e. error) function. Of the targets, ∼80% are chosen (randomly) to be *masked*—i.e. replaced by a special [MASK] token—∼10% are randomly chosen to be replaced by another randomly chosen token, while the remaining ∼10% of the targets are left unperturbed. The model's pre-training objective is to use the surrounding context to de-corrupt the target tokens—i.e. predict the correct (original) token. Under this paradigm, loss is computed in terms of mean cross-entropy (Mao, Mohri, and Zhong, 2023) between the model's predicted categorical distribution over the vocabulary and the distribution with all probability mass concentrated on the target word, for each target word in the input sequence.

For superficial models such as BERT, the goal of the MLM pretraining objective is for the model to learn textual co-occurrence relations between tokens: by masking and corrupting random tokens, and training the model to denoise the input, the model learns to predict the context-dependent presence/absence of a given token based on the presence/absence—and position—of other tokens in the context(s) in which that token appears. Similarly, the goal of the masked-node modeling objective for the GFoLDS model is to learn *logical* co-occurrence relations between predicate symbols in DMRS-derived graph representations: for example, the model may learn that the verb *see* tends to occur in the past tense with a third-person plural second-place argument.

While BERT is pretrained using next sentence prediction (NSP; see Devlin et al., 2019) in conjunction with MLM, Liu et al.'s (2019) experimental results suggest that NSP does not improve—and in some cases even *harms*—downstream performance. For this reason, subsequent masked LMs such as RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2021), and DeBERTa (He et al., 2021) omit the NSP objective in their pretraining procedures. Therefore, I also chose not to include a secondary pretraining objective for the GFoLDS model; I discuss the possibility of utilizing a task similar to NSP in conjunction with MNM

in order to pretrain GFoLDS to understand *entailment relations*—rather than sentence-level subsequence as with NSP—in Chapter 7, but leave the implementation of such a procedure to future work.

4.4.3 Hyperparameters

During BERT's pretraining procedure, while 15% of the input tokens are *selected* for prediction, only 80% of the selected tokens are *masked* (as discussed in Section 4.4.2): 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. However, for GFoLDS, the [MASK] token *does* occur during fine-tuning as well, due to OOV items (as discussed in Section 4.3.1). 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, I chose to mask 100% of the selected tokens during pre-training, with the slightly higher selection probability of 20%.

For pretraining (and the subsequent experiments described in Chapters 5-6), I employed a GFoLDS model with two SWA layers and ten encoder layers with eight attention heads each, and set $d_{SWA} = d_{model} = 1024$. The MNM prediction head that I 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.

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.

I pretrained GFoLDS with a batch size of 16 for four epochs: although BERT is pretrained for ∼40 epochs, Muennighoff et al. (2024) find that while re-using data for up to four epochs results in negligibly decreased performance compared to pretraining on the same amount of

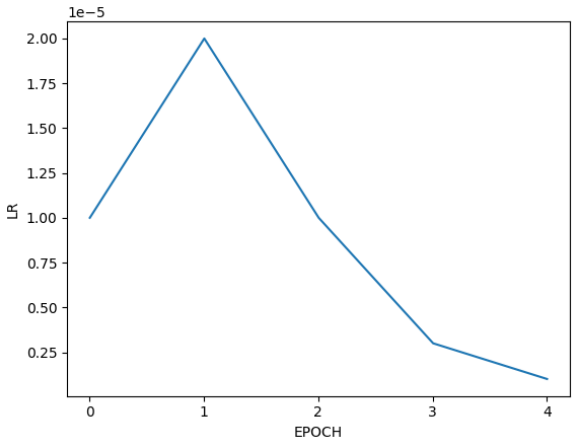


Figure 4.12: GFoLDS pretraining learn rate schedule.

unique data, LMs experience rapidly dimishing returns thereafter. To update the model's parameters, I employed the AdamW optimizer (Loshchilov and Hutter, 2017) with a weight decay value of 10^{-5}. I split the pre-training dataset into 200 uniformly-sized folds: the data was randomized at each epoch by first shuffling the order that the folds were loaded, then shuffling the order of the examples within each fold (rather than shuffling the entire dataset, which was intractable due to hardware constraints). I 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 4.12). 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 (the learn rate was updated at the end of each fold—i.e. 200 times per epoch).

Although many recent models (e.g. Touvron et al., 2023; Dubey et al., 2024, etc.) utilize more sophisticated learn rate schedulers (such as cosine annealing; Loshchilov and Hutter, 2016), I chose to use a linearly interpolated learn rate due to its flexibility. Note that the learn rate always updates at discreet points during training—regardless of the scheduler—due to the discrete nature of neural network training datasets. Therefore, for any learn rate scheduler $\sigma\colon [0,1] \to \mathbb{R}_+$—expressed here as a function mapping the proportion of the training procedure that has been completed thus far to a learn rate—we can set the value of a linearly

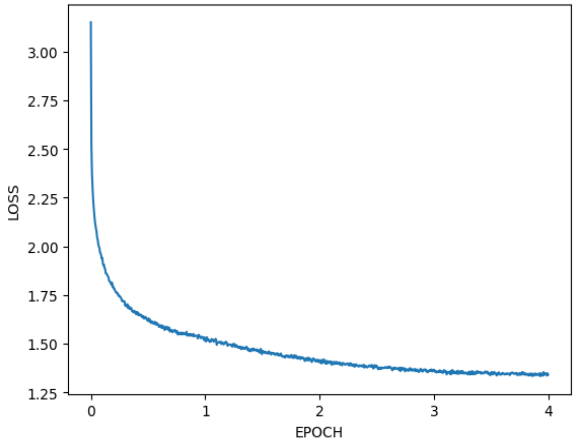


Figure 4.13: GFoLDS pretraining cross-entropy loss.

interpolated learn rate scheduler to $\sigma(u)$ at each update point $u \in [0, 1]$: linear interpolation can approximate *any* scheduler.

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 4.13).

4.5 Discussion

In this chapter, I introduced the GFoLDS model: a pretrained, transformer encoder model over logical forms that is intended to address the deficiencies of the FoLDS model of Chapter 3. In the following chapter (Chapter 5), I compare this pretrained GFoLDS model to competing approaches on a range of logical-reasoning-oriented tasks. As discussed in Chapter 1, the primary objective of this dissertation is to demonstrate the feasibility and utility of language modeling over logical forms: as such, in Chapter 6, I analyze the GFoLDS model discussed in the present chapter, with a particular focus on its propensity to scale to larger amounts of pretraining data.

Chapter 5

GFoLDS: Experiments

In Chapter 4, I introduced the GFoLDS model, an implementation of the graph transformer paradigm (Wu, Peng, and Smith, 2021) over DMRS-derived graph representations of logical forms. In this chapter, I evaluate GFoLDS on a variety of logical-reasoning-oriented tasks, with the goal of comparing GFoLDS to a comparably-sized superficial LM (BERT; Devlin et al., 2019), in order to provide evidence in support of the Accelerated Learning Hypothesis of Chapter 1. In particular, I aim to demonstrate that language models over logical forms are able to learn from less data than their superficial counterparts.

To that end, I first pretrain $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ *comparison* models (Section 5.1), which are architecturally identical to the original BERT models introduced in Devlin et al. (2019), but pretrained on the same amount of data as GFoLDS. While I also evaluate GFoLDS against the original BERT models, the comparison models allow for a more direct evaluation of the two architectures.

In Section 5.2, I then evaluate GFoLDS and the four BERT models on the RELPRON (Rimell et al., 2016) dataset, which evaluates the models' ability to compose relative clauses in a distributional setting. As the FDS (Emerson, 2018) and FDSAS (Lo et al., 2023) models discussed in Chapter 4 were also evaluated on RELPRON, this dataset additionally provides the opportunity to compare GFoLDS to these two distributional models over logical forms—its

most directly-comparable counterparts.

In the next experiment (Section 5.3), I fine-tune GFoLDS and the BERT models on the Stanford NLI (SNLI; Bowman et al., 2015) dataset, which demonstrates GFoLDS' applicability to large-scale NLP benchmarks containing multi-sentential data. I then turn to a binary sentence-classification task: the MegaVeridicality V2.1 (White et al., 2018) factuality dataset. Finally, I evaluate GFoLDS and BERT on the property inference task derived from the McRae et al. (2005) dataset discussed in Chapter 3, which allows for a direct comparison between GFoLDS, BERT, and the FoLDS model introduced in Chapter 3.

By evaluating GFoLDS on such a wide range of tasks, I intend to demonstrate the versatility and practical viability of this model—and, therefore, of language modeling over logical forms in general.

5.1 BERT Comparison Models

As discussed above, in the experiments in Sections 5.2-5.5 of this chapter, I compare GFoLDS to the $\text{BERT}_{\text{base}}$- and $\text{BERT}_{\text{large}}$-uncased models of Devlin et al. (2019), both of which are pretrained on roughly 6.5 times more data than the GFoLDS model introduced in Chapter 4. Furthermore, both BERT models were pretrained for roughly 40 epochs, whereas I used only four pretraining epochs for the GFoLDS model: in total, the BERT models received $\sim$65 times more pretraining than GFoLDS.

While the Accelerated Learning Hypothesis (ALH) posits that language models over logical forms can learn useful representations with less data than their superficial counterparts (as discussed in Chapter 1), the gulf between the respective amounts of pretraining afforded to GFoLDS and BERT may be too large to provide a fair comparison of these models with respect to the ALH. Therefore, in order to obtain a more accurate baseline for comparison, I additionally pretrained $\text{BERT}_{\text{base}}$- and $\text{BERT}_{\text{large}}$-uncased models from scratch on the same dataset as GFoLDS (the surface sentences, of course, rather than the DMRS graphs), for the

same number of epochs.

The purpose of these BERT comparison models is to demonstrate that GFoLDS is able to outperform a superficial transformer when pretrained on the same amount of data. If this is in fact the case, it then follows that GFoLDS would be able to perform as well as BERT with *less* training data, thereby providing strong experimental evidence in support of the ALH.

5.1.1 Pretraining Data

In the context of the current discussion, there is an important distinction to be made between *base* training data and *actual* training data. Recall that, in Chapter 4, I randomly selected ∼17.5 million sentences from English Wikipedia to serve as the GFoLDS model's pretraining corpus: I will refer to this set of sentences as the base dataset. However, as discussed in Chapter 4, the ACE/ERG (Copestake and Flickinger, 2000) MRS parser employed over the base dataset was only able to parse ∼84% of those sentences, and the ∼16% of the sentences that remained unparsed were discarded: I will refer to the ∼84% of the base dataset (∼14.6 million sentences) that was successfully parsed as the actual dataset.

On the other hand, for a superficial LM such as BERT that employs a WordPiece (or similar) tokenizer (Wu et al., 2016), the actual dataset is identical to the base dataset (in the absence of data deduplication or similar preprocessing techniques): these models do not use a rule-based (or any) parser, and so do not suffer from the resulting loss of data due to unparsable sentences.

From a purely theoretical perspective, the most fair comparison between BERT and GFoLDS would be obtained by pretraining the BERT models on GFoLDS' actual dataset. This would demonstrate that language models over logical forms are able to outperform comparable superficial models, under the assumption that we have a rule-based parser that is able to parse any sentence, while leaving the actual development of such a parser to future work.

The objectives of this dissertation, however, are not purely theoretical. While I do

admittedly leave some pieces of this puzzle to future work (such as the CARG/OOV items discussed in Chapter 4), the overall goal of this work is to demonstrate the validity of the Accelerated Learning Hypothesis of Chapter 1—and the general viability of language modeling over logical forms—given the tools that are *currently available*. If (solely for the sake of example) GFoLDS were able to match the performance of BERT with ∼84% of the pretraining data, then—given the ACE/ERG parser's ∼84% successful parse rate—this model would provide no practical benefit over its superficial counterparts.

I therefore chose to pretrain the BERT comparison models on GFoLDS' base—rather than actual—dataset. This means that, in reality, the BERT comparison models are pretrained with ∼1.19 times more actual data than GFoLDS.

5.1.2 Hyperparameter Selection

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. I 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 $\text{BERT}_{\text{large}}$—which is over three times larger than $\text{BERT}_{\text{base}}$—I performed the majority of the trials with $\text{BERT}_{\text{base}}$, then transferred the best-performing configuration found during these experiments to $\text{BERT}_{\text{large}}$.

I 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 discussed in Chapter 4 (and Devlin et al., 2019). Due to hardware constraints, I was limited to using a batch size of 16 for all of the BERT pretraining trials.

The first configuration (a) was identical to that which I employed for GFoLDS in Chapter 4: a weight decay value of 10^{-5}; a linearly interpolated learn rate between values of 10^{-5} at the beginning of the first epoch, 2×10^{-5} at the end of the first epoch, 10^{-5} at the end of

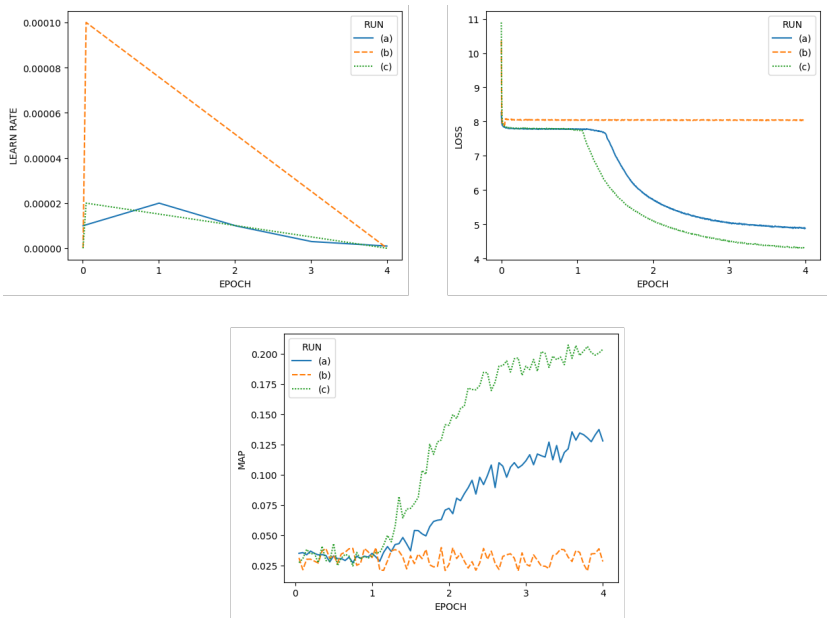


Figure 5.1: Learning rates (top left), cross-entropy loss values (top right), and MAP scores on the RELPRON development split (bottom) across training steps for the $\text{BERT}_{\text{base}}$ pretraining trials (a)-(c).

the second, 3×10^{-6} at the end of the third, and 10^{-6} at the end of the fourth; and a token selection probability of 20% with a masking probability of 100%.

In the second configuration (b), I 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 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 I was not able to increase the batch size due to hardware constraints (as mentioned above), Granziol, Zohren, and Roberts (2022) show that learn rate scaling proportional to batch size can be used to control for the effect of batch size on training loss. I therefore introduced a third trial (c): this configuration was identical to that of (b) with the exception of the peak learn rate value, which I scaled down to 2×10^{-5} to account for the difference in batch size.

I then evaluated trials (a)-(c) with respect to loss and a validation task that does not

require fine-tuning: the development split of the RELPRON (Rimell et al., 2016) dataset (discussed further in Section 5.2). The results of these experiments are shown in Figure 5.1.

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) outperformed (a) both in terms of minimum cross-entropy (4.2988 vs. 4.8716) and peak MAP score (0.207 vs. 0.137).

As discussed in Chapter 4, Liu et al. (2019) suggest that pretraining with the secondary NSP objective does not improve—and in some cases may even hinder—model performance. I 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 5.2.

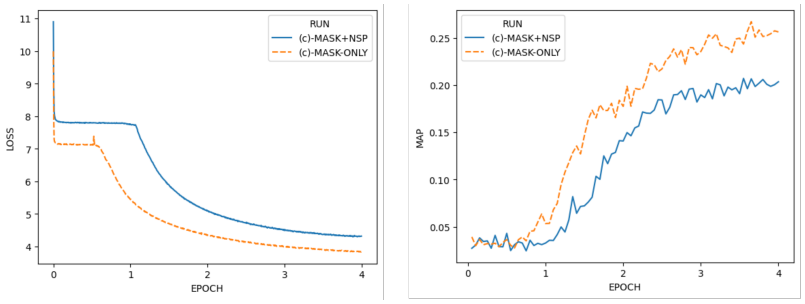


Figure 5.2: Cross-entropy loss values (left), and MAP scores on the RELPRON development split (right) across training steps for the $\text{BERT}_{\text{base}}$ pretraining configuration (c) with and without the secondary NSP objective.

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 5.2)—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, I 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 the remainder of this chapter.

I then pretrained $\text{BERT}_{\text{large}}$ with (non-NSP) hyperparameter configuration (c). As shown in Figure 5.3 (configuration (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, Lawrence, and Giles, 2000; Salman and Liu, 2019), I scaled the peak learn rate by a factor of 1/10 in trial (b) to account for the difference in size between the $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ models.

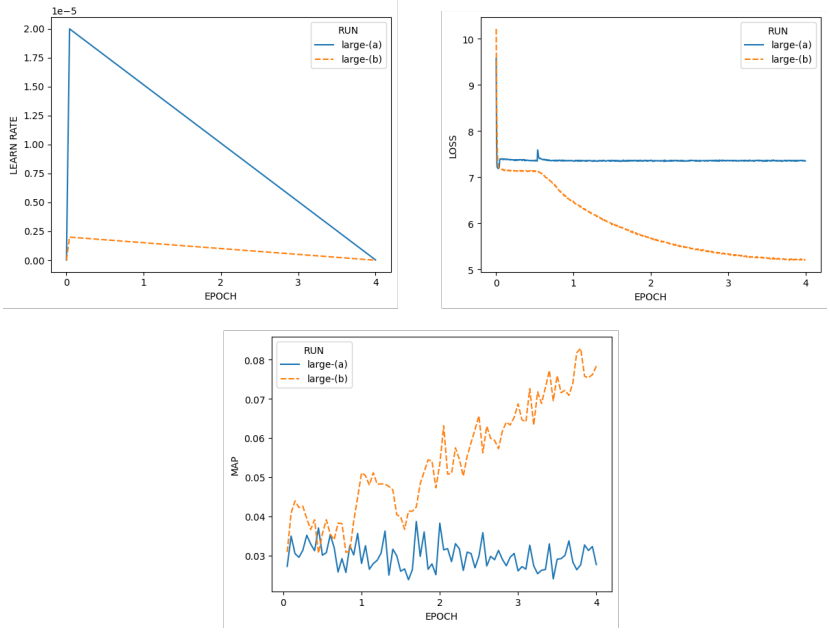


Figure 5.3: Learning rates (top left), cross-entropy loss values (top right), and MAP scores on the RELPRON development split (bottom) across training steps for the $\text{BERT}_{\text{large}}$ pretraining trials (a)-(b).

Trial (b) vastly outperformed (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; 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—I 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 comparison, 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).

5.2 RELPRON

I then evaluated GFoLDS and the comparison and original BERT models on the RELPRON (Rimell et al., 2016) dataset. I additionally compare these models' performance to that of FDS (Functional Distributional Semantics; Emerson, 2018) and its successor, FDSAS (Functional Distributional Semantics at Scale; Lo et al., 2023)—discussed in Chapters 3 and 4, respectively—which, as distributional models over DMRS graphs, are both conceptually (if not architecturally) closely related to GFoLDS.

The objective of this section is to compare the ability of GFoLDS to comprehend subset relations between the denotations of concepts at a compositional (as opposed to simply lexical) level to that of these competing approaches.

5.2.1 Task Description

This dataset consists of *terms* (nouns), each paired with a hypernym and up to ten unique *properties*: relative clauses that restrict that hypernym. For example, the term *telescope* is paired with the hypernym *device*, and has the property *astronomers use* (among others—see Table 5.1). These entries are divided into development and test splits: 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 properties per term).

The task is then to retrieve the properties that pertain to a given term, without including those that do not. Precisely, for each term t, the objective is to rank all of the properties in (the split in question of) the dataset, such that all of the *relevant* properties of t—i.e. those that pertain to t—are ranked above the *irrelevant* properties (those that do not).

As this task is effectively one of information retrieval, the evaluation metric is Mean Average Precision (MAP; Zhu, 2004), a common measure of performance in information retrieval tasks. Given a set of queries (terms) T, the MAP score for T with respect to a given information retrieval system is the mean of the average precision scores $AP(t) \in (0, 1]$ for each $t \in T$. Given a term t, a list of N "documents" (properties) $\pi^{(t)}_{(-)}$ ordered/ranked by the information retrieval system with respect to t, and a set of relevant properties $R(t)$ for t, the average precision for t is given in Equation 5.1 below.

$$AP(t) = \sum_{i=1}^{N} \frac{P_i(\pi^{(t)}_{(-)}) \cdot \mathbb{1}(\pi^{(t)}_i \in R(t))}{|R(t)|} \tag{5.1}$$

Where $P_i(\pi^{(t)}_{(-)}) = \sum_{k=1}^{i} \mathbb{1}(\pi^{(t)}_k \in R(t)) \,/\, i$, and $\mathbb{1}(\pi^{(t)}_i \in R(t)) = 1$ if $\pi^{(t)}_i \in R(t)$, and 0 otherwise. For example, suppose that $N = 6$ and $R(t) = \{\pi^{(t)}_1, \pi^{(t)}_2, \pi^{(t)}_4, \pi^{(t)}_6\}$: the first-, second-, fourth-, and sixth-highest-ranked terms are relevant, while the third- and fifth-highest-ranked are not. Then the numerators of Equation 5.1 are calculated as in Equation 5.2.

$$\begin{aligned} &P_1(\pi^{(t)}_1) \cdot 1 \;+\; P_2(\pi^{(t)}_2) \cdot 1 \;+\; P_3(\pi^{(t)}_3) \cdot 0 \;+\; P_4(\pi^{(t)}_4) \cdot 1 + \\ &P_5(\pi^{(t)}_5) \cdot 0 \;+\; P_6(\pi^{(t)}_6) \cdot 1 \\ &= 1/1 \;+\; 2/2 \;+\; 0 \;+\; 3/4 \;+\; 0 \;+\; 4/6 \\ &= 3.42 \end{aligned} \tag{5.2}$$

Which is then divided by $|R(t)| = 4$ to yield $AP(t) = 0.855$; the average precision score rewards ranking relevant properties over irrelevant ones, and penalizes ranking irrelevant

properties over relevant ones.

Note that, in the RELPRON dataset, properties may be either *subject*-type—in which the corresponding hypernym is the subject of the relative clause—or *object*-type (where the hypernym is the object of the relative clause): for example, *astronomers use* and *detects planets* are object- and subject-type properties of *telescope*, respectively (see Table 5.1).

The dataset additionally includes *confounder* properties, which deliberately contain other terms in the dataset. For example, the property *astronomers use* of the term *telescope* is a confounder, because it contains the word *astronomer*, which is another term in the RELPRON dataset: these confounder properties are designed to fool models that rely on shallow heuristics such as lexical overlap.

To evaluate the GFoLDS and BERT models on RELPRON, I 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*" (see Table 5.1). I then replace the target term—in the above example, *telescope*—with the [MASK] token. The probability that the model assigns to the masked token being a given term is then taken as proportional to the probability that the property applies to that term.

For example, given the property/template p_i = "*A device that astronomers use is a* [MASK]", I extract—from the distribution over the [MASK] token—the probability $p_i^{(t)}$ that the model assigns to each term t in the RELPRON dataset. For each term t, I then order the list of properties $\pi_{(-)}^{(t)}$ with respect to t (see Equation 5.1) according to $p_{(-)}^{(t)}$: $\pi_1^{(t)} = argmax_k(p_k^{(t)})$ is the property/template k in which t is assigned the highest probability out of all of the properties in the dataset, $\pi_2^{(t)} = argmax_{k:k\neq p_1^{(t)}}(p_k^{(t)})$ is the is the property/template k in which t is assigned the *second*-highest probability, and so on.

Due to the small size of the dataset and lack of a training split, there is no fine-tuning involved in this task: the frozen, pre-trained models were used to obtain the term probabilities $p_i^{(t)}$.

Term/Hypernym	Properties	Corresponding Templates
telescope/ device	*astronomers use*	"A device that astronomers use is a telescope"
	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 5.1: The 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.

5.2.2 Results

Model	All	No-UNK/NE
GFoLDS	—	0.651
$\text{BERT}_{\text{large}}$	**0.768**	**0.769**
$\text{BERT}_{\text{base}}$	0.667	0.690
$\text{BERT-C}_{\text{large}}$	0.047	0.056
$\text{BERT-C}_{\text{base}}$	0.174	0.193
FDS	0.160	—
FDSAS	0.580	—

Table 5.2: MAP scores on the RELPRON test split, where $\text{BERT-C}_{\text{base}}$/$\text{BERT-C}_{\text{large}}$ denote the comparison models discussed in Section 5.1. Note that I did not directly evaluate FDS and FDSAS, but rather record the scores reported in Emerson (2018) and Lo et al. (2023), respectively.

To evaluate the GFoLDS model, I discarded all examples that contained CARG-bearing predicates[1] (i.e. named entities) or OOV items: this subset of the dataset corresponds to the "No-UNK/NE" column of Table 5.2. As discussed in Chapter 4, CARGs and OOV items are effectively ignored by the GFoLDS model: given that each template only contains four content words—namely, 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 CARG-bearing predicates. If the target term itself is an OOV item or CARG-bearing predicate, then the task is impossible: in this case, the term is not in the vocabulary, and so it can never be predicted by the model.

After discarding all examples containing CARGs or OOV items, the resulting No-UNK/NE subset of the RELPRON test split contains 63 terms (∼86% of the original 73) and 421 properties (∼74% of the original 569), for a total of ∼6.68 properties per term on average.

On the No-UNK/NE subset, GFoLDS achieves a MAP score of 0.651, placing the model behind $\text{BERT}_{\text{base}}$ (0.690) and $\text{BERT}_{\text{large}}$ (0.769), as shown in Table 5.2. Although GFoLDS did not surpass the performance of either BERT model, it is reasonably competitive with $\text{BERT}_{\text{base}}$, a remarkable result given that GFoLDS was pretrained on ∼6.5 times less data.

[1]While none of the RELPRON examples *actually* contain named entities, the ACE/ERG parser occasionally misinterprets common nouns as named entities.

On the other hand, the two BERT comparison models—which were pretrained on the same amount of data as GFoLDS—both lag far behind GFoLDS, with MAP scores of 0.193 and 0.056 for BERT-C_{base} and BERT-C_{large}, respectively. It may initially seem counterintuitive that BERT-C_{large} obtained a lower MAP score than BERT-C_{base} despite the fact that the original $BERT_{large}$ model *outperformed* $BERT_{base}$. However, note that BERT-C_{base} also outperformed BERT-C_{large} on the RELPRON *development* split in Section 5.1.2: as discussed in that section, such results are to be expected due to the difference in size between the base and large BERT architectures, and the limited amount of pretraining data that these comparison models were exposed to.

Unfortunately, the pre-trained FDS and FDSAS models are not publicly available, so I was unable evaluate them on the No-UNK/NE subset of the dataset in order to obtain a direct comparison to GFoLDS. However, based on the differences in MAP scores from the complete RELPRON test set 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; see Table 5.2), it seems quite reasonable to assume that GFoLDS (0.651) would outperform FDS (0.160) and FDSAS (0.580) on that data.

Model outputs for GFoLDS and the four BERT models on this task are available on GitHub[2].

5.3 Natural Language Inference (SNLI)

Next, I compared GFoLDS to the BERT models with respect to a natural language inference (NLI) task; there are two salient reasons that I elected to evaluate the model on this benchmark. First, (as discussed in Chapter 2), NLI tasks require logical reasoning capabilities that extend beyond basic linguistic competence (including real-world knowledge; Richardson et al., 2020), and therefore provide an excellent setting in which to assess the soundness of the Accelerated Learning Hypothesis of Chapter 1. Additionally, NLI is a large-scale NLP benchmark with

[2]https://github.com/mjs227/GFoLDS/tree/main/RELPRON_model_outputs

practical real-world applications, and GFoLDS' successful performance of an NLI task would therefore represent a substantial step towards achieving one of the primary objectives of this dissertation: namely, demonstrating the general viability of language modeling over logical forms.

In this experiment, I specifically chose to evaluate GFoLDS (and the BERT models) on the Stanford NLI (SNLI; Bowman et al., 2015) dataset over other popular NLI datasets such as MultiNLI (MNLI; Williams, Nangia, and Bowman, 2017) and Adversarial NLI (ANLI; Nie et al., 2020). The MNLI dataset is problematic for the GFoLDS model in particular, as one of the sources from which its examples are derived is the Linguistic Data Consortium's Switchboard[3] corpus, which consists of transcribed telephone conversations. Transcribed speech data contains many incomplete sentences and fragments, which are exceedingly difficult to successfully parse for the ACE/ERG rule-based parser that I employ to generate DMRS graphs for GFoLDS.

ANLI, on the other hand, was designed to be exceptionally difficult: the current best-performing model on that dataset, T5 (Raffel et al., 2020), has three billion parameters ($\sim$10 times larger than $\text{BERT}_{\text{large}}$; $\sim$20 times larger than GFoLDS) and only achieves 81% accuracy—for comparison, $\text{BERT}_{\text{large}}$ achieves 91.1% accuracy on SNLI. The best-performing BERT variant (InfoBERT; Wang et al., 2021) attains a mere 75% accuracy on ANLI. The extreme difficulty of this dataset would therefore hinder any attempt to discern differences between the performances of the models that I evaluate in this experiment.

5.3.1 Task Description

Although I provide a description of NLI tasks in Chapter 2, I do so again here for the sake of convenience. An NLI dataset such as SNLI consists of (premise, hypothesis, label) triples (P_i, H_i, L_i). The label $L_i = \textit{entailment}$ if the truth of the premise necessitates the truth of the hypothesis; $L_i = \textit{contradiction}$ if the truth of the premise necessitates the falsity of the

[3]https://catalog.ldc.upenn.edu/LDC97S62

Premise	**Hypothesis**	**Label**
It is raining	The ground is wet	*Entailment*
The man is lying down	The man is standing	*Contradiction*
It is December	It is five o'clock	*Neutral*

Table 5.3: Example (premise, hypothesis) pairs for each of the three NLI labels.

hypothesis; and $L_i = \textit{neutral}$ otherwise. Examples of all three classes are given in Table 5.3.

The SNLI dataset consists of 550,152 training examples and 10,000 test examples[4]—both sets contain a roughly balanced proportion of examples from each of the three classes. The ACE/ERG parser was able to successfully parse ∼88% of these (premise, hypothesis) pairs, for a total of 8,813 DMRS graph pairs in the test split (3,053 entailment, 2,887 contradiction, 2,873 neutral) and 487,590 in the training split (163,670 entailment, 162,476 contradiction, 161,444 neutral).

5.3.1.1 Constructing Graph Representations

The GFoLDS architecture described in Chapter 4 cannot process multiple sentences at once: the largest linguistic unit that it is capable of taking as input is a single sentence. While this obviously represents a serious general limitation of the model (and I suggest a potential avenue to overcoming this impediment in Chapter 7), it also presents a more immediate complication with respect to the SNLI dataset, whose examples are given as (premise, hypothesis, label) triples. Given a premise P_i with the corresponding hypothesis H_i, it is technically possible to simply represent the pair (P_i, H_i) as the disjoint union $G(P_i) \oplus G(H_i)$ of their respective DMRS-derived graphs: no aspect of the GFoLDS architecture inherently assumes that input graphs are connected.

However, the disjoint union $G \oplus G'$ of any two DMRS-derived graphs does not encode any information about the linear order between G and G'. It is therefore impossible for the GFoLDS model to know which graph denotes the premise, and which denotes the hypothesis.

[4]Along with a further 10,000 examples in the development set, which I do not use in this experiment.

While this is not problematic for identifying examples of contradiction[5], order is necessary to distinguish between the entailment and neutral labels: for example, (*it is raining*, *the ground is wet*) is an example of entailment, while (*the ground is wet*, *it is raining*) is neutral.

This is to say that we need a representation that combines $G(P_i)$ and $G(H_i)$ into a single graph $G(P_i, H_i)$ that encodes the order between the premise and hypothesis (in a way that GFoLDS is able to interpret), while also preserving all of the information of $G(P_i)$ and $G(H_i)$. This is achieved using the DMRS *if_x_then* token, which corresponds (perhaps unsurprisingly) to the word "if".

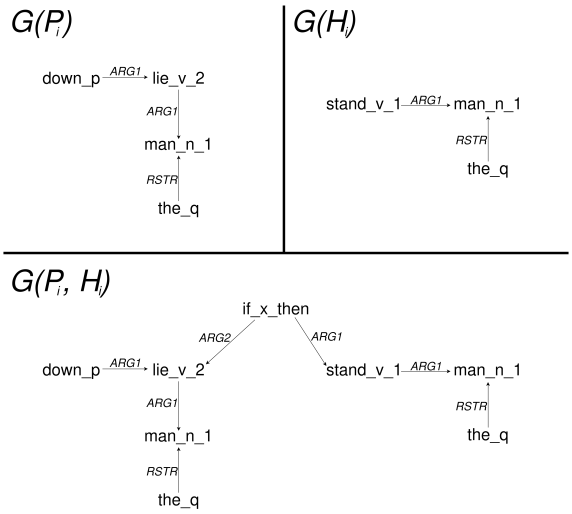


Figure 5.4: Illustration of the derivation of $G(P_i, H_i)$ (bottom) from $G(P_i)$ (top left) and $G(H_i)$ (top right) for the contradiction example (*the man is lying down, the man is standing*). At first glance, it may seem unusual that the premise is connected to *if_x_then* by the *ARG2* edge, while the hypothesis is connected by *ARG1*, especially if one verbalizes $G(P_i, H_i)$ as "*if the man is lying down, then the man is standing*". The intended verbalization, however, is "*the man is standing, if the man is lying down*": here, the numerical edge labels correctly correspond to the surface order of the conjuncts.

To be precise: $G(P_i, H_i)$ is constructed from $G(P_i) \oplus G(H_i)$ by adding the node *if_x_then*, and then inserting edges *if_x_then* $\xrightarrow{ARG1}$ $htop(G(H_i))$ and *if_x_then* $\xrightarrow{ARG2}$ $htop(G(P_i))$, as illustrated in Figure 5.4. Given an MRS structure M and its DMRS-derived graph $G(M)$, $htop(G(M)) = p$ is the node corresponding to the predicate p that outscopes[6] all other

[5] $(P \rightarrow \neg H) \leftrightarrow (\neg(P \wedge \neg\neg H)) \leftrightarrow (\neg(P \wedge H)) \leftrightarrow (\neg(H \wedge \neg\neg P)) \leftrightarrow (H \rightarrow \neg P)$

[6] *Outscopes* is a technical term in (D)MRS, and does not necessarily refer to e.g. quantifier or negation scope.

predicates p' in M. I defer interested readers to Copestake et al. (2005) for an in-depth description of the *outscopes* relation in (D)MRS—for the purposes of the discussion at hand, it suffices to state that $htop(G(M))$ typically corresponds to the main verb of the sentence from which M is derived.

Note that this is in fact how the *if_ x_ then* token functions in DMRS: given a sentence $S =$ "*if* S_1, *then* S_2", the DMRS representation $D(S)$ is (roughly[7]) equivalent to the graph formed from $D(S_1) \oplus D(S_2)$ by adding an *if_ x_ then* node, and inserting edges *if_ x_ then* $\xrightarrow{ARG2}$ $htop(D(S_1))$ and *if_ x_ then* $\xrightarrow{ARG1}$ $htop(D(S_2))$. Critically, this means that the structure of $G(P_i, H_i)$ is familiar to the pretrained model. This representational format therefore satisfies the desiderata laid out above: namely, $G(P_i, H_i)$ preserves the information of $G(P_i)$ and $G(H_i)$, while encoding the order between the premise and hypothesis in a manner that is interpretable to the GFoLDS model.

5.3.1.2 CARGs and OOV Items

There is of course the danger that the presence of out-of-vocabulary (OOV) items and CARG-bearing predicates in the data could negatively effect GFoLDS' performance: as discussed in Chapter 4, I removed (masked) all CARGs and OOV items from the model's DMRS input graphs during pretraining. For example, consider the premise "*the dog walked towards the door*" and the hypothesis "*the dog sat towards the door*"—a clear example of contradiction. Now suppose (for the sake of example) that *walk* and *sit* are not in GFoLDS' vocabulary. In this case, the model would only see "*the dog* [MASK] *towards the door*" and "*the dog* [MASK] *towards the door*"—it would be impossible to make an informed prediction.

In order to evaluate the effect that CARGs and OOV items may have on GFoLDS' performance on the SNLI dataset, I additionally constructed No-UNK/NE subsets (as in Section 5.2) of the SNLI train and test splits in which all examples containing OOV items or CARG-bearing predicates—in either the premise or hypothesis—were omitted. This removed

[7]Note that $htop(D(S)) =$ *if_ x_ then*. This is, however, irrelevant to the GFoLDS model, whose input graphs do not include any information about *htop* nodes.

roughly half of the examples from each split, resulting in a total of 4,496 test (1,607 entailment, 1,452 contradiction, 1,437 neutral) and 254,926 training examples (90,211 entailment, 83,027 contradiction, 81688 neutral). I evaluated all five models (GFoLDS and the four BERT models) on both the full dataset and the No-UNK/NE subset.

5.3.2 Results

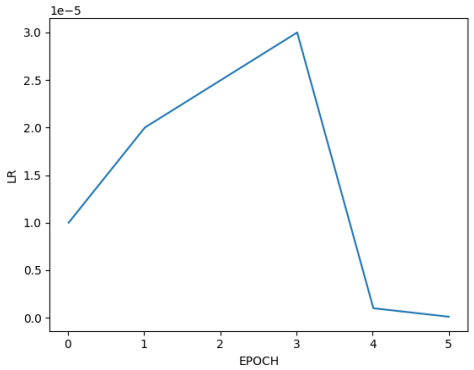


Figure 5.5: SNLI fine-tuning learn rate schedule for GFoLDS and the $\text{BERT}_{\text{base}}$ models.

On both datasets (the full SNLI dataset and the No-UNK/NE subset), I 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 5.5). I fine-tuned $\text{BERT}_{\text{large}}$ with identical hyperparameters, except all learn rates mentioned above were multiplied by 1/10 for this model, as $\text{BERT}_{\text{large}}$ 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 in Section 5.1.2. However, the accuracy that the original $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ models attained with the hyperparameter configurations described in the above paragraph (see Table 5.4) matched that reported for those models on the SNLI dataset in Zhang et al. (2020c). It is therefore reasonably safe to

Model	All	No-UNK/NE
GFoLDS	81.0%	80.8%
$\text{BERT}_{\text{large}}$	**91.1%**	**90.7%**
$\text{BERT}_{\text{base}}$	90.7%	90.0%
$\text{BERT-C}_{\text{large}}$	62.0%	60.1%
$\text{BERT-C}_{\text{base}}$	79.9%	78.3%

Table 5.4: Model accuracy on the original SNLI test set (**All**) and the **No-UNK/NE** subset. The best results for each experiment (column) are highlighted in bold.

assume that this set of hyperparameters is (near-)optimal for the BERT models with respect to this data.

The results of these experiments are shown Table 5.4. $\text{BERT}_{\text{large}}$ attains the highest accuracy out of all five models on both the original dataset and the No-UNK/NE subset (91.1% and 90.7%, respectively). GFoLDS trails behind the original BERT models on both datasets, with 81.0% accuracy on the original dataset and 80.8% on the No-UNK/NE subset. However, GFoLDS outperforms the BERT comparison models of Section 5.1—$\text{BERT-C}_{\text{base}}$ and $\text{BERT-C}_{\text{large}}$—which achieve respective accuracies of 79.9% and 62.0% on the original datset, and 78.3% and 60.1% on the No-UNK/NE subset. As in Section 5.2.2, $\text{BERT-C}_{\text{base}}$ outperformed $\text{BERT-C}_{\text{large}}$, despite the fact that $\text{BERT}_{\text{large}}$ outperformed $\text{BERT}_{\text{base}}$. Again, this is to be expected: $\text{BERT-C}_{\text{base}}$ converged to a significantly lower cross-entropy loss during pretraining than $\text{BERT-C}_{\text{large}}$ (see Section 5.1.2), due to the small size of the pretraining corpus and the relative difference in parameter count between the two models.

Note that the original BERT models were pretrained with the additional Next Sentence Prediction (NSP; see Devlin et al., 2019) objective, which is intended to teach the model to detect relationships between pairs of sentences. It is possible that the BERT comparison models' performance on this task was negatively impacted by the fact that they were *not* pretrained with the NSP objective (as discussed in Section 5.1.2). However, RoBERTa (Liu et al., 2019)—which is architecturally identical to BERT, but was not pretrained with NSP—matches the performance of the original BERT on SNLI (Sun et al., 2020), suggesting that pretraining with the NSP objective does not significantly improve performance on this

dataset.

Table 5.4 shows that none of the models experienced a substantial decrease[8] in accuracy on the No-UNK/NE subset in comparison to the original dataset. Critically, this suggests that the GFoLDS model is fairly robust to the masked OOV items and CARGs (see the discussion in Chapter 4) present in the original dataset. While the inability to incorporate these items into the DMRS-derived input graphs still represents a serious limitation to the model—such as in the RELPRON experiment (Section 5.2), where entire examples containing CARGs and OOV items had to be removed—these results suggest that this shortcoming may not present a significant barrier with respect to NLI tasks.

Overall, these results demonstrate that GFoLDS is able to outperform superficial models when pretrained on similar amounts of data on SNLI. As discussed above, NLI tasks such as SNLI are large-scale NLP benchmarks with practical real-world applications. These experimental results therefore represent a significant step towards accomplishing one of the main overarching objectives of this dissertation: demonstrating the viability of language modeling over logical forms.

5.4 Factuality

I then compared GFoLDS to the four BERT models on a *factuality* task. The primary objective of this section is to establish a baseline for comparison in Chapter 6, where I probe the model for weaknesses and study its propensity to scale to larger amounts of data. However, as it requires classifying entire sentences, this task also presents an opportunity to showcase one of the key advantages of GFoLDS over its predecessor—the FoLDS model introduced in Chapter 3—which is incapable of performing NLP tasks necessitating awareness of linguistic units beyond the level of the word.

[8]A (slight) decrease in accuracy is to be expected, as the No-UNK/NE subset is roughly half the size of the original dataset (as discussed in Section 5.3.1).

5.4.1 Task Description

The objective in factuality tasks is to determine whether the event denoted by a subordinate clause is presupposed to be factual, given the context of the matrix clause. To be clear, the task is not to determine the *actual* factuality of the event, but rather the speaker/author's communicative intent: whether or not they are intending to portray the event as factual. For example, in the sentence "*someone agreed that a particular thing happened*", the subordinate event (underlined) is presupposed to be factual, while in the sentence "*someone was tricked into thinking that a particular thing happened*", the event is presupposed to be non-factual.

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 5.5: 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 presupposed to be true, while a label of 0 indicates that is not.

For this experiment, I used the *MegaVeridicality V2.1* dataset (White et al., 2018), which consists of 5,026 examples, each with ten annotations of *yes*—i.e. the event is presupposed to be factual—*no*, or *maybe*. I 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): I then assigned each example a value of 1 (i.e. the subordinate event is portrayed as factual) if its mean value was greater than zero, and 0 otherwise. A value of 0 does not necessarily indicate that the subordinate event is presupposed to be *non-factual*, merely that it is not presupposed to be factual. Examples from the MegaVeridicality V2.1 dataset—along with their corresponding (binary) labels—are given in Table 5.5.

Note that there are few heuristics that models can leverage when performing this task. For example, negation is present in the matrix clauses of both positive and negative sentences:

in Table 5.5, "*a particular person didn't mean to do a particular thing*" has a label of 1, while "*someone didn't tell a particular person to do a particular thing*" has a label of 0. Furthermore, negation can—but doesn't necessarily—trigger an alternation between the positive and negative classes: "*John didn't find that a particular thing happened* " has a label of 0, while "*John found that a particular thing happened*", "*John was upset that a particular thing happened*", and "*John wasn't upset that a particular thing happened*" all have a label of 1. The models are therefore forced to learn the effect of negation on class labels for each matrix verb in the dataset.

As with the RELPRON task (Section 5.2), I removed all examples containing OOV items and/or CARG-bearing predicates from the dataset, due to the lower amount of content words per example in MegaVeridicality V2.1 in comparison to SNLI. This resulted in the removal of 1,900 examples from the dataset, leaving 3,126 remaining examples (∼62%). I withheld 20% as a test set (626 examples), which resulted in a total of 2,500 training examples.

5.4.2 Results

I trained all five models with a learn rate of 10^{-6}, a weight decay value of 10^{-5}, and a batch size of 8. I trained the models for an indefinite number of epochs, halting training once performance on the test set did not improve for five epochs (i.e. early stopping).

This is admittedly not good evaluation practice: when comparing models, early stopping should never be determined by test set performance. However, as mentioned above, this experiment is primarily intended merely to establish a baseline for further evaluation in Chapter 6, rather than to demonstrate the effectiveness of the GFoLDS model. Given this intention—and the relatively small size of the dataset—I decided to forgo the creation of a development split in order to retain sufficient data to both fine-tune and evaluate the models.

In Table 5.6, I report each model's best test set performance—i.e. five epochs before stopping training. $\text{BERT}_{\text{large}}$ reaches the highest peak accuracy (85.6%), while GFoLDS trails the two original BERT models at 81.3%. Additionally, both original BERT models take seven

Model	Accuracy	Epochs
GFoLDS	81.3%	11
$\text{BERT}_{\text{large}}$	**85.6%**	**7**
$\text{BERT}_{\text{base}}$	84.2%	7
$\text{BERT-C}_{\text{large}}$	76.2%	3
$\text{BERT-C}_{\text{base}}$	78.1%	15

Table 5.6: Results on the MegaVeridicality V2.1 binary classification task. **Epochs** denotes the number of epochs to reach peak accuracy; total training time includes an additional five epochs due to early stopping.

epochs to reach peak accuracy, while GFoLDS requires eleven.

However, GFoLDS outperforms both of the BERT comparison models of Section 5.1. With an accuracy of 78.1%, $\text{BERT-C}_{\text{base}}$ is more competitive with GFoLDS than $\text{BERT-C}_{\text{large}}$ (76.2%), but requires more epochs to converge (15 vs. 3). Again, these results suggest that—when pretrained on a similar amount of data—GFoLDS can outperform superficial models on this task.

5.5 Property Inference

In Chapter 3, I compared the simple, count-based FoLDS model trained on Simple English Wikipedia (∼85.9 times smaller than BERT's pre-training corpus) to the results obtained by Rosenfeld and Erk (2022) on the McRae et al. (2005) property inference database. In this section, I evaluate GFoLDS and the four BERT models on this task, with the primary objective of directly comparing the performance of GFoLDS to that of FoLDS. A secondary goal of this experiment is to compare the performance of FoLDS to a more advanced superficial architecture (i.e. BERT) than the LSA-vector-based model employed by Rosenfeld and Erk (2022) in their analysis.

5.5.1 Task Description

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 5.7: McRae et al. (2005) feature norms for the concept *knife* (duplicated from Table 3.1). For all other features Q, $F(knife)_Q = 0$.

Although I describe property inference tasks—and the McRae et al. (2005) database in particular—in Chapter 3, I summarize that description here for the sake of convenience.

A property inference task uses as its dataset a *feature norm database*, which consists of a set of *concepts* (words) and a set of *features*, in which each concept w is assigned a feature vector $F(w) \in \mathbb{R}^n$ (see Table 5.7), where n is the number of features in the database. The value of $F(w)_Q$ is the value of the feature Q for the word w. The McRae et al. (2005) feature norm database—which I employed in this experiment (and in Chapter 3)—consists of 541 concepts and 2,526 features; feature values are obtained from experimental participants' judgments.

Rosenfeld and Erk (2022) create ten random folds consisting of 50 concepts each from the dataset. On each fold, the concepts within the fold represent the set U of *unknown* words: words which have been observed in text but are not grounded to real-world concepts. The concepts outside of the fold represent the set K of *known* words. For each unknown word $u \in U$, the feature vector $F(u)$ is withheld (i.e. zeroed out): the task is to reconstruct $F(u)$ given the features of the known words in K and the similarity between u and each word in K. The evaluation metric used for this task is Spearman's rank correlation coefficient (Spearman ρ; Spearman, 1904), which measures the degree to which the order (i.e. ranking) of two variables is correlated.

In their experiments, Rosenfeld and Erk (2022) use the *Modified Adsorption* (ModAds;

Talukdar and Crammer, 2009) label propagation algorithm over LSA vectors generated from a PPMI-transformed co-occurence matrix obtained from a lemmatized and PoS-tagged 10.3 billion word corpus: ∼420 times larger than FoLDS' training set, ∼32 times larger than GFoLDS', and ∼5 times larger than BERT's. In Chapter 3, I employed a simple, vector-addition based approach to this task, as the complex-valued vectors generated by the FoLDS model are incompatible with ModAds. However, GFoLDS and BERT both generate real-valued vectors, and so I follow Rosenfeld and Erk (2022) and employ the ModAds algorithm over these models' embeddings in this experiment.

I refer interested readers to Talukdar and Crammer (2009) and Rosenfeld and Erk (2022) for an in-depth description of the ModAds algorithm. For the purposes of the current discussion, it suffices to note that ModAds is a label propagation algorithm over a weighted, undirected graph $G = (V, E)$ with associated feature vectors $F(x) \in \mathbb{R}^n$ for each $x \in V$, where for any two nodes $x, y \in V$, the weight of the edge $x \to y \in E$ is intended to correspond to a notion of similarity between x and y.

For each of the five models (GFoLDS and the four BERT models) M, the nodes of its corresponding ModAds graph G_M are the set of concepts/words in the McRae et al. (2005) database, while the weight of each edge $x \to y \in E$ is the cosine distance between the model's embeddings of x and y, clipped between 0 and 1.

5.5.2 Results

Hyperparameter	**Value(s)**
nn	1, 5, 10, 20
μ_{cont}	10^{-8}, 10^{-4}, 10^{-2}, 1, 10, 100, 1000
μ_{abdn}	10^{-8}, 10^{-4}, 10^{-2}, 1, 10, 100, 1000
μ_{inj}	1
β	2

Table 5.8: ModAds hyperparameters and their corresponding sets of possible values used in the grid search procedure.

In their analysis, Rosenfeld and Erk (2022) use the first fold as a development split, and

perform a grid search over the ModAds hyperparameter[9] values listed in Table 5.8. They then use the best-performing set of hyperparameters from the first fold to evaluate on the remaining nine. I replicate this procedure with GFoLDS and the four BERT models in this experiment.

Model	**Spearman ρ**
GFoLDS	0.205
FoLDS	0.253
Rosenfeld & Erk	**0.281**
$\text{BERT}_{\text{large}}$	0.241
$\text{BERT}_{\text{base}}$	0.247
$\text{BERT-C}_{\text{large}}$	0.134
$\text{BERT-C}_{\text{base}}$	0.167

Table 5.9: Property inference results on the McRae et al. (2005) database.

The results of this experiment are given in Table 5.9. Rosenfeld and Erk's (2022) approach outperforms all other models evaluated in this experiment, with a Spearman ρ of 0.281. GFoLDS ($\rho = 0.205$) outperforms the BERT comparison models—$\text{BERT-C}_{\text{base}}$ (0.167) and $\text{BERT-C}_{\text{large}}$ (0.134)—but trails behind (the original) $\text{BERT}_{\text{base}}$ (0.247) and $\text{BERT}_{\text{large}}$ (0.241).

Notably, the FoLDS model of Chapter 3—which was trained on 13.2 times less data than GFoLDS, 85.9 times less than BERT, and 420.4 times less than Rosenfeld and Erk's (2022) model—performs the second-best on this task, with a Spearman ρ of 0.253. This demonstrates that, for some applications, a carefully-constructed, specialized LM over logical forms can outperform larger, superficial models trained on significantly more data. This result constitutes a strong demonstration of the advantages of language modeling over logical forms.

While the original BERT models outperform GFoLDS by a fairly wide margin, it is important to note that Rosenfeld and Erk (2022) outperform both BERT models using LSA embeddings generated from ∼5 times more data than was used to pretrain BERT. Given the relative simplicity of LSA vectors in comparison to BERT embeddings, it stands to

[9]Again, I refer interested readers to Talukdar and Crammer (2009) and Rosenfeld and Erk (2022) for an in-depth description of these hyperparameters.

reason that for neural (and similar) models, more data does better on this task, and that the model in question has less of an impact on performance: regardless of its architecture and/or modality, a neural model must see each term in several contexts in order to learn useful embeddings. On the other hand, a model such as FoLDS, with its carefully hand-crafted procedure for model-world generation, can generate effective representations from a much more sparse training signal.

5.6 Discussion

The results of this chapter (summarized in Table 5.10) demonstrate that GFoLDS is able to compete with (near-)SoTA superficial models on a wide range of NLP benchmarks. Although the actual $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ models outperform GFoLDS, GFoLDS in turn outperforms both BERT comparison models (BERT-C_{base} and BERT-C_{large}) on all four tasks. The results of each of the four experiments conducted in this chapter (Sections 5.2-5.5) lend strong support to the Accelerated Learning Hypothesis of Chapter 1: when pretrained on the same amount of data, a language model over logical forms consistently outperforms comparably-sized superficial LMs across a broad range of NLP tasks. This indicates that language models over logical forms present a promising avenue to overcome the impending LLM training-data bottleneck discussed in Chapter 1.

To the best of my knowledge, the experiments in Sections 5.2-5.5 of this chapter represent the first time that a language model pretrained solely over logical forms has been evaluated on a wide range of downstream tasks. In particular, GFoLDS' capacity to be applied to tasks ranging from RELPRON (Section 5.2) to SNLI (Section 5.3)—while remaining competitive with a high-quality superficial model such as BERT (and vastly outperforming BERT-C)—demonstrates this model's versatility, and these results therefore represent a significant step towards accomplishing the second main objective of this dissertation: demonstrating the practical viability of language modeling over logical forms.

Model	GFoLDS	$\text{BERT}_{\text{large}}$	$\text{BERT}_{\text{base}}$	$\text{BERT-C}_{\text{large}}$	$\text{BERT-C}_{\text{base}}$
Parameters (Millions)	174	335	110	335	110
Pretraining Epochs	4	∼40	∼40	4	4
Pretraining Data: Base/Actual (Millions of Words)	508/427	3,300/ 3,300	3,300/ 3,300	508/508	508/508
Pretraining Time (FLOPS)	$\sim 1 \times 10^{11}$	$\sim 4.4 \times 10^{13}$	$\sim 1.4 \times 10^{13}$	$\sim 6.8 \times 10^{11}$	$\sim 2.2 \times 10^{11}$
RELPRON MAP (No-UNK/NE Subset)	0.651	0.769	0.690	0.056	0.193
SNLI Accuracy (Full Dataset)	81.0%	91.1%	90.7%	62.0%	79.9%
MegaVeridicality V2.1 Accuracy/Epochs	81.3%/11	85.6%/7	84.2%/7	76.2%/3	78.1%/15
McRae et al. (2005) Dataset Spearman ρ	0.205	0.241	0.247	0.134	0.167

Table 5.10: Summary of the main results of this chapter for GFoLDS and the four BERT models, along with a comparison of all five models' pretraining datasets and parameter counts. The distinction between *base* and *actual* pretraining data is discussed in Section 5.1.1.

While the FDS (Emerson, 2018) and FDSAS (Lo et al., 2023) models can arguably be considered to be pretrained LMs over logical forms, these models are thus far unable to be applied to (multi-)sentence-level NLP tasks such as NLI, due to architectural limitations. Aside from (likely) outperforming these models on the RELPRON dataset, GFoLDS' largest advantage over FDS/FDSAS is its ability to perform large-scale tasks with practical real-world applications, as is most saliently demonstrated in Section 5.3.

The experiments in this chapter, however, reveal critical limitations of the GFoLDS model. In particular, the model's inability to account for CARGs and OOV items (see Chapter 4) severely impeded its evaluation on the RELPRON (Section 5.2) and MegaVeridicality V2.1 (Section 5.4) datasets, and entirely precluded a direct comparison with FDS/FDSAS on the former benchmark. Although the results of Section 5.3 suggest that this limitation may not necessarily have a significant negative impact on all benchmarks, the inability to account for CARGs and OOV items still restricts the applicability of the model in real-world use cases: it is not able to provide an output for all possible inputs. In Chapter 7, I propose potential

future avenues to overcome this constraint.

Overall, this chapter demonstrates that the pretrained GFoLDS model is able to outperform superficial models trained on the same amount of data on a variety of downstream tasks. However, as discussed in Chapter 1, this model is only intended to serve as a proof-of-concept of language models over logical forms: it still fails to match the performance of superficial models pretrained on substantially more data. To make the case for language models over logical forms—and inform future directions in this research program—it is necessary to perform an in-depth analysis of the GFoLDS model in order to establish its scalability and determine its limitations.

Chapter 6

GFoLDS: Model Analysis

In Chapter 5, I evaluated the GFoLDS model on a series of NLP benchmarks, and demonstrated the utility and feasibility of language modeling over logical forms, and of GFoLDS in particular. Critically, I showed that when pretrained on the same amount of data, GFoLDS outperforms a comparably-sized, near-SoTA superficial model (i.e. BERT; Devlin et al., 2019) on a wide range of tasks. In this chapter, I turn from the comparison of GFoLDS against competing architectures to a fine-grained analysis of the model itself.

As discussed previously, one of the primary objectives of this dissertation is to demonstrate the viability of language modeling over logical forms. While Chapter 5 certainly shows that GFoLDS is a viable competitor to superficial models trained on similar amounts of data, this model is still outperformed by superficial models that are pretrained on substantially more data (i.e. the original BERT models of Chapter 5). In order to convincingly make the case that language models over logical forms represent a plausible means to continue the improvement of LLMs at a more sustainable rate of data consumption, it is crucial to establish the *scalability* of this model: the degree to which we would expect the model's performance to scale if it were larger—i.e. contained more parameters—and pre-trained on significantly more data. To that end, I dedicate Section 6.1 of this chapter to an analysis of the scalability of GFoLDS, and focus on evaluating the extent to which the current model is

under- or over-trained.

Chapter 1 stated the Accelerated Learning Hypothesis (ALH) as follows: (i) linguistically-informed LMs immediately begin learning more complex patterns, because the (aspects of) linguistic knowledge incorporated into such models obviates the need to learn elementary linguistic phenomena, and (ii) this accelerated learning of complex patterns allows linguistically-informed LMs to learn from less data than their superficial counterparts. Although the results of the experiments of Chapter 5 strongly support part (ii) of this hypothesis, they do not explain *why* language models over logical forms are able to learn useful representations with less data. To address this lack of evidence, Section 6.2 consists of a series of interrelated experiments that are designed to directly probe the validity of part (i) of the ALH.

As briefly discussed in Chapter 5, the experiments conducted therein revealed a major weakness of the GFoLDS model: its inability to include CARGs and out-of-vocabulary items (see Chapter 4) in its DMRS-derived input representations. In Section 6.3, I probe the model to identify any remaining limitations or weaknesses in its architecture, with the goal of informing the plans for future improvements to GFoLDS laid out in Chapter 7.

6.1 Scalability

As discussed above, this section is dedicated to an analysis of the scalability of the GFoLDS model. In Section 6.1.2, I apply the techniques of Hoffmann et al.'s (2022) and Muennighoff et al.'s (2024) research on the scalability of LLMs (discussed in Section 6.1.1) to GFoLDS, with the goal of estimating the model's propensity to scale with respect to parameter count and pretraining dataset size. The results of this experiment indicate that GFoLDS is likely scalable along both axes (see Section 6.1.3), both in terms of final pretraining loss and performance on downstream tasks.

6.1.1 Background

Hoffmann et al. (2022) investigate the effect of model size (i.e. number of parameters) and amount of pretraining data on the final pretraining cross-entropy loss of LLMs. The optimal number of parameters and amount of data measured in number of tokens for a fixed compute budget—expressed in terms of floating-point operations (FLOPs)—is given in Equation 6.1, 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.

$$(N_{opt}, D_{opt}) = \underset{(N,D)\,:\,FLOPs(N,D)=C}{argmin} L(N, D) \tag{6.1}$$

Evaluating over 400 language models trained on 70 million to 16 billion tokens, Hoffmann et al. (2022) employ the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS; Nocedal, 1980) algorithm to fit a model of LLM final pretraining loss as a function of N and D, minimizing the Huber loss (Huber, 1964) between predicted and observed cross-entropy. This model is given in Equation 6.2, where E, A, B, α, and β are learned constants[1].

$$L(N, D) \approx \hat{L}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \tag{6.2}$$

This model predicts that $N_{opt}, D_{opt} \propto \sqrt{6ND}$: critically, this implies that the optimal number of model parameters scales proportionally to the number of training tokens. This suggests that the vast majority of current language models are overparameterized (i.e. too large) for the amount of data on which they are trained. To illustrate this point, the authors train the *Chinchilla* language model with one-quarter of the parameters (70 billion) as *Gopher* (280 billion parameters; Rae et al., 2021)—but with the same compute budget: Chinchilla is pretrained on four times as many tokens as Gopher (1.4 trillion vs. 300 billion). The smaller Chinchilla model outperforms Gopher on all benchmarks on which Hoffmann et al. (2022)

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

evaluate the two models.

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 6.2 (Equation 6.3a), $\hat{L}^{(r)}(N, D)$, which estimates the final loss for a language model with N parameters trained on D tokens for r repetitions (i.e. epochs).

$$L^{(r)}(N, D) \approx \hat{L}^{(r)}(N, D) = E + \frac{A}{\hat{N}^{\alpha}} + \frac{B}{\hat{D}^{\beta}} \tag{6.3a}$$

$$\hat{D} = U_D + U_D R_D^*(1 - e^{-r/R_D^*}) \tag{6.3b}$$

$$\hat{N} = U_N + U_N R_N^*(1 - e^{-R_N/R_N^*}) \tag{6.3c}$$

$$U_N = min\{N, N_{opt}(U_D)\} \tag{6.3d}$$

$$R_N = \frac{N}{U_N} - 1 \tag{6.3e}$$

Where $U_D = D$ denotes the number of *unique* tokens (i.e. the amount of tokens in a single epoch), $N_{opt}(U_D)$ is estimated as in Hoffmann et al. (2022) (see Equations 6.1 and 6.2), and E, A, B, α, β, R_D^*, and R_N^* are learned constants[2] that are estimated via minimization of Huber loss with the L-BFGS algorithm (as in Hoffmann et al., 2022). This model suggests that pretraining LLMs for up to four epochs results in a negligible difference in loss in comparison to pretraining on $4U_D$ tokens for a single epoch, but additional epochs yield diminishing returns—and can eventually result in overfitting.

Of particular interest to the current discussion is Muennighoff et al.'s (2024) exploration of the impact of overparameterization in the setting of repeated data, which is incorporated into their model via the term R_N in Equation 6.3e. Specifically, their model predicts that an overparameterized model will outperform a model with fewer parameters, given the

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

same amount of pretraining tokens and epochs, albeit with rapidly diminishing returns (see Equation 6.3c).

6.1.2 Experimental Setup

In order to determine the degree to which GFoLDS (as defined and pretrained in Chapter 4) is under- or over-parameterized—and therefore, by the scaling laws established in Hoffmann et al. (2022) and Muennighoff et al. (2024), over-/under-trained—I set out to establish the effect of pretraining tokens on the model's final pretraining loss.

However, as loss is not an exact predictor of a given model's downstream performance (Shin et al., 2022; Tay et al., 2022; Xia et al., 2023), I follow Hoffmann et al. (2022) and evaluate the impact of pretraining tokens on a validation task as well. I used the RELPRON dataset (Rimell et al., 2016) for this purpose, as it is the only task in Chapter 5 that does not depend on factors external to the model: SNLI (Bowman et al., 2015) and MegaVeridicality V2.1 (White et al., 2018) introduce potential confounding factors in the form of the fine-tuning procedure and classification head, while the McRae et al. (2005) property inference task depends not only on the language model, but also on the performance of the ModAds (Talukdar and Crammer, 2009) label-propagation algorithm.

In this experiment, I pretrained five additional GFoLDS models on 50%, 25%, 12.5%, 6.25%, and 3.125% of the pretraining data used when originally pretraining the model in Chapter 4. I randomly-selected exactly half of the data of the preceding iteration to pretrain each model: the 50% run was pretrained on exactly half of the dataset used in Chapter 4, the 25% run on exactly half of that half, and so on—following Muennighoff et al. (2024), I ensured that the iterations pretrained on less data always use a subset of the dataset used in those with more data. All models used an identical architectural configuration—that of the GFoLDS model employed in Chapters 4 and 5—as the focus of this experiment was the effect of training tokens with a fixed parameter count. Aside from the differing number of pretraining tokens, all models were pretrained using the same procedure and hyperparameters

as employed in Chapter 4.

6.1.3 Results

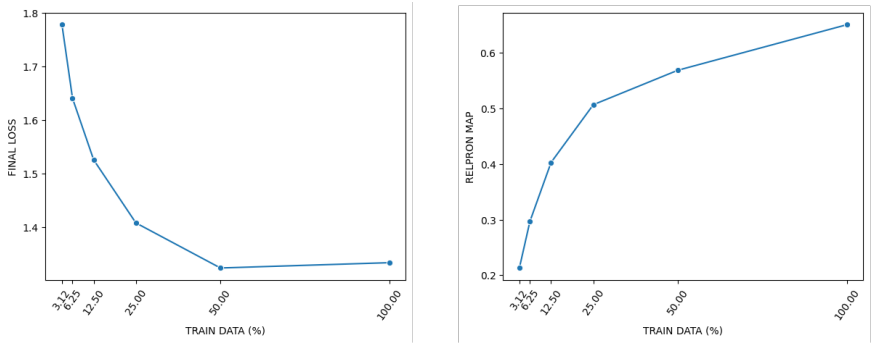


Figure 6.1: Final pretraining cross-entropy loss values and RELPRON MAP scores (No-UNK/NE test split; see Chapter 5) for each of the six training runs. The 100% run denotes the GFoLDS model pretrained in Chapter 4.

Figure 6.1 shows that final pretraining loss consistently decreases with the number of pretraining tokens from 3.125% to 50% of the data. After this point, however, the final loss value plateaus: the 50% run (1.3232) actually finishes with a *lower* cross-entropy loss than the actual (i.e. 100%) run (1.3331)—although this slight difference is likely explained by noise introduced by the random initialization of the models' parameters. This is to say that we should treat the model's loss with 50% of the pretraining data as roughly equivalent to its loss with 100% of the data.

Note that the term D in Equation 6.3 denotes the number of pretraining *tokens*. This metric is likely not a perfect predictor for GFoLDS: recall from Chapter 4 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 dissertation to establish exact scaling laws for GFoLDS and determine the graph-based analogue to the term D in Equation 6.3. However, it is clear that

such a D value scales (more or less) linearly with the number of graphs in the pretraining dataset: as with a superficial model, 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 $D' \approx D/2$, where D denotes the dataset for the 100% run, and D' that of the 50% run. We may express the relationship between the 50% run loss and the 100% run loss in the notation introduced in Section 6.1.1, while leaving the exact definition of D to future work: $L^{(4)}(N, D) \approx L^{(4)}(N, D/2)$. Recall from Equation 6.3a that the term E is a constant, and so can be ignored. We are then left with the following (approximate) equality in Equation 6.4a.

$$\frac{A}{\hat{N}_1^{\alpha}} + \frac{B}{\hat{D}_1^{\beta}} \approx \frac{A}{\hat{N}_2^{\alpha}} + \frac{B}{\hat{D}_2^{\beta}} \tag{6.4a}$$

$$\hat{D}_1 = U_D + U_D R_D^*(1 - e^{-4/R_D^*}) \tag{6.4b}$$

$$\hat{D}_2 \approx (U_D/2) + (U_D/2)R_D^*(1 - e^{-4/R_D^*}) \tag{6.4c}$$

$$\hat{N}_1 = U_N^{(1)} + U_N^{(1)} R_N^*(1 - e^{-R_N^{(1)}/R_N^*}) \tag{6.4d}$$

$$\hat{N}_2 = U_N^{(2)} + U_N^{(2)} R_N^*(1 - e^{-R_N^{(2)}/R_N^*}) \tag{6.4e}$$

$$U_N^{(1)} = min\{N, N_{opt}(U_D)\} \tag{6.4f}$$

$$U_N^{(2)} \approx min\{N, N_{opt}(U_D/2)\} \tag{6.4g}$$

$$R_N^{(k)} = \frac{N}{U_N^{(k)}} - 1 \tag{6.4h}$$

Where the left-hand expression in Equation 6.4a corresponds to the 100% run ($\hat{L}^{(4)}(N, D)$), and the right-hand expression corresponds to the 50% run ($\hat{L}^{(4)}(N, D/2)$). Note that $\hat{D}_2 \approx \hat{D}_1/2$ (Equation 6.5).

$$\begin{aligned}\hat{D}_2 &\approx \frac{U_D}{2} + \frac{U_D}{2} R_D^*(1 - e^{-4/R_D^*}) \\ &= \frac{U_D + U_D R_D^*(1 - e^{-4/R_D^*})}{2} = \frac{\hat{D}_1}{2}\end{aligned} \tag{6.5}$$

Given the vast architectural and modal differences between GFoLDS and superficial transformer models, I 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 dissertation 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, we know that the coefficients B and β in Equation 6.4 must be positive: this—in conjunction with the (approximate) equality in Equation 6.5—means that it cannot be the case that $B/\hat{D}_1^\beta > B/\hat{D}_2^\beta$.

Assume for the sake of argument that $B/\hat{D}_1^\beta \ll B/\hat{D}_2^\beta$: given the equality in Equation 6.4a, 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 6.4d). As the terms A and α are constants in Equation 6.3a—and therefore in Equation 6.4a—we may reduce to the inequality expressed in Equation 6.6, where the last logical equivalence is by definition of $R_N^{(2)}$ (see Equation 6.4h).

$$
\begin{aligned}
&\hat{N}_1^{-\alpha} \gg \hat{N}_2^{-\alpha} \leftrightarrow \hat{N}_1 \ll \hat{N}_2 \\
&\leftrightarrow N \ll U_N^{(2)} + U_N^{(2)} R_N^* (1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow \frac{N - U_N^{(2)}}{U_N^{(2)}} \ll R_N^* (1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow \frac{N}{U_N^{(2)}} - 1 \ll R_N^* (1 - e^{-R_N^{(2)}/R_N^*}) \\
&\leftrightarrow R_N^{(2)}/R_N^* \ll 1 - e^{-R_N^{(2)}/R_N^*}
\end{aligned}
\tag{6.6}
$$

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 the model's optimal number of parameters scales monotonically with the amount of pretraining data—i.e. that $N_{opt}(U_D/2) < N_{opt}(U_D)$—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 6.3c, 6.4d-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 $L^{(4)}(N, D) \approx L^{(4)}(N, D/2)$, 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 6.5), 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 $B/\hat{D}_1^{\beta} \approx 0$, which indicates that additional pretraining data will not serve to further decrease the model's final pretraining loss.

Given that $B/\hat{D}_1^{\beta} \approx B/\hat{D}_2^{\beta}$, the equality in Equation 6.4a 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 6.6 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 6.4d and 6.4e), which is congruent with the conclusion that $\hat{N}_1 \approx \hat{N}_2$.

Therefore, we may conclude that it is most likely the case that GFoLDS is underparameterized not only for the full pretraining dataset employed in Chapter 4, but also for a dataset half that size. This suggests that scaling to a larger model would likely result in increased model performance, even with the same amount of pretraining data.

Furthermore, recall that final model loss is not perfectly correlated with downstream model performance. Despite the plateau in cross-entropy loss observed in Figure 6.1, we observe improvement of +0.082 in MAP on the RELPRON validation task from the 50% (MAP = 0.569) to the 100% (MAP = 0.651) run. This is in fact *higher* than the +0.062 increase in MAP score from the 25% (MAP = 0.507) to the 50% run.

The results of this experiment lead to the conclusion that the GFoLDS model is likely scalable in terms of both model size and pretraining data. The application of the Muennighoff et al. (2024) scaling laws to GFoLDS indicates that increasing the model's parameter count will likely lead to decreased final pretraining loss—assuming that the scaling laws hold for

GFoLDS—while the model's run-over-run performance on the RELPRON test set suggests that increased pretraining dataset size will most likely result in corresponding increases in the GFoLDS model's downstream performance.

6.2 The Accelerated Learning Hypothesis

As mentioned above, this section is dedicated to an investigation of the validity of part (i) of the Accelerated Learning Hypothesis (ALH) of Chapter 1, which posits that the (aspects of) linguistic knowledge incorporated into linguistically-informed LMs obviates the need to learn elementary linguistic phenomena, allowing them to immediately begin learning more complex patterns. In the context of the current discussion, it is critical to note that there is a temporal aspect to this part of the hypothesis: in order to validate the claim that linguistically-informed LMs *immediately* begin to learn complex patterns, we must probe the model at regular intervals throughout its pretraining procedure, in order to locate the point(s) at which it begins to learn those patterns.

Additionally, this part of the ALH can be broken down into two distinct claims: (i) that the aspects of linguistic knowledge incorporated into linguistically-informed LMs obviate the need to learn elementary linguistic phenomena; and (ii) that this enables such models to then immediately begin learning more complex patterns. I therefore divide the experiment in this section into two parts, which respectively probe the model's knowledge of elementary and complex linguistic phenomena.

While this chapter is devoted to analyses of GFoLDS itself—rather than a comparison to competing models—part (ii) of the ALH states that that language models over logical forms can learn useful representations with less data than their superficial counterparts: this necessitates the comparison of GFoLDS to a superficial model in this section. In these experiments, I specifically compare GFoLDS to the BERT comparison models employed in Chapter 5. There are two main advantages to using these particular models as baselines

for comparison: first, I pretrained these models myself, and therefore have access to the model weights at various stages of the pretraining process. Furthermore, these models were pretrained on the same base data as GFoLDS, reducing the risk of confounding factors that may arise from distributional differences in the models' respective pretraining corpora.

6.2.1 Experimental Setup

This experiment revolves around four probing tasks, consisting of two sets of two probes each, that are respectively designed to evaluate the models' knowledge of elementary and complex linguistic phenomena. I evaluate GFoLDS and the BERT comparison models on each of the four tasks at twenty evenly-spaced intervals per epoch, for a total of eighty points of comparison.

Under the assumption that part (i) of the ALH holds, we should expect to see GFoLDS outperform the BERT comparison models on the complex tasks at each stage of the pre-training process, as—according to this hypothesis—GFoLDS is able to learn complex patterns faster than BERT. On the elementary tasks, we still expect GFoLDS to outperform BERT, but also predict that the GFoLDS model's performance will *not* improve significantly across pre-training stages on the elementary probes, while BERT's performance on these tasks steadily increases: part (i) of the ALH asserts that—by its very nature—a linguistically-informed LM is already equipped with elementary linguistic knowledge, and therefore its performance at the first pre-training stage should be near (or above) that of a superficial model such as BERT at the *last* pre-training stage. In other words, assuming that part (i) of the ALH holds, the GFoLDS model's performance on the elementary probing tasks shouldn't improve significantly across pre-training stages, because there shouldn't be any room for the model to improve—it should already be near peak performance from the start.

The complex probes that I employ in this experiment are simply the MegaVeridicality V2.1 (White et al., 2018) factuality and RELPRON (Rimell et al., 2016) relative-clause composition tasks used in Chapter 5, with their respective evaluation metrics. The McRae

et al. (2005) property inference dataset was not suitable to use as a complex probe, as none of the fully-pretrained GFoLDS and BERT-comparison models performed particularly well on this task, and so it would not serve as an effective metric of improvement over time. Conversely, while SNLI would likely serve as an effective benchmark, the fine-tuning times on this dataset are prohibitively long: on this dataset, GFoLDS and $\text{BERT}_{\text{base}}$ each take roughly five hours to train on an NVIDIA RTX 3080 Ti GPU[3], and $\text{BERT}_{\text{large}}$ takes roughly 15 hours. Given that I evaluated each model at 80 points during pretraining, fine-tuning each of the GFoLDS and $\text{BERT}_{\text{base}}$ checkpoints on SNLI would require ∼800 hours: along with the ∼1200 required for $\text{BERT}_{\text{large}}$, fine-tuning all three models on SNLI 80 times would necessitate ∼2000 hours—over 83 days.

6.2.1.1 Elementary Probes

As discussed above, I created two probing tasks designed to evaluate the models' knowledge of elementary linguistic phenomena. The first task, *POS-prediction*, evaluates the LMs' ability to model the distribution of parts-of-speech within a sentence: this is a commonly-employed probing task used to assess the elementary linguistic competence of language models (Waldis et al., 2024).

For this task, I evaluated the models on 200 sentences drawn from English Wikipedia, that were not used in the models' pretraining data. I first parsed each sentence using the ACE/ERG DMRS parser employed in Chapters 4 and 5, which automatically labels the part-of-speech of each predicate in the DMRS representation of a sentence. I 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 the part-of-speech of the selected word. This resulted in a dataset consisting of 58 masked-quantifier, 28

[3]While training times would likely be significantly faster on an H100 or A100 on UB's CCR servers (which I used for pretraining in Chapter 4), I am using the free tier and therefore face substantial queue times for these training runs. This is compounded by CCR's "fair-share" policy, which further extends queue times as a function of the user's resource usage over the last 30 days.

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

masked-preposition, 33 masked-verb, 63 masked-noun, and 18 masked-adjective sentences.

I extracted the models' probability distributions 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 the model's top ten most-likely predictions were all nouns, then the model received a perfect score (1.0) for that example. If nine of the top ten predictions were nouns, the model received a score of 0.9 for that example, and so on.

Determining the part-of-speech for each prediction of the GFoLDS model was trivial: DMRS predicates include part-of-speech tags, so I simply checked the tag of each predicted predicate. The BERT models, however, necessitated the use of the NLTK POS tagger[4]. For each sentence s, and each of the model's top-ten predicted tokens w for s, I 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, Sørensen, and Derczynski, 2021), and therefore is sufficiently robust to yield an estimate of the model's performance.

I 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). I recorded each model's mean precision across all 200 sentences as its final score for this task.

The second elementary probe, *quantifier-agreement*, evaluates the models' knowledge of

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

quantifier number agreement—i.e. whether a quantifier restricts a singular or plural noun (or both)—across 179 sentences drawn from English Wikipedia (as in the POS-prediction probe described above, none of these sentences were 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 linguistic competence (see e.g. Huebner et al., 2021; Waldis et al., 2024).

I 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. I then randomly selected a single quantifier from each sentence to mask, and recorded the number of the noun in the quantifier's restriction.

I sorted all of the quantifiers into one of three categories/types (see Table 6.1): *singular*, consisting of quantifiers which can only restrict singular nouns; *plural*, which can only restrict plural nouns; and *both*, which can restrict either kind of noun. As in the POS-prediction task above, I extracted the models' probability distributions over the masked quantifier of each sentence, in order to compute their precision at k. Here, k varied as a function of the number of the masked quantifier's restriction: $k = 14$ for the quantifiers that restrict singular nouns—eight singular-type quantifiers, along with the six both-type quantifiers—and $k = 13$ for the plural-noun-restricting quantifiers. 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, I recorded each model's mean precision across all 179 sentences as its final score for this task.

6.2.2 Results

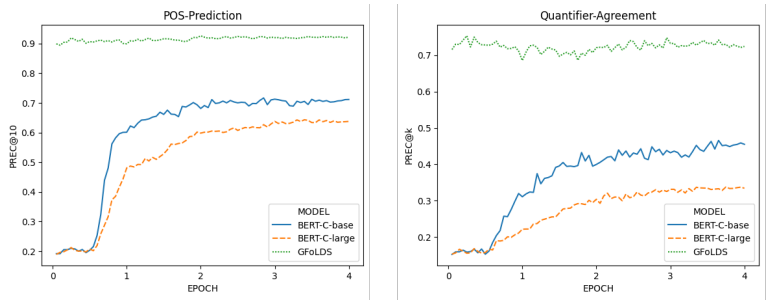


Figure 6.2: Precision scores across the 80 evenly-spaced training snapshots (20 per epoch) for GFoLDS and the BERT comparison models on the two elementary tasks.

The results of the two elementary probing tasks described in Section 6.2.1.1 are shown in Figure 6.2. The results for all three models conform almost exactly to their predicted behavior discussed in Section 6.2.1: the performance of GFoLDS on these probes remains relatively constant throughout pretraining—because the model is already near peak performance from the beginning—while that of the BERT comparison models (more or less) steadily increases. Additionally, note that the BERT models do not begin to improve on these tasks until roughly halfway through the first pretraining epoch.

These results constitute strong evidence towards the first half of part (i) of the Accelerated Learning Hypothesis: namely, that the aspects of linguistic knowledge incorporated into linguistically-informed LMs obviates the need to learn elementary linguistic phenomena. It is clear from these experiments that GFoLDS is able to model these elementary phenomena from the onset, and retains this ability throughout pretraining.

It now remains to evaluate the second half of part (i) of the ALH: linguistically-informed LMs' built-in knowledge of elementary linguistic phenomena enables these models to then immediately begin learning more complex patterns. To that end, I first examined the performance of these models on the RELPRON test set, which is given in Figure 6.3. Again, these results resemble almost exactly the behavior predicted by the ALH: GFoLDS begins improving immediately, while the performance of BERT-C_{base} does not begin to meaningfully increase until the latter half of the first epoch (and BERT-C_{large} does not improve substantially

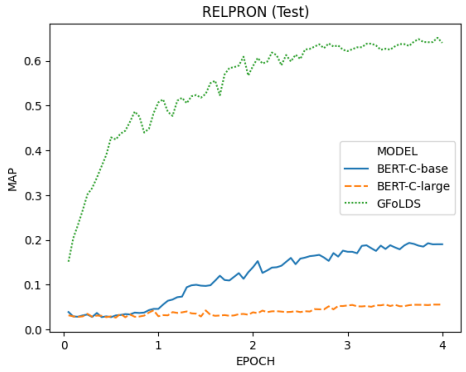


Figure 6.3: MAP scores on the RELPRON test set across the 80 training snapshots for GFoLDS and the BERT comparison models.

at all)—and at a much lower rate than that of GFoLDS.

The results of the factuality experiment (see Figure 6.4) are not nearly as straightforward: the accuracy of all three models on the MegaVeridicality V2.1 dataset is rather erratic throughout pretraining. While this task is clearly not as effective as RELPRON or the elementary probes as a metric of improvement over time, the other tasks introduced in Chapter 5 (property inference and NLI) are not suitable in that role, as discussed in Section 6.2.1.

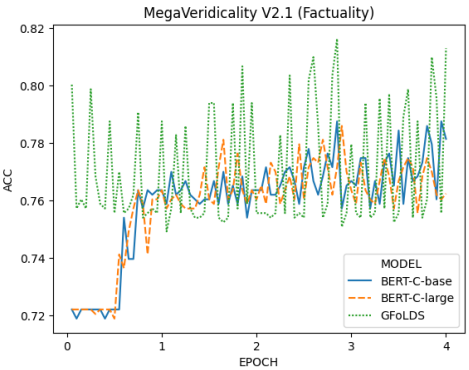


Figure 6.4: Accuracy on the MegaVeridicality V2.1 factuality dataset (test set) across training snapshots for GFoLDS and the BERT comparison models.

Although this data is much noisier than that of the previous experiments in this section, we can still extract useful information with respect to the ALH: it appears that across the 80 pretraining snapshots, the central tendency of GFoLDS's performance was higher than

that of the BERT comparison models. To formalize this intuition, I smoothed the data in Figure 6.4 with a 1-dimensional Gaussian filter ($\sigma = 3$), which measures its cross-correlation with the Gaussian distribution with a mean of zero and standard deviation of σ (Young and Van Vliet, 1995); Gaussian filters are commonly used in image and signal processing to smooth and minimize rise/fall time in noisy data (Mafi et al., 2019).

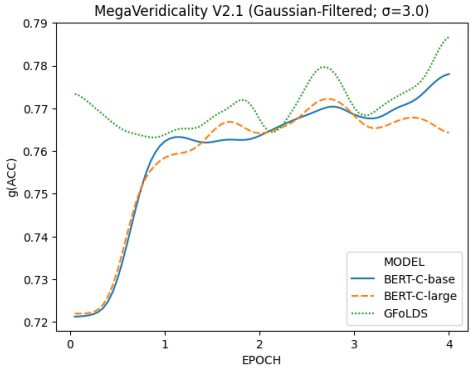


Figure 6.5: Results of Figure 6.4, smoothed via Gaussian filter ($\sigma = 3.0$).

The Gaussian-filtered MegaVeridicality V2.1 data shown in Figure 6.5 provides a much clearer picture than Figure 6.4. While the evidence provided by this experiment is not nearly as strong as that of the RELPRON experiment discussed above, these results do still lend support to the ALH. Although the smoothed performance of GFoLDS actually decreases across the majority of the first epoch, this decrease spans less than two percentage points, and the model's performance improves thereafter—albeit with drops at the ends of the second and third epochs. Furthermore, the Gaussian-filtered values for GFoLDS are almost constantly above those of the BERT comparison models, with the exception of a brief period near the end of the second epoch. We again observe the performance of the BERT comparison models starting from a lower value than that of GFoLDS, and (more or less) steadily increasing.

Note that the performance of the BERT comparison models remains relatively static until roughly halfway through the first epoch (this is more clear in Figure 6.4): the same point at which they begin to improve on the elementary tasks (see Figure 6.2). We observe the same pattern with BERT-C_{base} for the RELPRON experiment in Figure 6.3 (the performance

of BERT-C_{large} remains effectively constant throughout). While this temporal relationship between complex and elementary task performance for the BERT-C models obviously does not demonstrate causation, it does suggest a relationship between model performance on the elementary and complex tasks.

Overall, the results of the elementary and complex probing experiments constitute strong evidence in support of part (i) of the Accelerated Learning Hypothesis. The elementary probes (Figure 6.2) clearly demonstrate that GFoLDS is able to learn elementary linguistic phenomena far more rapidly than the BERT comparison models, and the results of the complex tasks—in particular, the RELPRON experiment (Figure 6.3)—suggest that this does in fact correspond to faster learning of more complex patterns.

6.3 Limitations and Weaknesses

In this section, I probe the GFoLDS model's ability to inductively learn the law of the excluded middle (Section 6.3.1), replicating the experiment conducted in Chapter 2. I find that, as was the case with the superficial models in that experiment, GFoLDS is unable to inductively learn to cancel double negation with respect to NLI tasks. However, unlike the superficial LMs, I show in Section 6.3.2 that GFoLDS' poor performance on this task is likely due to a proven architectural failing (see Theorem 2), rather than the model's inability to learn the logical role of negation. In Section 6.3.3, I demonstrate that this architectural weakness also inhibits the model's ability to encode the global positions of nodes within a graph, likely hindering its performance on multi-sentence tasks such as NLI.

Fortunately, all of the limitations uncovered in this section can be traced back to the same root cause—that uncovered by Theorem 2—and I propose a potential solution to this problem in Chapter 7.

6.3.1 Double-Negation Cancellation

In Chapter 2, I investigated the ability of near-SoTA superficial transformer NLI models to inductively learn the law of the excluded middle (LEM)—i.e. double-negation cancellation. Specifically, I probed these models' capacity to generalize LEM to lengths of repeated negation longer than what was seen during fine-tuning. Here, I replicate those experiments with the GFoLDS model on the SNLI (Bowman et al., 2015) dataset. For the sake of convenience, I provide a brief overview of the experimental setup below; a more comprehensive description is located in Chapter 2.

6.3.1.1 Task Description

For each $1 \leq n \leq 5$, I constructed training sets $D^{\leq n}$ by randomly selecting 9,999 examples from the SNLI train split (3,333 examples from each class: *entailment*, *contradiction*, and *neutral*). For each $1 \leq k \leq n$, $1/n$ of the examples in each class in $D^{\leq n}$ are depth-k negated by prepending the *trigger* prefix T_{NT} = *"it is not true that"* to the original hypothesis sentence k times (i.e. by converting (P, H) to $(P, (T_{NT})^k H)$). For example, in $D^{\leq 5}$, 1/5 of the examples in each class are depth-5 negated, 1/5 are depth-4 negated, 1/5 are depth-3 negated, etc.

External negation changes the examples' class labels in a predictable way: after odd-depth negation—i.e. an odd number of repeated external negation prefixes—an example that was originally *entailment* becomes *contradiction*, and vice-versa, while *neutral* examples do not change class labels. Even-depth negation, on the other hand, does not change the examples' class labels, as double negation cancels out. A more detailed discussion of this phenomenon is given in Chapter 2.

Then, for each $1 \leq m \leq 8$, I constructed the depth-m test set D^m_{NT}. The test sets are constructed in a similar manner to the train sets $D^{\leq n}$, the difference being that the examples are *only* depth-m negated in D^m_{NT}—whereas $1/n$ of the examples in $D^{\leq n}$ are depth-k negated for each $1 \leq k \leq n$.

After constructing the train and test sets, I conducted 30 training runs, inoculating (see

Liu, Schwartz, and Smith, 2019) each of the six models on each of the five training sets $D^{\leq n}$ ($1 \leq n \leq 5$). After fine tuning each model on $D^{\leq n}$, I then evaluated the models' performance on D^m_{NT} for each $n < m \leq 8$, in order to determine their ability to generalize beyond what they had seen during inoculation/fine-tuning.

I found that of the six models, only the three RoBERTa models were able to generalize up to depth-8 after being inoculated on $D^{\leq 5}$. I then found that, after depth-≤ 5 inoculation on T_{NT}, the RoBERTa models were unable to generalize this knowledge to the highly similar prefix "*it is false that*". A more comprehensive overview of the findings from these experiments is located in Chapter 2.

To replicate these experiment(s) with GFoLDS, I used the GFoLDS model fine-tuned on the entire SNLI dataset in Chapter 5. I did not replicate Experiment 2 of Chapter 2—i.e. generalization to "*it is false that*" (see Section 2.5)—with GFoLDS, because both "*it is not true that*" and "*it is false that*" map to the same representation in logical form (*neg*).

6.3.1.2 Results

The results of this experiment (Table 6.2) show that GFoLDS is unable to inductively learn double-negation cancellation with respect to NLI tasks. As with the superficial models evaluated in Chapter 2, these results appear to seriously call into question the ability of GFoLDS to learn to reason logically, at least when fine-tuned on NLI datasets.

Depth-m Test	**B/I**	**Inoc-1**	**Inoc-2**	**Inoc-3**	**Inoc-4**	**Inoc-5**
1	0.63	0.89	—	—	—	—
2	0.30	0.35	0.88	—	—	—
3	0.58	0.61	0.31	0.87	—	—
4	0.38	0.49	0.88	0.30	0.87	—
5	0.55	0.53	0.31	0.88	0.31	0.62
6	0.48	0.49	0.88	0.30	0.88	0.48
7	0.47	0.48	0.30	0.87	0.30	0.58

Table 6.2: Accuracy for the GFoLDS model on depth-($m \geq n$) external negation (D^m_{NT}) after depth-$\leq n$ inoculation ($1 \leq n \leq 5$) on $D^{\leq n}$ (column *Inoc-n*). Column *B/I* denotes the model before inoculation (i.e. the GFoLDS model fine-tuned on SNLI in Chapter 5).

Depth-m Test	GFoLDS	Mean	Median	Max	Min
1	0.63	0.53	0.52	0.69	0.39
2	0.30	0.72	0.76	0.79	0.56
3	0.58	0.36	0.35	0.40	0.32
4	0.38	0.84	0.84	0.85	0.82
5	0.55	0.32	0.32	0.34	0.30
6	0.48	0.86	0.86	0.89	0.83
7	0.47	0.31	0.32	0.36	0.28

Table 6.3: Non-inoculated accuracy of the GFoLDS model on the depth-m external-negation test sets (D^m_{NT}) for $1 \leq m \leq 7$, compared to the mean, median, maximum, and minimum accuracy of the six superficial models evaluated in Chapter 2.

Depth-m Test	GFoLDS	Mean	Median	Max	Min
1	0.78	0.60	0.59	0.78	0.43
2	0.36	0.82	0.85	0.91	0.62
3	0.71	0.40	0.39	0.46	0.37
4	0.47	0.94	0.95	0.95	0.92
5	0.67	0.36	0.36	0.39	0.33
6	0.59	0.97	0.97	1.0	0.94
7	0.58	0.36	0.36	0.41	0.31

Table 6.4: Table 6.3, viewed as a proportion of each respective model's accuracy on the original SNLI development split.

However, note that the non-inoculated GFoLDS model—i.e. that fine-tuned on the standard SNLI dataset in Chapter 5—exhibits a pattern that is almost entirely opposite to that displayed by the superficial language models of Chapter 2 (see Table 6.3). Specifically, for the superficial models, we observe high accuracy (∼80%) on the even-depth test sets, and low accuracy on the odd-depth test sets. GFoLDS, on the other hand, exhibits higher accuracy on the odd-depth test sets than on the even-depth splits. When the models' test-set performance is viewed as a *proportion* of their accuracy on the original SNLI development split—measuring the degree to which repeated external negation degrades their performance—the contrast between the pattern displayed by GFoLDS and that of the superficial models is far more pronounced (see Table 6.4).

In Chapter 2, I argued that the pattern displayed by the superficial models in Table 6.3 suggests that, before inoculation, they had learned to essentially entirely ignore the

external negation prefixes and treat them as distractors—despite fine-tuning on SNLI, which is (ostensibly) a logical-reasoning task: depth-m negation does not alter an example's class label for even values of m, and so an LM treating the prefix as a distractor will retain high accuracy on those examples, without any need to learn to model negation.

Following this same line of reasoning, if a model had learned to model negation—but not double-negation cancellation—we would expect it to flip the entailment and contradiction class labels on both the odd-depth *and* even-depth test sets, resulting in higher accuracy on the odd-depth test sets than on the even-depth splits: (more or less) exactly what we observe with GFoLDS. This suggests that, before inoculation, GFoLDS may have more effectively learned the logical role of negation than its superficial counterparts—that the model did not learn to model double-negation cancellation is likely due to the sparsity of such constructions in its pretraining data and the SNLI dataset: double negation is exceedingly rare in real-world data (Larrivée, 2016).

6.3.2 Mod-2 Counting

The findings of Section 6.3.1 are rather puzzling: the results in Tables 6.3 and 6.4 suggest that GFoLDS had more effectively learned the logical role of negation before inoculation than its superficial counterparts, which raises the question as to why GFoLDS is unable to inductively learn to cancel double-negation (see Table 6.2).

6.3.2.1 Theoretical Results

In Theorem 1 of Chapter 2, I proved that superficial transformer encoders are theoretically able to *model* cancellation of repeated external negation, suggesting that their empirically observed inability to do so results from the (lack of) structure of purely textual data and/or their training procedures. However, as a *graph* transformer (Wu et al., 2021), GFoLDS does not satisfy the assumptions of Theorem 1, and so that theorem cannot be expected to hold for this model.

In Theorem 2 below, I prove that GFoLDS has a critical failing in the architecture of its positional encoding module that renders it unable to distinguish between the embeddings of nodes in the middle of a chain of repeated instances of the same token longer than twice the number of step-wise aggregation (SWA; described in detail in Section 4.2.2 of Chapter 4) layers in the model—such as the repeated external negation prefix used in the experiment in Section 6.3.1. This suggests that its inability to cancel double negation exhibited in Section 6.3.1 is in fact due to an architectural limitation.

Theorem 2. *Given a GFoLDS model (as defined in Chapter 4) M with n SWA layers and an input graph G, let $M(G)_i$ denote the embedding that M assigns to the i^{th} node x_i of G. Suppose that G contains a path $p = x_1 \xrightarrow{\ell} \ldots \xrightarrow{\ell} x_k$ of length k such that all nodes (and all edges) in p have the same label (respectively), and that for all $1 < i < k$, x_i has no incoming or outgoing edges not in p. Then:*

i. for all $1 \leq i \leq n$, and all $1 \leq j \leq k$ such that $i \neq j$: $M(G)_i \neq M(G)_j$

ii. for all $k - n \leq i \leq k$, and all $1 \leq j \leq k$ such that $i \neq j$: $M(G)_i \neq M(G)_j$

iii. for all $n < i, j < k - n$: $M(G)_i = M(G)_j$

Proof. Section 6.5. □

A critical consequence of Theorem 2 is that, given a length-$(k-1)$ path $p = x_1 \xrightarrow{\ell} \ldots \xrightarrow{\ell} x_k$ consisting of identical node and argument labels, the GFoLDS model requires at least $\lfloor k/2 \rfloor$ SWA layers in order to assign a unique embedding to each node in p. The external negation introduced in the double-negation cancellation experiment of Section 6.3.1 results in exactly such a path: $p^{(k)}_{neg} = (neg)_1 \xrightarrow{ARG1} \ldots \xrightarrow{ARG1} (neg)_k$ (see Figure 6.6). GFoLDS therefore requires three or more SWA layers in order to assign unique embeddings to each *neg* token in $p^{(6)}_{neg}$ and $p^{(7)}_{neg}$ sequences: the GFoLDS model used in all experiments in this dissertation (with the exception of Section 6.3.2.2 below) has only two SWA layers.

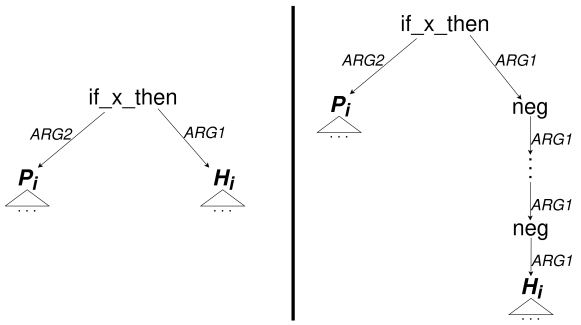


Figure 6.6: Example of an SNLI premise/hypothesis graph before (left) and after (right) repeated external negation. For the sake of representational simplicity, an edge $X \xrightarrow{\ell} \boldsymbol{Y_i}$ (i.e. with a bolded target) in the figure denotes an edge $X \xrightarrow{\ell} \mathit{htop}(G(Y_i))$ in the actual graph (see the discussion of DMRS *htop* nodes in Section 5.3.1.1 of Chapter 5).

For sequence classification tasks, GFoLDS employs mean pooling over graph node embeddings to yield a vector representation $e(\vec{G})$ of a given input graph G, as defined in Equation 6.7.

$$e(\vec{G}) = \sum_{x \in V(G)} \frac{M(\vec{G})_x}{|V(G)|} \tag{6.7}$$

In order to successfully model double-negation cancellation in the experiment of Section 6.3.1, GFoLDS must be able to determine whether there is an even or odd number of external negation prefixes in each hypothesis sentence. But by Theorem 2, the model's embeddings of the third through $(k-3)^{th}$ *neg* tokens are identical for any p^k_{neg} sequence with $k \geq 6$.

The impact of these tokens on the resulting graph embedding $e(\vec{G})$ is therefore negligible. To illustrate this, let n_3 denote the third *neg* token in p^k_{neg}, and let $N_3^{k-2} = \{n_3, \ldots, n_{k-2}\}$ be the set containing the third through $(k-2)^{th}$ *neg* tokens. For graphs G containing a p^k_{neg} sequence[5] with $k \geq 5$, the sum in Equation 6.7 is then equivalent to the following formula in Equation 6.8.

[5]In the case $k = 5$, $N_3^{k-2} = \{n_3\}$, and so the model can still assign unique embeddings to each node in $p^{(5)}_{neg}$.

$$
\begin{aligned}
e(\vec{G}) &= \left(\sum_{z \in N_3^{k-2}} \frac{M(\vec{G})_z}{|V(G)|} \right) + \left(\sum_{x \in V(G) - N_3^{k-2}} \frac{M(\vec{G})_x}{|V(G)|} \right) \\
&= \left(\frac{(k-4)}{|V(G)|} \cdot M(\vec{G})_{n_3} \right) + \left(\sum_{x \in V(G) - N_3^{k-2}} \frac{M(\vec{G})_x}{|V(G)|} \right)
\end{aligned}
\tag{6.8}
$$

When factoring in the noise that results from the fact that the input graphs G in the double-negation cancellation experiment of Section 6.3.1 contain varying numbers of nodes with varying labels, it is almost impossible for the model to distinguish an input graph containing five external negation prefixes from a graph containing $k > 5$ prefixes. For $k > 5$, it is therefore exceedingly difficult for the model to determine if the length of a given p^k_{neg} sequence is even or odd, which—as discussed above—is critical for the double-negation cancellation task of Section 6.3.1.

Superficial models such as BERT—which have learned positional embeddings for each token in their input sequence—do not suffer from this limitation. This is to say that such models are able to determine whether the number of external negation prefixes in a given sequence is even or odd: their inability to inductively learn to cancel double negation arises from another, yet-to-be-determined weakness (as discussed in Chapter 2).

6.3.2.2 Experimental Results

In order to verify Theorem 2 experimentally, I evaluated four GFoLDS architectures: one model with n SWA layers for each $1 \leq n \leq 4$—aside from the number of SWA layers, these models are architecturally identical to that introduced in Chapter 4. As the number of SWA layers in three of the four models differs from that employed in the rest of the dissertation, all four models were evaluated with reset parameters.

The models' objective in this probing task was to classify a given $p^{(k)}_{neg}$ sequence as either *odd* or *even*, for $1 \leq k \leq 12$. For each $1 \leq k \leq 12$, I trained each model on 24 $p^{(i)}_{neg}$ sequences

for each $1 \leq i \leq k$: each input graph consisted of *only* a $p^{(i)}_{neg}$ sequence, and did not contain any other tokens or edges.

The test data for this experiment was identical to the training data: the objective was to determine whether the models are *architecturally capable* of counting modulo 2 the number of nodes in the input sequence. For each $1 \leq k \leq 12$, I trained each model on the $p^{(i)}_{neg}$ sequences (for all $1 \leq i \leq k$) until it reached 100% classification accuracy on the test set—which is, again, identical to the training set—halting training at 100 epochs if the model was unable to achieve perfect test accuracy. All models were trained with a learn rate of 10^{-5} and a batch size of 24.

Sequence Length	**1-SWA**	**2-SWA**	**3-SWA**	**4-SWA**
1	✓	✓	✓	✓
2	✓	✓	✓	✓
3	✓	✓	✓	✓
4	X	✓	✓	✓
5	X	✓	✓	✓
6	X	X	✓	✓
7	X	X	✓	✓
8	X	X	X	✓
9	X	X	X	✓
10	X	X	X	✓
11	X	X	X	✓
12	X	X	X	X

Table 6.5: Results of the mod-2 counting experiment for four GFoLDS models with n SWA layers ($1 \leq n \leq 4$). A ✓ symbol at row r indicates that the model was able to correctly classify each $p^{(k)}_{neg}$ sequence as odd or even within 100 training epochs, for all $1 \leq k \leq r$.

The results of this experiment are shown in Table 6.5. For $1 \leq n \leq 3$, the n-SWA layer GFoLDS model is only able to correctly classify $p^{(k)}_{neg}$ sequences for $k \leq n + 2$, as predicted by the conclusions of Theorem 2.

The 4-SWA layer model is able to correctly classify sequences up to a length of 11—two more than predicted. However, Theorem 2 only states that this model will assign identical embeddings to the fifth through $(k-5)^{th}$ *neg* tokens: it does not necessarily claim that a GFoLDS model with n SWA layers is unable to *classify* the (mod 2) length of $p^{(k)}_{neg}$ sequences of

$k > n+2$: the conclusions of the theorem merely indicate that it is exceedingly difficult for the model to do so. In particular, although Theorem 2 and Equation 6.8 predict that the graph embeddings of $p_{neg}^{(k_1)}$ and $p_{neg}^{(k_2)}$ (where $k_1, k_2 > n+2$ and $k_1 \neq k_2$) will be very similar, they are not *identical*: the term $M(\vec{G})_{n_3}$ in Equation 6.8 is scaled by $(k_1 - 8)/|V(G)|$ for $p_{neg}^{(k_1)}$, and by $(k_2 - 8)/|V(G)|$ for $p_{neg}^{(k_2)}$. Given the ten encoder layers after its SWA stack, it is not infeasible that the model could learn to leverage the small differences between graph embeddings to yield accurate predictions for $p_{neg}^{(k>n+2)}$ sequences—although the results for the $\{1, 2, 3\}$-SWA models show that this is still rather difficult—especially absent the noise that exists in the external-negation data of Section 6.3.1: namely, other, non-*neg* premise/hypothesis sentence tokens.

6.3.2.3 Discussion

The results of the experiment in Section 6.3.2.2—along with Theorem 2—shed light on GFoLDS' inability to learn to cancel double negation exhibited in Section 6.3.1: due to a critical architectural weakness with respect to the model's positional embedding module, it is effectively unable to determine whether the length of a sequence of six or more repeated external negation prefixes is even or odd. The ability to determine the (mod 2) length of a given negation sequence is essential for determining the polarity of the hypothesis sentence, and therefore the class label of a given example.

While Theorem 2 and the results of the experiment in Section 6.3.2.2 suggest that GFoLDS may be able to learn to cancel double negation within the bounds of the experiment of Section 6.3.1—i.e. for sequences up to length seven—with an additional SWA layer[6], such a solution would merely address the symptoms, rather than the cause, of the problem: with three SWA layers, the model would still fail to extrapolate double-negation cancellation to $p_{neg}^{(k)}$ sequences for $k > 7$, and so on. The results of this section (and of Section 6.3.1) therefore indicate

[6]An experiment to evaluate this hypothesis would require pretraining a second, 3-SWA layer GFoLDS model from scratch. Due primarily to distance-related constraints (the desktop computer that I used to parse the model's pretraining data is located at the University at Buffalo), I estimated that this process would take a minimum of three to four months, and would therefore not be feasible for this dissertation.

the need to redesign the positional encoding architecture of the model in order to overcome this limitation. I leave the exploration of alternative approaches to the positional encoding module to the discussion of future work in Chapter 7.

Furthermore, even after inoculation on depths 1-4, GFoLDS is unable to inductively learn to cancel negation up to depth 5 (see Table 6.2)—i.e. within the bounds of its architectural limitations. This is to say that GFoLDS also fails to inductively learn the law of the excluded middle (LEM). In Chapter 2, I argued that one cause of the superficial LMs' inability to inductively learn LEM may be the procedure used to (pre-)train them: as the GFoLDS model's pretraining and SNLI fine-tuning procedures was essentially identical to that of those superficial models, it may be the case that GFoLDS' failure to learn inductively arose from the same (pre-)training deficiencies. I leave further investigation into the cause of these models' failure to learn inductively to future work.

6.3.3 Sentence Membership Classification

The GFoLDS model's inability to determine the (mod 2) length of sequences indicates that it is likely unable to determine the global location of a given node in a graph structure as the size of the graph increases. In this section, I evaluate that hypothesis by assessing the model's ability to determine whether a given node is in the first or second sentence of a two-sentence pair.

6.3.3.1 Experiment

I used the SNLI development and test splits as the training and test sets (respectively) for this experiment, employing the same procedure as in the SNLI experiment of Chapter 5 (described in detail in Section 5.3.1.1) to convert each premise/hypothesis pair into a single graph: adding an *if_x_then* node, and inserting edges $\textit{if\_x\_then} \xrightarrow{ARG1} \textit{htop}(G(H))$ and $\textit{if\_x\_then} \xrightarrow{ARG2} \textit{htop}(G(P))$, as illustrated in Figure 6.7.

The model's objective was then to classify each node (except for the inserted *if_x_then*)

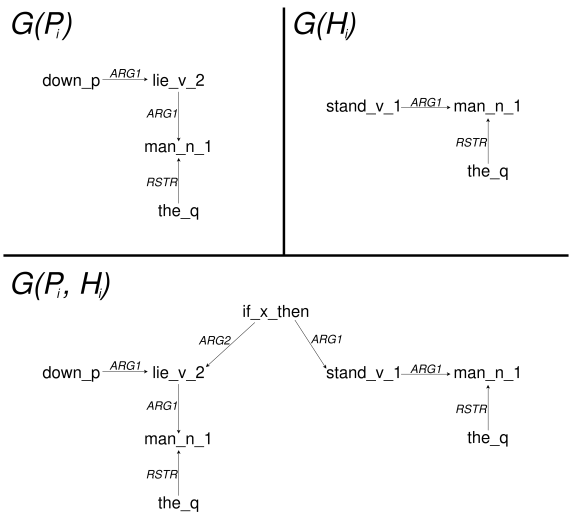


Figure 6.7: Illustration of the derivation of a single graph from a premise (P_i), hypothesis (H_i) sentence pair (duplicated from Figure 5.4)

in each input graph as belonging to either $S_1 = P$ or $S_2 = H$: if a node $n \in G(P_i, H_i)$ originated from P_i (see Figure 6.7), then its class label is S_1. Conversely, if n originated from H_i, then its class label is S_2. As the label of a given node n is determined entirely by the label of the edge from the *if_x_then* node to the *htop* of the subgraph to which n belongs, this task is an effective proxy for measuring the model's ability to encode the global positions of nodes in a graph structure.

For this experiment, I employed a GFoLDS model with the same architecture as described in Chapter 4—i.e. that used in all experiments in this dissertation aside from that of Section 6.3.2.2—and reset the model's parameters to avoid any confounding factors: for example, the model fine-tuned on SNLI may have learned that certain tokens occur more frequently in hypothesis sentences. I trained the model with a learn rate of 5×10^{-6}, weight decay of 5×10^{-6}, and a batch size of 16 for an indefinite number of epochs, halting training when combined S_1/S_2 test set accuracy failed to increase: the model reached convergence after eleven epochs.

6.3.3.2 Results

The results of this experiment (see Table 6.6) show that S_2 classification accuracy begins to sharply decline after a depth—i.e. undirected distance from the *if_ x_ then* node—of two. Note that the GFoLDS model employed in this experiment only has two SWA layers: as dicussed in Chapter 4, the message-passing distance of the model's positional encoding module is limited by the number of SWA layers—with two SWA layers, nodes can only pass messages within their two-hop neighborhoods. While the transformer encoder built into the model's architecture permits global *attention* between nodes, these results demonstrate that this does not result in global graph *positional encoding*. This is to say that nodes at distances greater than two are able to attend to the *if_ x_ then* node, but their positional encodings do not suffice to determine their position relative to the *if_ x_ then* node in the graph.

While S_1 accuracy does not begin to substantially decay until depth 15, this is easily explained by an imbalance in class-label frequency at higher depths: premises (S_1) in the SNLI development (the training set for this experiment) and test splits are on average over twice as long as their corresponding hypothesis (S_2) sentences. As premise sentences are longer, they tend to have more nodes at a further distance from the *htop* than hypothesis sentences. This is to say that, beyond a distance of two from *htop*—at which point the model cannot determine distance from *htop*—the model can learn that it is more likely to guess the correct class label if it defaults to S_1.

This conjecture is not contradicted by the fact that the model does not attain near-perfect S_1 accuracy on depths 15-19 in Table 6.6 despite the lack of S_2 nodes at those depths, as the *development* set—which the model was trained on—does have S_2 nodes at depths 15-19.

These results indicate that GFoLDS is unable to encode the global positions of nodes within graphs. This deficiency likely had a negative impact on GFoLDS' performance in the SNLI experiment of Chapter 5, as the model is unable to accurately predict whether a given node belongs to the premise sentence or to the hypothesis sentence. This calls into question the model's applicability to other multi-sentence tasks—although the model was not designed

Depth	S_1 Accuracy	S_2 Accuracy	Total Accuracy
Overall	87.0%	72.3%	82.1%
1	80.3%	82.4%	81.3%
2	86.3%	89.0%	87.6%
3	83.9%	66.8%	76.5%
4	88.5%	72.5%	83.6%
5	89.3%	61.9%	82.8%
6	89.6%	50.5%	82.6%
7	89.2%	45.3%	83.3%
8	88.5%	46.3%	84.3%
9	87.4%	45.4%	84.1%
10	88.2%	42.1%	85.3%
11	88.5%	43.4%	86.5%
12	87.1%	26.9%	84.6%
13	88.0%	0.0%	85.9%
14	87.4%	0.0%	85.4%
15	77.1%	—	77.1%
16	62.5%	—	62.5%
17	58.8%	—	58.8%
18	66.7%	—	66.7%
19	33.3%	—	33.3%

Table 6.6: GFoLDS' test set accuracy on the S_1/S_2 classification task, by node depth (undirected distance from the *if_x_then* node). *Total accuracy* denotes the accuracy for all S_1 and S_2 nodes at the depth in question. The — symbol in the S_2 column for depths 15-19 indicates that there were no S_2 nodes in the test set at those depths.

with such tasks in mind, as discussed in Chapter 5.

Fortunately, this problem stems from the same underlying cause as that of the limitations uncovered in Sections 6.3.1 and 6.3.2: the model's positional encoding architecture. As discussed in Section 6.3.2.3, I defer the discussion of potential remedies for this limitation to Chapter 7.

6.4 Discussion

The scalability experiment of Section 6.1 strongly suggests that GFoLDS is scalable in terms of final pretraining loss and downstream task performance. The fact that the model is likely to scale with respect to parameter count and pretraining dataset size represents a significant

step towards achieving one of the primary objectives of this dissertation: demonstrating the viability of language modeling over logical forms. These results indicate the applicability of language models over logical forms in real-world world settings, suggesting that such models are not merely esoteric curiosities, but rather have the potential to compete with superficial LMs at scale.

As discussed in Chapter 1, Villalobos et al. (2024) estimate that, given the Chinchilla Scaling Laws (Hoffmann et al., 2022) and the rate at which SoTA LLMs are expanding, the stock of available high-quality natural language training data will be exhausted at some point between 2026 and 2032. As recent advances in LLM performance have been primarily driven by increasing model size—and necessarily, increasing training dataset size—(Muennighoff et al., 2024) this impeding exhaustion of available training data represents a potential ceiling with respect to the further improvement of language models' performance.

However, the results of Section 6.1 indicate that GFoLDS (∼174 million parameters) is *under*parameterized with half of the pretraining data used in Chapter 4: i.e. at ∼254 million tokens. For comparison, Hoffmann et al.'s (2022) scaling laws suggest that $\text{BERT}_{\text{large}}$ (∼330 million parameters) is *over*parameterized for its pretraining corpus of ∼3.3 billion tokens. This is to say that GFoLDS requires significantly less pretraining data per parameter than superficial LMs: Hoffmann et al.'s (2022) scaling laws predict that a superficial LLM with the same parameter count as GFoLDS necessitates ∼5 billion pretraining tokens—roughly twenty times more than the ∼254 million tokens at which GFoLDS becomes overparameterized. Language models over logical forms may therefore present an avenue for continuing the improvement of language models at a more sustainable rate of data consumption than their superficial counterparts.

In addition, the results in Section 6.1 provide further evidence in support of the Accelerated Learning Hypothesis (ALH) of Chapter 1: namely, that language models over logical forms learn useful representations with less data than their superficial counterparts.

Section 6.2 provides direct evidence in support of part (i) of the ALH. I show that GFoLDS

maintains near-peak performance on two elementary probing tasks (POS prediction and quantifier agreement) throughout pretraining, starting at the first point of measurement (5% of the first epoch). This suggests that the linguistic knowledge injected into GFoLDS by means of its DMRS-derived input graphs drastically facilitates the model's learning of basic linguistic phenomena. In contrast, neither BERT comparison model even begins to improve on the elementary tasks until halfway through the first pretraining epoch.

Although GFoLDS' DMRS input graphs do not explicitly encode POS[7], they contain sufficient information to reconstruct this data: for example, only nouns have person and number features; only verbs have tense and mood features; quantifier nodes do not have features; etc. In contrast, a superficial model such as BERT learns words' parts-of-speech from much noisier information than GFoLDS: namely, textual co-occurrence. Similarly, the quantifier-agreement task is likely facilitated by the fact that the number of the head noun in each quantifier's restriction is directly encoded via features in GFoLDS' DMRS-derived inputs. BERT, on the other hand, must first learn the patterns in surface text that correspond to pluralization—and distinguish them from verbal person/number agreement—to be able to differentiate singular from plural nouns, before it can begin to learn which quantifiers co-occur with each type of noun.

This built-in linguistic knowledge likely facilitates GFoLDS' performance on more complex tasks, as demonstrated by the model's performance relative to that of the BERT comparison models on the RELPRON test set—and, to a lesser degree, the MegaVeridicality V2.1 factuality task: GFoLDS immediately begins improving, while the BERT models do not begin to see progress until approximately halfway through the first epoch—roughly the same point at which they begin to improve on the elementary tasks.

Together, the results of GFoLDS and the BERT comparison models on the elementary and complex tasks in Section 6.2 strongly support part (i) of the ALH: the (aspects of)

[7]While categories (parts-of-speech) *are* directly encoded in DMRS predicate labels (as suffixes of these strings), GFoLDS does not have access to this information: each predicate label is tokenized into an integer before being passed to the model's embedding layer (see Chapter 4).

linguistic knowledge incorporated into linguistically-informed LMs obviates the need to learn elementary linguistic phenomena, allowing them to immediately begin learning more complex patterns.

Section 6.3, however, reveals a serious limitation in the GFoLDS model's positional encoding architecture, which likely inhibits performance on a range of multi-sentence-level NLP tasks. This limitation results in the model's inability to count long sequences of repeated nodes—which are critical to the negation-cancellation task of Section 6.3.1—and determine the sentence to which a given node belongs, which may present a serious impairment to GFoLDS' performance on NLI tasks.

In Chapter 7, I propose a new positional encoding architecture to overcome this critical weakness in the model's architecture. I additionally discuss additional pretraining objectives intended to improve its performance on multi-sentence tasks such as NLI, and potential future directions—such as graph-based text generation—for language models over logical forms.

6.5 Proof of Theorem 2

In this section, I formally prove Theorem 2 of Section 6.3.2.1: uninterested readers may safely skip to Chapter 7. Although the main idea of the proof is conceptually rather simple, it requires a considerable amount of machinery. For this reason, I provide a conceptual sketch in Section 6.5.1 in order to illustrate the intuition behind the formal proof given in Section 6.5.2.

6.5.1 Proof Sketch

As discussed in Section 6.3.2.1, the main consequence of Theorem 2 is that, given a length-$(k-1)$ path $p^{(k)}_{neg} = neg \xrightarrow{ARG1} \ldots \xrightarrow{ARG1} neg$, a GFoLDS model with n SWA layers will assign identical embeddings to the $(n+1)^{th}$ through $(k-n)^{th}$ nodes in $p^{(k)}_{neg}$.

Recall from the discussion of the GFoLDS architecture in Chapter 4 that each SWA layer

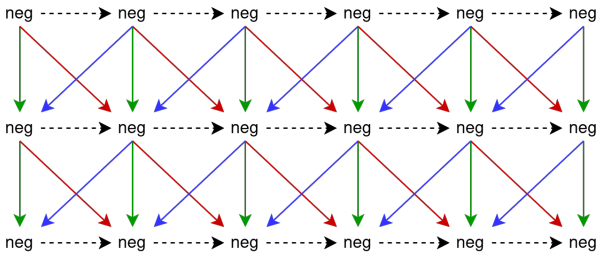


Figure 6.8: Illustration of the application of two SWA layers to the sequence $p^{(6)}_{neg}$ (see Section 6.3.2). The top row represents the input to the first layer (the output of the embedding layer), the second row the output of the first layer (equivalently, the input to the second layer), and the bottom row the output of the second layer. Green arrows indicate the identity (i.e. skip/residual) connections, red arrows the forward SWA block ($W^{(f)}_{ARG1}$), and blue arrows the backward SWA block ($W^{(b)}_{ARG1}$) of each layer. Dashed arrows denote the $neg \xrightarrow{ARG1} neg$ edges of the input graph $p^{(6)}_{neg}$.

aggregates local neighborhoods: with n SWA layers, the model only passes messages within a given node's n-hop neighborhood. The critical idea of the proof of Theorem 2 is that in the sequence $p^{(k)}_{neg}$, the $(n+1)^{th}$ through $(k-n)^{th}$ *neg* nodes have identical n-hop neighborhoods.

As an example, the application of two SWA layers to the sequence $p^{(6)}_{neg}$ is illustrated in Figure 6.8. The output embedding of the SWA stack for the i^{th} node in the sequence is entirely determined by the shape—i.e. labeled graph isomorphism class—of the subgraph $\mathcal{F}(y^{(2)}_i)$ spanned by all paths[8] from the input of the SWA stack (represented by the top row of Figure 6.8) to the i^{th} node in the output of the SWA stack (represented by the bottom row of Figure 6.8). This is to say that $\mathcal{F}(y^{(2)}_i)$ encodes all of the transformations performed by the SWA stack on the i^{th} node embedding: all nodes in $p^{(k)}_{neg}$ have the same label (*neg*)—and so have identical input embeddings to the SWA stack—and all edges have the same label, so all transformations within the SWA stack will be applications of the $W^{(f)}_{ARG1}$ and $W^{(b)}_{ARG1}$ projections in each SWA layer (see Section 4.2.2 of Chapter 4).

As shown in Figure 6.9, each of the subgraphs on the top and bottom rows—$\mathcal{F}(y^{(2)}_1)$ (Figure 6.9a), $\mathcal{F}(y^{(2)}_2)$ (Figure 6.9b), $\mathcal{F}(y^{(2)}_5)$ (Figure 6.9e), and $\mathcal{F}(y^{(2)}_6)$ (Figure 6.9f)—are unique: the first, second, fifth, and sixth *neg* nodes in $p^{(6)}_{neg}$ can receive unique embeddings

[8]Excluding those paths containing dashed arrows in Figure 6.8, which are only included in the figure for reference.

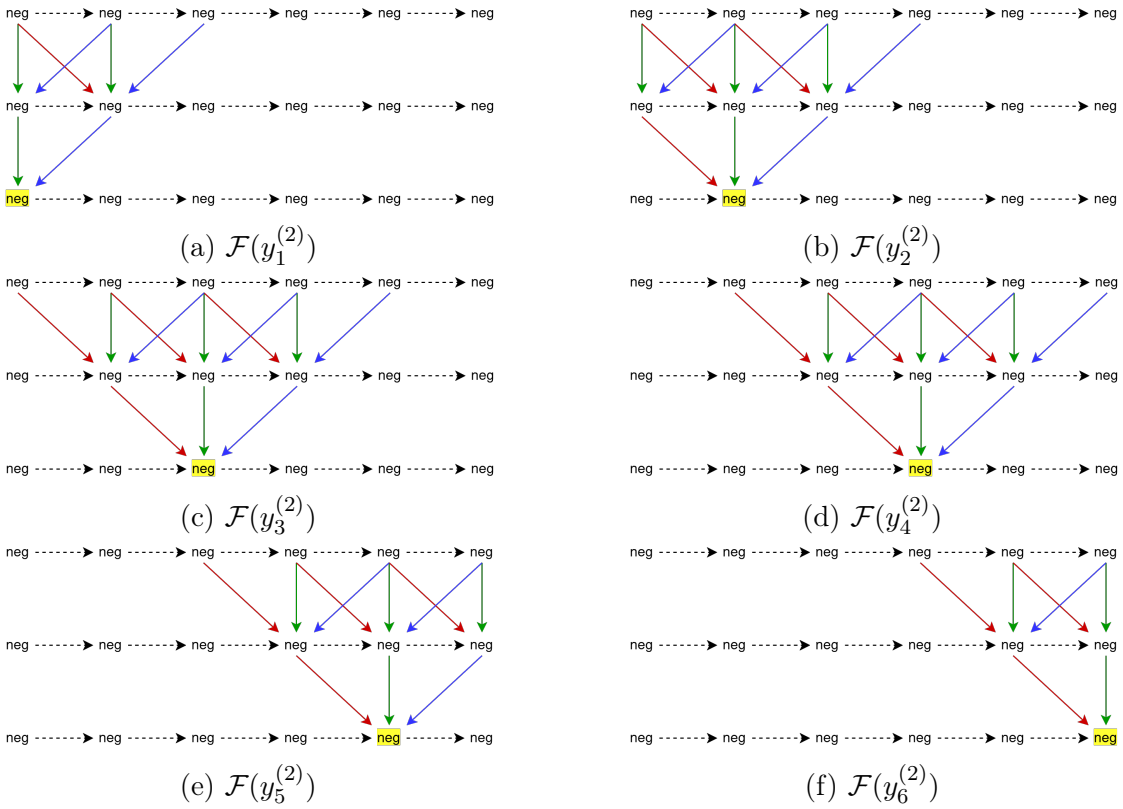


Figure 6.9: $\mathcal{F}(y_i^{(2)})$ subgraphs of the graph in Figure 6.8 for each *neg* node in the sequence $p_{neg}^{(6)}$. The i^{th} output (i.e. bottom-row) node in each $\mathcal{F}(y_i^{(2)})$ subgraph is highlighted for reference.

from a GFoLDS model with two SWA layers. However, the two graphs in the middle row are identical—i.e. $\mathcal{F}(y_3^{(2)}) \simeq \mathcal{F}(y_4^{(2)})$ (see Figures 6.9c-6.9d)—meaning that a two-layer SWA stack must assign identical embeddings to the third and fourth nodes in $p_{neg}^{(6)}$.

It is then straightforward to show that the third and fourth *neg* nodes will receive identical embeddings from the GFoLDS model: the embeddings outputted by the SWA stack for these two nodes are identical, so they will attend to all other nodes in an identical manner within the encoder stack.

6.5.2 Formal Proof

For all $1 \leq i \leq k$, let $y_i^{(0)} = \emptyset$, and for all $1 \leq i \leq k$ and all $1 \leq m \leq n$, define $y_i^{(m)}$ as follows in Equation 6.9a:

$$y_i^{(m)} = f_m^{(F)}(y_i^{(m-1)}, i) \cup f_m^{(B)}(y_i^{(m-1)}, i) \cup \{y_i^{(m-1)}\} \tag{6.9a}$$

$$f_m^{(F)}(y_i^{(m-1)}, i) = \begin{cases} \emptyset & \text{if } i = 1 \\ \{(0, \emptyset)\} & \text{if } m = 1 \\ \{(0, y_{i-1}^{(m-1)})\} & \text{otherwise} \end{cases} \tag{6.9b}$$

$$f_m^{(B)}(y_i^{(m-1)}, i) = \begin{cases} \emptyset & \text{if } i = n \\ \{(1, \emptyset)\} & \text{if } m = 1 \\ \{(1, y_{i+1}^{(m-1)})\} & \text{otherwise} \end{cases} \tag{6.9c}$$

Now, let $SWA(G)_i$ denote the embedding that the SWA module of M assigns to the i^{th} node x_i of G. I first prove the following lemma (Lemma 6.1):

Lemma 6.1. *For all* $1 \leq i, j \leq k$: $y_i^{(n)} = y_j^{(n)} \leftrightarrow SWA(G)_i = SWA(G)_j$.

Proof. First, note that the layer norm and feed-forward blocks of an SWA layer are vector-wise functions: they will always map identical input vectors to identical output vectors. For the purposes of this proof, those aspects of the model architecture can therefore be ignored.

Now, we observe that all node and argument labels in the path p are identical, by assumption. This implies that for any given nodes x_i, x_j in p, $SWA(G)_i$ and $SWA(G)_j$ are entirely differentiated by the history of applications of the forward and backward projection layers of each SWA layer in the model.

By definition, for all $1 \leq m \leq n$, $y_i^{(m)}$ encodes *exactly* the history of applications of the forward and backward projection layers of the first to m^{th} SWA layers. □

Define the mapping $\mathcal{F}\colon y_{(-)}^{(-)} \to \text{Gph}$ as in Equation 6.10a, where Gph denotes the class of labeled, directed graphs.

$$\mathcal{F}(X) = \begin{cases} (\{(0,0)\}, \emptyset) & \text{if } X = \emptyset \\ \left(\bigcup_{x \to y \in \mathcal{F}_E(X,(0,0))} \{x, y\}, \mathcal{F}_E(X, (0,0)) \right) & \text{otherwise} \end{cases} \tag{6.10a}$$

$$\mathcal{F}_E(X,t) = \begin{cases} \mathcal{F}_E^{(F)}(Y,t) \cup \mathcal{F}_E^{(I)}(W,t) & \text{if } X = \{(0,Y), W\} \\ \mathcal{F}_E^{(B)}(Z,t) \cup \mathcal{F}_E^{(I)}(W,t) & \text{if } X = \{(1,Z), W\} \\ \emptyset & \text{if } X = \emptyset \\ \hat{\mathcal{F}}_E(X,t) & \text{otherwise} \end{cases} \tag{6.10b}$$

$$\mathcal{F}_E^{(I)}(X,(a,b)) = \{(a,b-1) \xrightarrow{I} (a,b)\} \cup \mathcal{F}_E(X,(a,b-1)) \tag{6.10c}$$

$$\mathcal{F}_E^{(F)}(X,(a,b)) = \{(a-1,b-1) \xrightarrow{F} (a,b)\} \cup \mathcal{F}_E(X,(a-1,b-1)) \tag{6.10d}$$

$$\mathcal{F}_E^{(B)}(X,(a,b)) = \{(a+1,b-1) \xrightarrow{B} (a,b)\} \cup \mathcal{F}_E(X,(a+1,b-1)) \tag{6.10e}$$

$$\hat{\mathcal{F}}_E(\{(0,Y),(1,Z),W\},t) = \mathcal{F}_E^{(I)}(W,t) \cup \mathcal{F}_E^{(F)}(Y,t) \cup \mathcal{F}_E^{(B)}(Z,t) \tag{6.10f}$$

Lemma 6.2. *For all $1 \leq i, j \leq k$ and all $0 \leq m \leq n$: $\mathcal{F}(y_i^{(m)}) = \mathcal{F}(y_j^{(m)}) \leftrightarrow y_i^{(m)} = y_j^{(m)}$.*

Proof. The left-to-right direction follows from the fact that $\mathcal{F}$ is a function. It remains to prove the injectivity of $\mathcal{F}$—i.e. $\mathcal{F}(y_i^{(m)}) = \mathcal{F}(y_j^{(m)}) \to y_i^{(m)} = y_j^{(m)}$.

Note that each $y_{(-)}^{(m)}$ belongs to the class $\mathcal{Y}$ of structures, which consists of sets of the form $\{(0,X),(1,Y),Z\}$, $\{(1,Y),Z\}$, $\{(0,X),Z\}$, $\{Z\}$, or $\emptyset$, where X, Y, and Z are in the class $\mathcal{Y}$. The local neighborhood of the root node $(0,0)$ of any $\mathcal{F}(y_q^{(m)})$ is entirely determined by the pattern of $\mathcal{Y}$ that matches $y_q^{(m)}$. Each daughter node d of the root $(0,0)$ is itself the root node of a the subgraph of $\mathcal{F}(y_q^{(m)})$ spanned by all nodes x such that there exists a path $x \Rightarrow d$, and the local neighborhood of d is entirely determined by the structure of X, Y, or Z. By induction, each $\mathcal{F}(y_q^{(m)})$ is entirely determined by $y_q^{(m)}$. □

Lemma 6.3. *For all $1 \leq m \leq n$:*

i. for all $1 \leq i \leq m$, there does not exist $j \neq i$ such that $y_i^{(m)} = y_j^{(m)}$

ii. for all $(k-m) \leq i \leq k$*, there does not exist* $j \neq i$ *such that* $y_i^{(m)} = y_j^{(m)}$

Proof. I prove (i): the proof of (ii) is formally dual. The proof proceeds by contradiction: assume there exists some $j \neq i$ such that $y_i^{(m)} = y_j^{(m)}$. Then $\mathcal{F}(y_i^{(m)}) = \mathcal{F}(y_j^{(m)})$.

By construction (Definition 6.10), $\mathcal{F}(y_i^{(m)})$ contains a path $p_0 = (1-i, -m) \xrightarrow{I} \ldots \xrightarrow{I} (1-i, 1-i) \xrightarrow{F} \ldots \xrightarrow{F} (0,0)$, which consists of exactly $m+1-i$ I-labeled edges, followed by exactly $i-1$ F-labeled edges, and a path $p_1 = (k-i, -m) \xrightarrow{I} \ldots \xrightarrow{I} (k-i, i-k) \xrightarrow{B} \ldots \xrightarrow{B} (0,0)$, containing exactly $m+i-k$ I-labeled edges, followed by exactly $k-i$ B-labeled edges.

By the assumption that $y_i^{(m)} = y_j^{(m)}$, $\mathcal{F}(y_j^{(m)})$ also contains a sequence $p_0' = x_0 \xrightarrow{I} \ldots \xrightarrow{I} x_q \xrightarrow{F} \ldots \xrightarrow{F} x_r$, containing exactly $m+1-i$ I-labeled edges, followed by exactly $i-1$ F-labeled edges. By construction, the length of the longest path in $\mathcal{F}(y_j^{(m)})$ is m, therefore $x_r = (0,0)$. Again by construction, $\mathcal{F}(y_j^{(m)})$ only contains F-labeled edges $(a-1, b-1) \xrightarrow{F} (a,b)$ (for all (a,b) such that $a > 1-i$, $b > -m$)—therefore, $x_q = (1-i, 1-i)$. Note that—once again by construction—for any t, $min\{a \mid (a,b) \in V(\mathcal{F}(y_t^{(m)}))\} = max(1-t, -m)$, where $V(\mathcal{F}(y_t^{(m)}))$ denotes the set of nodes in $\mathcal{F}(y_t^{(m)})$. Therefore, the existence of $(1-i, 1-i)$ in $V(\mathcal{F}(y_j^{(m)}))$ implies that $j \geq i$.

Again by the assumption that $y_i^{(m)} = y_j^{(m)}$, $\mathcal{F}(y_j^{(m)})$ contains a sequence $p_1' = x_0 \xrightarrow{I} \ldots \xrightarrow{I} x_q \xrightarrow{B} \ldots \xrightarrow{B} x_r$, containing exactly $m+i-k$ I-labeled edges, followed by exactly $k-1$ B-labeled edges. As the length of the longest path in $\mathcal{F}(y_j^{(m)})$ is m, $x_r = (0,0)$. By construction, $\mathcal{F}(y_j^{(m)})$ only contains B-labeled edges $(a+1, b-1) \xrightarrow{F} (a,b)$ (for all (a,b) such that $a < k-i$, $b > -m$)—therefore, $x_q = (k-i, i-k)$. Again by construction, for any t, $max\{a \mid (a,b) \in V(\mathcal{F}(y_t^{(m)}))\} = min(k-t, m)$: therefore, the existence of $(k-i, i-k)$ in $V(\mathcal{F}(y_j^{(m)}))$ implies $j \leq i$. This, combined with the above result, implies that $j = i$: this is a contradiction. □

Lemma 6.4. *For all* $m < i, j \leq k-m$*:* $y_i^{(m)} = y_j^{(m)}$

Proof. By construction, $max\{a \mid (a,b) \in V(\mathcal{F}(y_t^{(m)}))\} = min(k-t, m)$ and $min\{a \mid (a,b) \in V(\mathcal{F}(y_t^{(m)}))\} = max(1-t, -m)$. As $i \leq k-m$ and $j \leq k-m$, $max\{a \mid (a,b) \in V(\mathcal{F}(y_i^{(m)}))\} =$

$max\{a \mid (a,b) \in V(\mathcal{F}(y_j^{(m)}))\} = m$. As $m < i$ and $m < j$, $min\{a \mid (a,b) \in V(\mathcal{F}(y_i^{(m)}))\} = min\{a \mid (a,b) \in V(\mathcal{F}(y_j^{(m)}))\} = -m$. As the length of the longest path in both graphs is m, $V(\mathcal{F}(y_i^{(m)}))$ and $V(\mathcal{F}(y_j^{(m)}))$ both contain $\{(-q,-q) \mid 0 \leq q \leq m\} \cup \{(q,-q) \mid 0 \leq q \leq m\}$, and do not contain any node (a,b) such that there exists some $1 \leq q \leq m$ such that $b < -q \wedge (a < -q \vee a > q)$. By construction, for any t and any $(a,b) \in V(\mathcal{F}(y_t^{(m)}))$, $V(\mathcal{F}(y_t^{(m)}))$ contains $\{(c,b) \mid -m \leq c < a\}$: therefore, $V(\mathcal{F}(y_i^{(m)})) = V(\mathcal{F}(y_j^{(m)}))$.

Again by construction, the edges of $\mathcal{F}(y_t^{(m)})$ are entirely determined by $V(\mathcal{F}(y_t^{(m)}))$ for any t. This implies that $\mathcal{F}(y_i^{(m)}) = \mathcal{F}(y_j^{(m)})$, which, by Lemma 6.2, in turn implies that $y_i^{(m)} = y_j^{(m)}$. □

By Lemma 6.3, for all $i,j \leq n$ such that $i \neq j$, and all $i',j' \geq k-n$ such that $i' \neq j'$: $y_i^{(m)} \neq y_j^{(m)}$ and $y_{i'}^{(m)} \neq y_{j'}^{(m)}$;. By Lemma 6.1, this implies that $SWA(G)_i \neq SWA(G)_j$ and $SWA(G)_{i'} \neq SWA(G)_{j'}$. By Lemma 6.4, for all $n < i,j < k-n$, $y_i^{(m)} = y_j^{(m)}$. Lemma 6.1 then implies that $SWA(G)_i = SWA(G)_j$.

It remains to show that $SWA(G)_i = SWA(G)_j \leftrightarrow M(G)_i = M(G)_j$. Note that, as in the SWA layers, the layer norm and feed-forward blocks of the GFoLDS encoder layers always map identical input vectors to identical output vectors: for the purposes of this proof, they can be ignored. It therefore suffices to prove that, for each multi-head attention block MHA, $\vec{x_i} = \vec{x_j} \rightarrow MHA(X)_i = MHA(X)_j$.

For any q, $MHA(X)_q$ is defined as in Equation 6.11, where $||$ denotes vector concatenation, H the number of attention heads, $A^{(h)}$ the h^{th} attention head, and $W^{(O)}$ is a $\mathbb{R}^{d_{model}} \times \mathbb{R}^{d_{model}}$ matrix.

$$MHA(X)_q = \left(\Big\|_{h=1}^{H} A^{(h)}(X)_q \right) W^{(O)} \tag{6.11}$$

Again, $W^{(O)}$ always maps identical input vectors to identical output vectors, and so can be ignored for the purposes of this proof. It therefore suffices to prove that $\vec{x_i} = \vec{x_j} \rightarrow A^{(h)}(X)_i = A^{(h)}(X)_j$ for all $1 \leq h \leq H$. Each attention head $A^{(h)}(X)$ is defined as in Equation 6.12,

where $\hat{Q} = XQ$, $\hat{K} = XK$, and $\hat{V} = XV$ (Q, K and V are $\mathbb{R}^{d_{model}} \times \mathbb{R}^{d_{model}/H}$ projection matrices).

$$A^{(h)}(X) = softmax\left(\frac{(\hat{Q}W^{(Q,h)})(\hat{K}W^{(K,h)})^\top}{\sqrt{d_{key}}}\right)\hat{V}W^{(V,h)} \tag{6.12}$$

The term $\sqrt{d_{key}}$ is a normalizing constant and can therefore be ignored. For any q, $A(X)_q$ is then equivalent to the formula given in Equation 6.13, where N denotes the number of input tokens.

$$A^{(h)}(X)_q = \sum_{r=1}^{N}((\hat{Q}W^{(Q,h)})_q \cdot (\hat{K}W^{(K,h)})_r)(\hat{V}W^{(V,h)})_r \tag{6.13}$$

Note that $(\hat{Q}W^{(Q,h)})_q = \vec{x_q}QW^{(Q,h)}$, $(\hat{K}W^{(K,h)})_r = \vec{x_r}KW^{(K,h)}$, and $(\hat{V}W^{(V,h)})_r = \vec{x_r}VW^{(V,h)}$. The assumption that $\vec{x_i} = \vec{x_j}$ therefore implies that $(\hat{Q}W^{(Q,h)})_i = (\hat{Q}W^{(Q,h)})_j$. This in turn implies that $A^{(h)}(X)_i = A^{(h)}(X)_j$, by definition (Equation 6.13).

This completes the proof of Theorem 2.

Chapter 7

Future Directions

This chapter is dedicated to a variety of proposed future directions in the research program of language modeling over logical forms. I first discuss a potential application of the GFoLDS model that was left unexplored in this dissertation: namely, its use for lower-resource languages such as German, Turkish, and Korean (Section 7.1).

In several portions of Chapters 4-6, I identified limitations and weaknesses of the GFoLDS model: Section 7.2 proposes adjustments to the current GFoLDS architecture and pretraining procedure that are intended to both address those limitations and yield further improvements to the model's performance. I then dedicate Section 7.3 to a discussion of the next step in the research program that I have outlined over the course of this dissertation: a graph-to-graph generative model over logical forms.

7.1 Applications to Lower-Resource Languages

GFoLDS' demonstrated capability to learn more from less data than superficial LMs in Chapters 5-6 naturally suggests its applications in lower-resource settings. Take, for example, German, Turkish, and Korean, all of which are fairly major languages, with roughly 95

million[1], 80 million[2], and 78 million[3] native speakers, respectively.

Text in these languages, however, constitutes a fairly small proportion of the overall stock of available written language data: in the 2024 Common Crawl[4] release, German text represents 5.3948% of the overall data, 0.9864% is in Turkish, and only 0.6504% of the data is written in Korean. For comparison, English text comprises 46.4536% of the 2024 Common Crawl release.

From Villalobos et al.'s (2024) estimate of 7% growth in the stock of available language data per year, it follows that it will take ∼32, ∼57, and ∼63 years for the stock of German, Turkish, and Korean language data (respectively) to reach the amount of present-day English data. Given rule-based grammars that can generate (D)MRS representations for these languages, a more data-economical model such as GFoLDS would allow for more rapid progress towards higher-quality LMs in these languages.

Although constructing a rule-based, broad-coverage grammar for a language is admittedly not a trivial undertaking, it is not unreasonable to assume that this would be a significantly faster process than waiting the approximately 32-63 years required for the amount of data available in those languages to reach current English-language levels. Furthermore, broad-coverage grammars do exist for some lower-resource languages: e.g. Spanish (Zamaraeva, Allegue, and Gómez-Rodríguez, 2024), French (Emirkanian, Da Sylva, and Bouchard, 1996), German (Cramer and Zhang, 2009), and Japanese (Mitsuishi, Torisawa, and Tsujii, 1998).

7.2 Improvements to GFoLDS

Section 7.2.1 proposes a solution to the limitations of GFoLDS' positional encoding architecture uncovered in Chapter 6, and discusses how the proposed approach presents a likely avenue to rectify the model's weaknesses that were exposed in that chapter. In Section 7.2.2, I

[1]https://gsll.unc.edu/learning-resources/german-grammar-videos/
[2]https://ceus.indiana.edu/about/languages/turkish.html
[3]https://liberalarts.utexas.edu/languages/korean.html
[4]https://commoncrawl.github.io/cc-crawl-statistics/plots/languages

outline potential modifications to the model's embedding layer that permit the inclusion of out-of-vocabulary (OOV) items and CARGs in its input graphs.

Section 7.2.3 suggests a possible adaptation of the embedding module that would allow the input of sequences of multiple sentences to the GFoLDS model, and discusses a potential new pretraining objective that would be licensed by such a capability (Section 7.2.3.2). Finally, in Section 7.2.4, I discuss the potential for non-Euclidean—namely, hyperbolic—embeddings to improve the model's performance, in light of the hierarchical structure induced by GFoLDS' DMRS-derived input graphs.

7.2.1 Positional Encoding Module

The double-negation cancellation experiments—along with Theorem 2—of Chapter 6 revealed a critical weakness of the GFoLDS positional encoding network: its embeddings do not encode nodes' global positions in the graph structure. Recall from the discussion of the GFoLDS architecture in Chapter 4 that the positional encoding module consists of an embedding layer that encodes the labels and DMRS features of each node, followed by a message-passing neural network (MPNN; i.e. the SWA block).

7.2.1.1 Background

Morris et al. (2019) show that standard MPNNs such as the GFoLDS SWA block are strictly as expressive as the 1-Weisfeiler-Leman graph isomorphism test (1-WL; Leman and Weisfeiler, 1968), which fails to distinguish a wide range of isomorphic (sub-)graphs (Arvind et al., 2015). However, Rampášek et al. (2022) demonstrate that MPNNs augmented with *positional* and *structural* encodings can surpass the expressive power of 1-WL. The authors divide these two encoding types into three categories, for a total of six subcategories: local, global, and relative positional/structural encodings.

Local positional encodings are those which represent each node's position and role relative to other nodes within its local k-hop neighborhood, while local structural encodings imbue

each node with a representation of its local neighborhood. The current SWA architecture clearly constructs local structural encodings: each node's SWA embedding is an aggregation of its k-hop neighborhood, and is sensitive to the labels of both the nodes and edges within that neighborhood.

On the other hand, *relative* positional encodings represent the distance between pairs of nodes across the graph, where the similarity between two nodes' local positional encodings decreases as the graph distance—i.e. the length of the shortest path—between them increases. Relative structural encodings permit the measurement of the difference between the local neighborhoods of any two nodes, and can be achieved with sufficiently powerful local structural encodings.

Global positional encodings represent each node's position within the entire graph: Lim et al. (2023) show that global positional encodings can be achieved via a combination of relative positional and local structural encodings. Arguably, local positional encodings are a consequence of global positional encodings, as a representation of a node's position within the entire graph structure leads to a representation of that node's position within its local neighborhood. Global structural encodings are features assigned to the graph as a whole, and are not relevant to the current discussion.

Therefore, to yield the full suite of structural and positional encodings outlined in Rampášek et al. (2022), it suffices to integrate relative positional encodings into GFoLDS' positional encoding module: GFoLDS already has local structural encodings; relative structural encodings can be achieved with sufficiently powerful local structural encodings; local positional encodings are a consequence of global positional encodings; and global positional encodings can be achieved with relative positional and local structural encodings.

Many distance-based approaches to graph embeddings employ *spectral* embedding methods: techniques that derive node embeddings from the eigendecomposition of the graph's Laplacian matrix (Chami et al., 2022). However, such approaches cannot readily generalize to unseen graphs, as the spectral decomposition depends on the choice of eigenbasis and sign of the

individual eigenvectors, and the dimension of the Laplacian's eigenvectors depends on the number of nodes in the graph.

Mialon et al. (2021) construct a spectral graph transformer that avoids this problem through the use of *diffusion kernels* (Kondor and Lafferty, 2002): a generalization of the classical heat kernel to graph domains. Concretely, the diffusion kernel $K_G^{(\beta)}(x, y)$ can be viewed as the proportion of thermal energy that propagates from x to y along the edges of the graph G within a fixed amount of time β. While the diffusion kernel *relies* on the eigendecomposition of the graph's Laplacian, its output $K_G^{(\beta)}(x, y)$ is simply a non-negative scalar.

Note that the diffusion kernel is only defined on undirected graphs, while the DMRS-derived graphs described in Chapter 4 are directed and edge-labeled. However, the SWA mechanism (also described in Chapter 4) passes messages along both directions of each directed edge: it is therefore reasonable to use an undirected notion of distance with respect to GFoLDS' input graphs. Letting $U(G)$ denote the underlying undirected graph[5] of a directed graph G, we may define a diffusion kernel $K_G^{(\beta)}(-,-)$ on a GFoLDS input graph as in Equation 7.1a, where $e^{(-)}$ is the matrix exponential, $\beta > 0$ is a tunable hyperparameter, and $H^{(G)} \in \mathbb{R}^{N \times N}$ (Equation 7.1b) is the negative of the Laplacian matrix $L^{(G)}$ of G ($H^{(G)} = -1 \cdot L^{(G)}$). The term $d_{U(G)}(i)$ denotes the degree of the i^{th} node in $U(G)$: $d_{U(G)}(i) = |\{j \mid \{i, j\} \in E(U(G))\}|$.

$$K_G^{(\beta)}(x_i, x_j) = \left(e^{\beta H^{(G)}}\right)_{i,j} \tag{7.1a}$$

$$H_{i,j}^{(G)} = \begin{cases} 1 & \text{if } \{i, j\} \in E(U(G)) \\ -d_{U(G)}(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{7.1b}$$

[5]Concretely, $U(G) = (V, U(E))$ is derived by converting each directed edge of $G = (V, E)$ into an undirected edge: $\{x, y\} \in U(E) \leftrightarrow (x \to y \in E \vee y \to x \in E)$.

7.2.1.2 Proposed Approach

Mialon et al.'s (2021) spectral graph transformer uses the diffusion kernel to construct a second attention mechanism in which the attention $\alpha_{i,j}$ between the nodes x_i and x_j is given by $K_G^{(\beta)}(x_i, x_j)$. Rather than injecting the graph structure directly into each encoder layer—which forces the encoder to adhere to the graph structure—I propose incorporating a diffusion-kernel-based message-passing block into GFoLDS' positional encoding network. The basic mechanism is similar to that of Mialon et al. (2021): message passing between all nodes in the graph, weighted by the diffusion kernel.

For a given DMRS-derived graph G and node $x \in V(G)$, its proposed diffusion-kernel-weighed positional encoding $P(x, G)$ is defined in Equation 7.2, where f and g are feed-forward layers (possibly linear layers, or even the identity function).

$$P(x, G) = g\left(\sum_{y \in V(G)} K_G^{(\beta)}(x, y) \cdot f(y)\right) \tag{7.2}$$

There are then three obvious ways to incorporate P into the pre-existing positional encoding network's architecture (see Chapter 4), which are laid out in Equation 7.3, where $E(X, G)$ denotes the node label/feature embedding layer.

$$EMB(X, G) = E(X, G) + SWA(P(E(X, G), G), G) \tag{7.3a}$$

$$EMB(X, G) = E(X, G) + P(SWA(E(X, G), G), G) \tag{7.3b}$$

$$EMB(X, G) = E(X, G) + SWA(E(X, G), G) + P(E(X, G), G) \tag{7.3c}$$

Under the first approach (Equation 7.3a), the output of $P(X, G)$ is fed into the SWA block. However, this configuration is likely to be problematic: the low expressivity of MPNNs such as the GFoLDS model's SWA block could cause the irreparable loss of information from the more expressive positional encoding architecture P (Rampášek et al., 2022).

This leaves the approach laid out in Equation 7.3b, in which the output of the SWA block is fed into P, and that of 7.3c, where the outputs of both modules are simply summed together. It is unclear which configuration would be more effective, and the more performant method will likely have to be determined empirically: based on my experience pretraining the GFoLDS model in Chapter 4, I estimate that this would require four to six pretraining runs in order to compare the two approaches with optimal hyperparameter configurations—well within the realm of feasibility.

7.2.1.3 Applications

Regardless, the use of this diffusion-kernel-based positional encoding is likely to overcome the limitations to the GFoLDS model uncovered in Chapter 6. Recall that in the sentence-membership classification experiment of Chapter 6, I joined two graphs G_1, G_2 with an *if_ x_ then* token: GFoLDS' task was to classify each node—aside from *if_ x_ then*—as belonging to S_1 (i.e. G_1) or S_2 (i.e. G_2).

Let x_1 denote the node that is connected to *if_ x_ then* via an edge $\textit{if\_ x\_ then} \xrightarrow{ARG2} x_1$ (i.e. $x_1 = \textit{htop}(G_1)$). Then membership of a given node x in S_1 is equivalent to the right-hand expression in Equation 7.4.

$$x \in S_1 \leftrightarrow K_G^{(\beta)}(x, x_1) > K_G^{(\beta)}(x, \textit{if\_ x\_ then}) \tag{7.4}$$

In words: x is closer to x_1 than it is to the connecting *if_ x_ then* node. The SWA block permits the detection of x_1, while P allows the model to determine the relative distance between nodes. Given that this is a binary classification task ($x \in S_2 \leftrightarrow x \notin S_1$), the ability to detect membership in S_1 suffices to achieve the S_1/S_2 membership detection objective.

Note that in the mod-2 counting experiment of Chapter 6, the input graphs were simply length-n linear sequences. In this case, the diffusion kernel is purely a function of the distance between nodes (Kondor and Lafferty, 2002). Specifically, for the i^{th} node in the sequence x_i, the kernel with respect to the initial node x_0 is a sinusoidal function (Equation 7.5).

$$K_G^{(\beta)}(x_0, x_i) = \frac{1}{n}\sum_{j=0}^{n-1} e^{-2\beta\left(1-cos\left(\frac{2\pi j}{n}\right)\right)} \cdot cos\left(\frac{-2\pi ij}{n}\right) \tag{7.5}$$

In this way, spectrum-based embedding methods can be viewed as a generalization to graphs of the sinusoidal positional embeddings introduced by Vaswani et al. (2017) for superficial transformers (Dwivedi et al., 2023). Critically, this ensures that each node in the sequence will receive a unique embedding, thereby facilitating the mod-2 counting task: as discussed in Chapter 6, the ability to perform mod-2 counting is critical for the double-negation cancellation task introduced in Chapter 2.

7.2.2 Incorporating OOV Terms and CARGS

In Chapter 4, I simply masked out-of-vocabulary (OOV) terms and CARGs from GFoLDS's DMRS-derived graphs, omitting these predicates and constant arguments from the model's input. The negative effects of the limitation introduced by this design choice were most apparent during the RELPRON experiment of Chapter 5, in which I had to create a new subset of the dataset that excluded examples containing OOV items and/or CARGs, preventing a direct comparison between GFoLDS and the FDS (Emerson, 2018) and FDSAS (Lo et al., 2023) models on this task.

Note that the OOV items and CARGs both present the same problem and, therefore, have the same solution: in its current implementation (i.e. as defined in Chapter 4) the model's token embedding layer $\mathcal{E}_T$ simply takes each node's label (i.e. predicate) and looks up the corresponding embedding (see Chapter 4 and Equation 7.7a). These items are not in the embedding layer's vocabulary, and so cannot be assigned embeddings.

To enable the inclusion of CARGs and OOV items in GFoLDS' input graphs, I propose modifying both the embedding layer and token-label prediction head of the model. Specifically, I intend to replace $\mathcal{E}_T$ with a (very) small, character-level encoder transformer (c.f. Yin et al., 2020, whose GNN model produces image node embeddings via CNN). Each predicate label

P will then be tokenized into its constituent characters $P_1, \ldots, P_n$, resulting in a small, fixed embedding layer vocabulary for the transformer $\mathcal{E}_T$: the 95 ASCII characters. The output of $\mathcal{E}_T$ will then be generated by mean pooling over the token (i.e. character) embeddings in P.

The remainder of the model will proceed as defined in Chapter 4, with the exception of the token-label prediction head. As incorporating OOV items and CARGs results in an unbounded set of node labels, the current pretraining objective—cross-entropy loss with respect to the predicted node label—is not viable. Therefore, I propose replacing the token-label prediction head with a small, character-level *decoder* transformer: for each node n_i with corresponding hidden state $\vec{x_i}$, the first token in the decoder sequence will be (downsampled) $\vec{x_i}$ (c.f. Tang et al., 2023). The decoder will be tasked with predicting the sequence of characters corresponding to the label of n_i, conditioned on $\vec{x_i}$.

Note that this approach does not necessarily preclude the masked-node modeling (MNM) pretraining objective laid out in Chapter 4. For each node n selected to be masked, the model's input will be replaced with the [MASK] token, as before. The decoder head's prediction target will then be the sequence $[\text{MASK}]_1, \ldots, [\text{MASK}]_k$, where k is the length in characters of the label $\ell(n)$ of n: loss for that node will then be calculated as the mean cross-entropy for the distribution over $[\text{MASK}]_i$ with respect to the character $\ell(n)_i$, for all $1 \leq i \leq k$.

7.2.3 Multiple-Sentence Model

As discussed in Chapter 5, one critical weakness of the GFoLDS model in its current form (i.e. as defined in Chapter 4) is its inability to process multiple sentences at once: the impact of this limitation is acutely displayed in Chapter 5, in which each premise/hypothesis pair in the SNLI dataset (Bowman et al., 2015) must be conjoined with an *if_ x_ then* token in order to yield a single graph that can be processed by the model. But this task-specific remedy cannot be applied to other multiple-sentence NLP benchmarks such as question-answering (e.g. Rajpurkar et al., 2016), chain-of-thought reasoning (e.g. Wei et al., 2022), etc. If GFoLDS is

to be applicable to the same range of tasks as superficial language models, then it is necessary to overcome this limitation.

Furthermore, the ability to input more than one sentence at a time has the potential to enrich the context available to the model: when reading a body of text, the ability to recall (or re-read) earlier sentences in the text is critical to disambiguating and contextualizing the current sentence. For example, the current GFoLDS model has no mechanism to allow for the identification of pro-forms with their respective anaphors, beyond anaphora occurring within a single sentence. Assuming that the model can learn to resolve anaphora with a reasonable degree of accuracy, feeding it multiple sentences at the same time would allow it to more effectively learn logical co-occurrence: identifying a given pronoun (for example) with the predicate P that describes the main referent of the entity to which it refers effectively increases the amount of contexts (from the perspective of the model) in which P occurs.

To illustrate, consider the following example (duplicated from Chapter 3): "*all alligators*$_i$ *are ...* *they*$_i$ *run frequently*", where the spans *all alligators* and *they* are located in different sentences. In its current form, GFoLDS has no way to link *they* to *all alligators*, and so the second half of this example will only serve as an instance of co-occurrence between *they* and *run frequently*. If the multiple-sentence variant of GFoLDS can learn to resolve anaphora, then the above passage would effectively serve as an instance of co-occurrence between *all alligators* and *run frequently*.

7.2.3.1 Proposed Approach

To modify the architecture of the GFoLDS model to permit it to take multiple sentences as input, we need look no further than BERT (Devlin et al., 2019). For tasks such as NLI, the output of the BERT embedding layer $E(t, i)$ for each token t at position i in the input sequence is defined as in Equation 7.6, where E_{tok} denotes the token embedding layer and E_{pos} the positional embedding layer.

$$E(t,i) = E_{tok}(t) + E_{pos}(i) + E_{sent}(i) \tag{7.6}$$

The term E_{sent} denotes the *sentence type* embedding layer, and $E_{sent}(i)$ returns $\vec{s_1}$—the embedding for tokens belonging to the first sentence—if the token at position i is in the first sentence (e.g. the premise in an NLI task), and $\vec{s_2}$ otherwise. This is to say that an embedding identifying the sentence to which the token belongs is added to each token embedding in the input sequence.

It is fairly straightforward to adapt this approach to GFoLDS: simply add a sentence-level embedding layer $\mathcal{E}_S$ to the model's embedding layer, so that the current embedding architecture (defined in Chapter 4; duplicated in Equation 7.7a) is modified to that in Equation 7.7b.

$$\vec{e_i} = E(n_i, G) = \mathcal{E}_T(n_i) + Norm\left(\sum_{\phi \in F(n_i,G)} \mathcal{E}_F(\phi)\right) \tag{7.7a}$$

$$\vec{e_{k,i}} = E(n_{k,i}, G_k) = \mathcal{E}_T(n_{k,i}) + Norm\left(\sum_{\phi \in F(n_{k,i},G_k)} \mathcal{E}_F(\phi)\right) + \mathcal{E}_S(k) \tag{7.7b}$$

Where $n_{k,i}$ and $\vec{e_{k,i}}$ denote the i^{th} node in the k^{th} input graph and its embedding, respectively.

Note that BERT's sentence-level embedding layer only has two possible values: $\vec{s_1}$ and $\vec{s_2}$. This is because each "BERT sentence" can correspond to multiple linguistic sentences—in the BERT terminology, a "sentence" simply denotes a sequence of tokens. Although the sentences encoded by the term $\mathcal{E}_S(k)$ in Equation 7.7b correspond to actual linguistic sentences (due to the ACE/ERG DMRS parser; see Chapter 4), it is fairly trivial to extend BERT's two-sentence embedding method to any number of sentences: simply increase the number of possible inputs to $\mathcal{E}_S$, so that $\mathcal{E}_S$ is effectively the BERT *positional* embedding layer (E_{pos} in Equation 7.6) applied at the sentence level.

With a traditional positional embedding layer such as BERT's, it would be necessary to choose a fixed number of embeddings u_s that $\mathcal{E}_S$ encodes, thereby setting an upper bound on the amount of sentences that the model can process in a single input sequence—although u_s could be set arbitrarily high, effectively limited only be hardware constraints. However, with more advanced positional encoding techniques—such as rotary embeddings (Su et al., 2024)—no such limit u_s is necessary. Regardless, due to hardware constraints, a fixed upper bound u_t on the number of tokens that can be contained within in a given input sequence must be set, as a sentence can be of any arbitrary length.

Let $D = \{D_1, \ldots, D_n\}$ denote the model's training dataset, consisting of n documents D_i: for example, Wikipedia articles—what is critical is that the sentences within each document are intended to follow one another. At training time, given a document $D_i = \{S_{i,1}, \ldots, S_{i,m}\} \in D$, select k sentences from D_i, subject to the constraints defined in Equation 7.8.

$$k = max\{\hat{k} \mid \hat{k} \leq m \wedge \sum_{j=1}^{\hat{k}} |V(G_{i,j})| \leq u_t\} \tag{7.8}$$

Then set $D_i \leftarrow \{S_{i,k+1}, \ldots, S_{i,m}\}$, and repeat the process laid out above until D_i is exhausted.

As this multi-sentence pretraining procedure is effectively the same as the single-sentence procedure laid out in Chapter 4—the difference being that the batch elements (i.e. sentences) can interact with one another—I expect the multi-sentence pretraining time to be roughly equivalent to that of single-sentence pretraining.

7.2.3.2 Entailment Prediction Objective

As discussed in Chapter 4, the GFoLDS model's pre-training objective—masked node modeling—is based on the masked language modeling (MLM) pre-training objective of encoder LMs such as BERT (Devlin et al., 2019). In addition to MLM, however, BERT is also pre-trained with the *next sentence prediction* (NSP) objective: during pre-training, BERT is fed pairs of "sentences" (contiguous sequences of text) S_1, S_2 in the form of the sequence

[CLS]S_1[SEP]S_2, where the [SEP] token separates S_1 and S_2, and the hidden state of the [CLS] (classification) token is used for the NSP task. In half of the input examples, S_2 immediately follows S_1 in a given document in the training corpus, while in the other half, S_2 is a randomly-selected sequence of text: the model is equipped with a small, feed-forward binary classifier that predicts from the resulting [CLS] embedding whether S_2 follows S_1 (hence "next sentence prediction"). Note that BERT is pre-trained with the NSP and MLM objectives *simultaneously*, and the loss values from both tasks are simply summed together before backpropagation.

The goal of this NSP objective is "to train a model that understands sentence relationships" (Devlin et al., 2019). Given the logical-reasoning-oriented nature of GFoLDS, it is sensible to devise an NSP-like pre-training objective that likewise has a logical-reasoning orientation. Therefore, I propose to incorporate *entailment*—rather than next-sentence—prediction as the secondary pre-training task for the GFoLDS model. Under this approach, the model will be fed pairs of (graph representations of) sentences G_1, G_2, where G_1 entails G_2 in half of the examples, and G_2 is unrelated to G_1 in the other half.

Unfortunately, it is significantly more difficult to obtain positive examples for entailment prediction than for NSP: if it were possible for a given G_1 to search through the pre-training dataset and reliably find some G_2 entailed by G_1, then the NLI problem would already be solved. However, while *finding* positive examples of entailment in the training data is exceedingly problematic (if not impossible), it is in fact plausible to *generate* positive examples using the DMRS graph description language developed by Copestake et al. (2016). This graph description language effects pattern matching and replacement for DMRS structures: for example, we can write a rule along the lines of $[X \text{ give } Y \text{ to } Z] \Rightarrow [Z \text{ get } Y \text{ from } X]$, which transforms the DMRS representation of a sentence such as "*Kim gave a book to Sandy*" to one corresponding to the sentence "*Sandy got a book from Kim*" (see Copestake et al., 2016, Figure 5). This tool should permit automatic generation of entailment examples using re-write rules to replace words with hypernyms, structures with converse/relational antonyms

(e.g. *give*/*get*), etc.

Obtaining negative examples is straightforward: simply select some G_2 at random from the data. While there is a chance that this approach will *occasionally* select a positive example, the likelihood that there happens to be an entailment relation between two randomly-selected sentences is presumably exceedingly low.

7.2.4 Hyperbolic Embeddings

Hyperbolic embeddings refer to vector representations in hyperbolic space: a manifold of constant negative curvature (Tifrea, Becigneul, and Ganea, 2018). Hyperbolic embeddings are generally modeled as points in n-dimensional manifolds whose underlying sets are subsets of n-dimensional Euclidean space, such as the Poincaré ball $\mathcal{B}^n \subset \mathbb{R}^n$, whose underlying set consists of all points within a distance of one from the origin (Riemann, 1854)—this use of underlying Euclidean space facilitates the adaptation of traditional machine learning techniques to hyperbolic applications.

The key difference between hyperbolic and Euclidean space lies in the distance metrics employed by the respective geometries: in $\mathcal{B}^n$, the distance between two points increases at an exponential rate relative to their Euclidean distance as the points grow further from the origin. Hyperbolic embeddings are useful for modeling hierarchical structures such as trees due to this property, which permits encoding fine-grained differences between sister nodes in a tree (for example), while still allowing those nodes to be close to their mother node in the embedding space (see Figure 7.1). Because of their utility for modeling hierarchical structures, Nickel and Kiela (2017) employ hyperbolic representations to embed the WordNet (Miller, 1995) noun hierarchy, and demonstrate that hyperbolic embeddings outperform their Euclidean counterparts in applications such as modeling the transitive closure of the hyponymy relation.

For this reason, it may be beneficial to incorporate hyperbolic embeddings into the GFoLDS model. As a consequence of the rule-based nature of the MRS parser, different

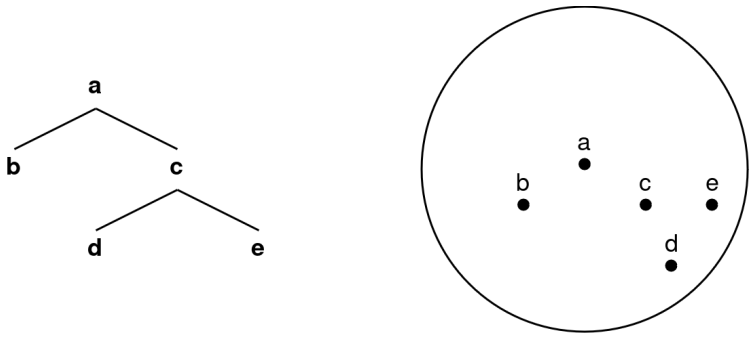


Figure 7.1: An example of a tree (left) and hyperbolic embeddings of its nodes in the two-dimensional Poincaré ball $\mathcal{B}^2$ (right). Note that despite all three points being equidistant by the Euclidean metric, $\vec{d}$ and $\vec{e}$ are further apart from one another than either one is to $\vec{c}$ in $\mathcal{B}^2$, as $\vec{d}$ and $\vec{e}$ are further from the origin.

parts of speech occur in distinct contexts in the derived graph representations: nouns have different features than verbs, quantifier nodes do not have features, etc. In other words, these MRS-derived graphs are implicitly heterogeneous (Zhang et al., 2019): their nodes can be sorted into separate, disjoint types—in this case, based on category/part-of-speech. The masked node modeling pre-training objective will cause any two nodes that occur in the same broad contexts—i.e. belong to the same category—to be embedded closer to one another, while more minute differences between the contexts in which any two nodes of the same category appear—for example, different feature *values*, different arguments, etc.—will result in finer-grained distinctions between the nodes' vector representations. In a Euclidean setting, this will force any pair of verbs to be within a cosine distance of (for example) 0.8 of one another, such that cosine distances between verbs range between 0.8 (more dissimilar) and 1.0 (more similar).

Such minute differences likely present difficulties to the model in learning to representationally distinguish between nodes of the same category. However, it is clear that the heterogeneous nature of the derived graph representations induces a hierarchical structure on the vocabulary: the above-mentioned categories (nouns, verbs, adjectives, quantifiers, etc.) lay at the top, while increasingly finer-grained distinctions between the respective contexts in which nodes of the same type occur correspond to a hierarchy of nested, latent sub-categories.

Hyperbolic space could provide a natural setting in which GFoLDS can represent this tree-like hierarchy.

Unfortunately, converting a model from Euclidean to hyperbolic representations is not a straightforward process: the Euclidean tensors outputted by each layer of the model must be mapped into hyperbolic space—necessitating a complete re-working of the architecture—and hyperbolic neural networks require unique loss functions and algorithms for computing their gradients (i.e. Riemannian Optimization; Ganea, Bécigneul, and Hofmann, 2018).

7.3 The Next Step: Graph-Generative Models

The GFoLDS model described within this dissertation is an encoder model, analogous to superficial models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021), ELECTRA (Clark et al., 2021), and so on. However, current, SoTA LLMs have almost exclusively trended towards generative (i.e. decoder or encoder-decoder) models in recent years: BART (Lewis et al., 2020), T5 (Raffel et al., 2020), Llama-2/3 (Touvron et al., 2023; Dubey et al., 2024), GPT-3/4 (Brown et al., 2020; OpenAI, 2023), etc. are all generative LMs. The text-to-text nature of these models lends the flexibility to accomplish the classification tasks that encoder models are capable of—for example, by simply generating the predicted class label—while also allowing the capacity to perform more advanced tasks such as chain-of-thought reasoning (e.g. Wei et al., 2022), dialogue generation (e.g. Zhang et al., 2018), summarization (e.g. Nallapati et al., 2016), and even human evaluation tasks such as the SAT, GRE, and LSAT (OpenAI, 2023).

In Chapter 4, I argued that a graph-to-graph generative model was decidedly outside of the scope of this work. That being said, the overarching goal of the research program outlined in this dissertation is to yield a language model over logical forms that can compete with its superficial counterparts at scale: such a model must have generative capabilities in order to be able to perform the same range of tasks as current SoTA LLMs.

To that end, I dedicate this section to an outline of a proposed generative architecture, built largely around the GFoLDS model described in Chapter 4, with some (or all) of the modifications proposed in Section 7.2. I envision this process consisting of three components: a graph-to-graph model M, a (potentially neural; e.g. Buys and Blunsom, 2017; Lin, Liu, and Shang, 2022) DMRS parser P, and a DMRS-to-text model P^{-1} (e.g. Hajdik et al., 2019; Guo et al., 2019; Zhang et al., 2020b; Wang, Wan, and Jin, 2020; Wang, Wan, and Yao, 2020, etc.).

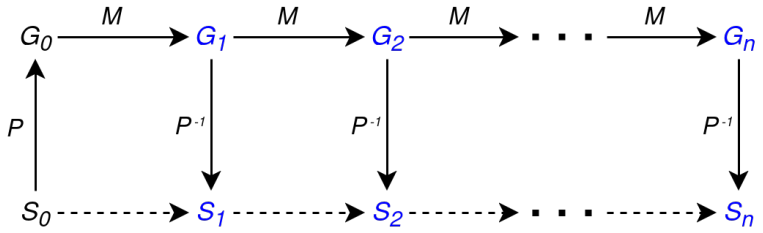


Figure 7.2: High-level illustration of the proposed graph-generative pipeline. Black nodes indicate the input sentence (and corresponding graph representation), while those in blue denote model-generated graphs and sentences.

The overall pipeline is illustrated in Figure 7.2. The idea is that an input sentence or sequence of sentences S_0 is parsed with P to yield its corresponding DMRS graph G_0. The graph-to-graph model M will then generate a graph G_1, conditioned on G_0: G_0 and G_1 can then be autoregressively fed back to M to yield G_2, and so on. This ultimately results in a sequence of graphs $G_0, \ldots, G_n$. For each non-input graph G_i $(1 \leq i \leq n)$, we can then use the graph-to-text model P^{-1} to generate a corresponding natural language sentence S_i, yielding a sequence of sentences $S_1, \ldots, S_n$ generated from the input S_0.

In Section 7.2.3, I described modifications that would allow GFoLDS to take multiple graphs as input, which is necessary to condition the generation of G_i on $G_0, \ldots, G_{i-1}$: the only missing piece of the puzzle is the GFoLDS-derived, graph-generative model M.

7.3.1 Background

The idea of neural graph generation is not a novel concept, although the majority of the work in this area is in the domain of molecule graphs (Zhu et al., 2022). These approaches can be broadly divided into two categories: *one-shot* and *sequential*. One-shot graph generation (e.g. Flam-Shepherd, Wu, and Aspuru-Guzik, 2020; Du et al., 2022) involves predicting the edges between a fixed set of input nodes. The key advantage to such approaches is that they do not introduce the potential inductive bias with respect to node ordering that is inherent in sequential generation methods. However, they require a predefined node set before generating edges.

Bresson and Laurent (2019) introduce a middle ground between one-shot and sequential generation: under their two-step approach, the model first generates a set of nodes in an autoregressive fashion in step one, then predicts the adjacency matrix—i.e. the edges between those nodes—in a single pass in step two. Although this method allows for a variable-size node set while avoiding node-order-related inductive bias, it is not suitable for our purposes. Consider, for example, a DMRS node set containing two nodes x_0, x_1 with the same label (e.g. *the_q*). As there are no edges between nodes in the first step, both x_0 and x_1 will recieve the same representation: namely, the embedding of their label. For any other node z, the edge predictions between x_0 and z will be identical to those between x_1 and z.

This indicates that a graph-to-graph generative language model requires sequential generation. As alluded to above, sequential graph generation (e.g. Liu et al., 2018) involves a back-and-forth, autoregressive process of predicting one or more nodes, then edges between them, then predicting additional nodes, and so on. It is not, however, strictly necessary to generate nodes and edges in separate steps: Bacciu, Micheli, and Podda (2020), Goyal, Jain, and Ranu (2020), and Bacciu and Podda (2021) employ *edge-based* generation. Under this approach, a node x_0 is selected, and edges $x_0 \leftarrow x_i$, $x_0 \rightarrow x_k$—along with the source/target nodes x_i/x_k—are autoregressively generated until local stopping conditions are met. The model then proceeds to the next unvisited node and repeats this edge/node generation

procedure, halting once global stopping conditions are reached.

7.3.2 Proposed Architecture

The connected nature of DMRS graphs lends itself to the edge-based sequential generation procedure desribed in Section 7.3.1: ignoring edge direction, any node in a DMRS graph can be reached by starting at any other node in the graph. As discussed above, I intend for this proposed graph-to-graph architecture to be built around the GFoLDS model described in this dissertation. In order to convert GFoLDS into a generative model, additional, smaller models that act on GFoLDS' hidden states are required: an edge direction model $\mathcal{M}_E$, an edge label model $\mathcal{M}_\ell$, and a family of feature value models $\mathcal{M}_F^{(-)}$.

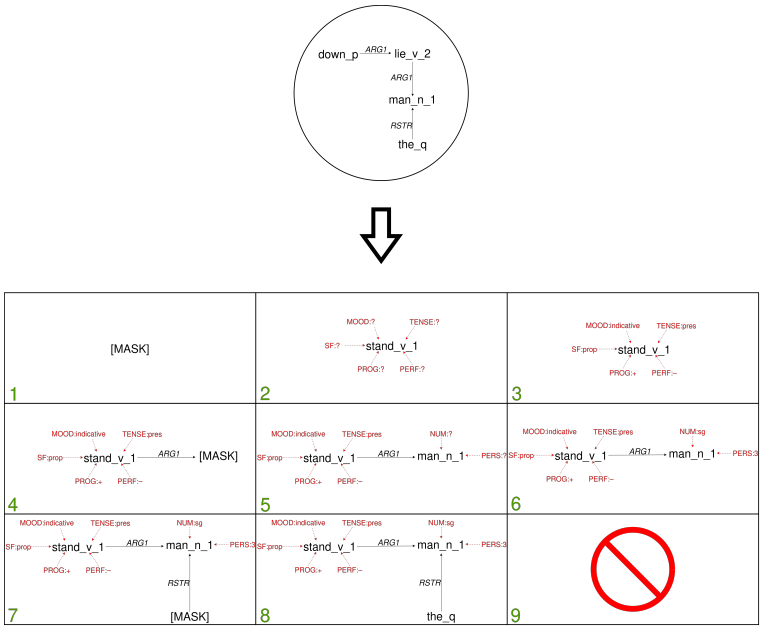


Figure 7.3: Illustration of the proposed graph-to-graph generation procedure: generating G_1 ("*the man is standing*"; bottom) from the input G_0 ("*the man is lying down*"; top)—for the sake of representational clarity, predicate features are omitted in G_0. The bottom image illustrates the step-by-step generation process for G_1: each step is numbered in green in the lower-left corner. A red, dashed arrow $\phi \dashrightarrow x$ indicates that ϕ is a feature of the node x.

As I view graph-to-graph generative models as the next major phase of the research program laid out in this dissertation, I will describe the generation process via a (somewhat lengthy) example, which is illustrated in Figure 7.3. Given an input graph G_0, G_1 begins with

a single node n_0, whose label is [MASK] (Step 1, Figure 7.3). The underlying GFoLDS model then predicts a node label ℓ ($\ell = \textit{stand\_v\_1}$ in Figure 7.3) for n_0, and we set $\textit{label}(n_0) \leftarrow \ell$ (Step 2, Figure 7.3).

The features[6] for a given DMRS node are entirely determined by its category (part-of-speech): all verbs have *MOOD*, *TENSE*, *SF*, *PROG*, and *PERF* features; all nouns have *NUM* and *PERS* features; etc. Therefore, based on the predicted category for the node label ℓ (categories are always encoded as a suffix in DMRS node labels), we know the set of features $\mathcal{F}_{n_0} = \{\phi_1, \ldots, \phi_k\}$ whose values must be predicted for n_0. For each $\phi \in \mathcal{F}_{n_0}$, we can then predict the value of ϕ for n_0 as $\textit{argmax}(\mathcal{M}_F^{(\phi)}(\vec{x_0}))$, where $\vec{x_0}$ is the last hidden state representation for n_0 generated by GFoLDS (Step 3, Figure 7.3). There are only ten features in DMRS (with 2-5 values each), necessitating ten feed-forward feature-value models $M_F^{(\phi)} \colon \mathbb{R}^{d_{model}} \rightarrow \textit{values}(\phi)$.

We then use the edge-direction model $\mathcal{M}_E \colon \mathbb{R}^{d_{model}} \rightarrow \{\leftarrow, \rightarrow, \emptyset\}$ to predict zero or more edges into/out of n_0 from the hidden state $\vec{x_0}$. If $\mathcal{M}_E(\vec{x_0}) = \rightarrow$ or $\mathcal{M}_E(\vec{x_0}) = \leftarrow$, we add an edge $e = n_0 \rightarrow n_1$ or $e = n_0 \leftarrow n_1$ (respectively), set the label of n_1 to [MASK], and add n_1 to the node stack σ. Letting A denote the set of edge labels ($|A| = 8$; see Chapter 4), we use the edge-label model $\mathcal{M}_\ell \colon \mathbb{R}^{2d_{model}} \rightarrow A$ to predict an edge label $\textit{label}(e) = \mathcal{M}_\ell(\vec{x_0}; \vec{x_1})$ (if $e = n_0 \rightarrow n_1$) or $\textit{label}(e) = \mathcal{M}_\ell(\vec{x_1}; \vec{x_0})$ (if $e = n_1 \rightarrow n_0$), from the concatenation of the hidden states for n_0 and n_1. In Step 4 of Figure 7.3, $\mathcal{M}_E(\vec{x_0}) = \rightarrow$ and $\mathcal{M}_\ell(\vec{x_0}; \vec{x_1}) = \textit{ARG1}$.

Once $\mathcal{M}_E(\vec{x_0}) = \emptyset$—which indicates that there are no more edges to generate for the current n_0—we pop a new node n_0 from the node stack σ, and repeat the process described above until σ is empty. The generated graph G_1 is then returned.

In Figure 7.3, $\mathcal{M}_E(\vec{x_0}) = \emptyset$ is predicted after Step 4, so we pop a new n_0 from σ, and predict its node label *man_n_1* (Step 5) and features (Step 6). In Step 7, $\mathcal{M}_E(\vec{x_0}) = \leftarrow$ and $\mathcal{M}_\ell(\vec{x_1}; \vec{x_0}) = \textit{RSTR}$. Then, $\mathcal{M}_E(\vec{x_0}) = \emptyset$, so we pop a new n_0 from σ and predict the label *the_q* (Step 8). At this point, $\mathcal{M}_E(\vec{x_0}) = \emptyset$ and σ is empty, so generation is terminated

[6]A detailed description of DMRS node features is presented in Chapter 4.

and G_1 is returned (Step 9).

Many generative LMs employ a *beam search* in generation (Sun et al., 2023), where the model generates k completions (beams) for a given input. At each time step (token) t, the top n tokens for each beam are selected, yielding nk beams, and the top k most-likely beams—in terms of the cumulative probabilities of all of their respective tokens—are retained for time step $t+1$; at the end of the beam search, only the most likely beam is returned.

Li et al. (2023) introduce a transformer that generates intermediate syntax trees in order to produce text, guided by a *structural* beam search: an adaptation of the standard beam search algorithm to non-linear, graph-like generation. This structural beam search could readily be adapted to the graph-generative procedure described in this section. In addition to (or separate from) the structural beam search, the pipeline proposed in this section could leverage prior linguistic knowledge in the form of rules—for example, quantifiers can only be the source node of *RSTR*-labeled edges, no node can have more than one outgoing edge with the same label, etc.—to preemptively eliminate structurally invalid predictions.

7.3.3 Training

The generative model's pretraining procedure will be implemented along the lines of the multiple-sentence procedure laid out in Section 7.2.3—i.e. over sequences of graphs $G_0, \ldots, G_n$. The last graph G_n in each sequence will then be treated as the generation target. In order to create generation-specific training data from G_n, I propose carrying out a breadth-first search over the underlying undirected graph $U(G_n)$ of G_n, starting at $htop(G_n)$ (see Section 5.3.1.1 of Chapter 5), in order to yield a canonical generation order. Letting $E(G_n)$ denote the edge set of G_n, this will result in a sequence of graphs $G_n^{(1)}, \ldots, G_n^{(|E(G_n)|)}$ such that $G_n^{(i)}$ is derived from $G_n^{(i-1)}$ $(2 \leq i \leq |E(G_n)|)$ by adding a single edge, along with its source or target node.

For each sequence of graphs $G_0, \ldots, G_n$ and each subgraph $G_n^{(i)}$ of G_n, the model will be trained to predict the label of the masked node in $G_n^{(i)}$ and the unique edge (and edge label) in $G_n^{(i+1)} - G_n^{(i)}$, conditioned on $G_0, \ldots, G_{n-1}, G_n^{(i)}$.

Note that the subgraphs $G_n^{(i)}$ of each G_n are in a one-to-one correspondence with the nodes of G_n: each subgraph $G_n^{(i)}$ adds a single source or target node—and an edge—to $G_n^{(i-1)}$. As discussed in Chapter 4, there were an average of ∼28.3 nodes per input graph in GFoLDS' pretraining data, so we can expect that pretraining the graph-generative model will take ∼28.3 times longer than GFoLDS. GFoLDS' total pretraining time was 102 hours and 24 minutes[7] (again, as discussed in Chapter 4), so we can expect a total pretraining time of ∼2898 hours (∼121 days) for the proposed graph-generative model on the same data.

While 121 days of pretraining time may seem prohibitive, note that the majority of the work in the graph-to-graph generation procedure proposed in Section 7.3.2 is carried out by the standard, encoder GFoLDS model. It therefore seems reasonable to assume that the encoder GFoLDS could be effectively "fine-tuned" for generation: a large part of the pretraining would be carried out as described in Chapter 4 (and Section 7.2.3), with a small subset of the pretraining data reserved to train the feed-forward classification heads ($\mathcal{M}_E$, $\mathcal{M}_\ell$, and $\mathcal{M}_F^{(-)}$) and tune GFoLDS. If, for example, we withhold 10% of the pretraining data for generation, we can expect a total pretraining time of 15 days and 22 hours.

7.4 Discussion

In this chapter, I discussed a wide range of potential applications and modifications to the GFoLDS model introduced in Chapter 4. In Section 7.1, I described how the current GFoLDS model (i.e. that introduced in Chapter 4) could be employed in low-resource settings, where a less data-intensive model such as GFoLDS could permit the more rapid development of higher-quality LMs.

I then proposed a series of architectural modifications to GFoLDS in Section 7.2, with the goal of addressing the limitations discussed in Chapters 4-6, and improving the model's performance in general: the use of diffusion kernels in the positional encoding module, in order to address the current model's inability to encode nodes' global positions (Section 7.2.1);

[7]With one NVIDIA A100 GPU.

a new embedding layer and prediction head architecture to enable the inclusion of CARGs and out-of-vocabulary items in the model's input graphs (Section 7.2.2); a sentence-level positional embedding layer to allow the model to take multiple sentences as input (Section 7.2.3); and the use of hyperbolic embeddings to better encode the hierarchical structure induced by DMRS representations (Section 7.2.4).

After the modifications proposed in Section 7.2, the GFoLDS model would still be an encoder graph transformer—i.e. a graph neural network coupled with a permutation-invariant encoder transformer (see Chapter 4 and Wu et al., 2021): the embedding layer would have an additional, linear sentence-level positional embedding component *across* graphs, but not *within* graphs (see Section 7.2.3). Although Section 7.2.2 proposes employing small transformer models to replace the embedding layer and prediction head, this would effectively make no difference from the GFoLDS model's perspective: for each input node n_i, GFoLDS would still receive a vector $\mathcal{E}_T(n_i)$ from the embedding layer, and pass a hidden state h_i to the prediction head. While the diffusion-kernel-based embeddings described in Section 7.2.1 would be an alteration of GFoLDS' internal architecture—akin to DeBERTa's (He et al., 2021) modification of BERT's positional embedding architecture—this proposed embedding module is to be built *around* the existing machinery of the positional encoding network, rather than replace any of its components.

This is to say that the result of the modifications discussed in Section 7.2 would be a further iteration of the model introduced in Chapter 4, rather than a radical departure from that architecture.

Finally, in Section 7.3, I outlined the next major step in the research program of language modeling over logical forms: graph-to-graph, generative language models: I discussed a potential pipeline for graph-to-graph generation, a step-by-step example of the proposed generation architecture, and the planned pretraining procedure of this model (and the computational feasibility thereof). The successful realization of such an architecture represents a fundamental goal in the development of language models over logical forms, as it would

allow their application to a much wider range of tasks, thereby bridging the current gap in utility between these models and their superficial counterparts.

Chapter 8

Conclusion

In this dissertation, I made the case for language models over logical forms by both theoretical and empirical means. With the GFoLDS model motivated in Chapters 2-3 and introduced in Chapter 4, this dissertation accomplished the two major goals set in Section 1.2 of Chapter 1.

The first objective was to demonstrate the validity of the Accelerated Learning Hypothesis (ALH): specifically, I showed that the aspects of linguistic knowledge incorporated into linguistically-informed LMs obviates the need to learn elementary linguistic phenomena, allowing them to immediately begin learning more complex patterns (see Chapter 6), which in turn leads to linguistically-informed LMs' ability to learn from less data than their superficial counterparts (see Chapter 5). The successful achievement of this objective is of importance to the field of Language AI, as the ALH—along with the empirical evidence towards its validity presented in this dissertation—indicates that the use of language models over logical forms provides the means to overcome the impeding shortage of LLM training data—and subsequent slowing of the rate of improvement of LLMs—that is predicted to occur in the near future (Villalobos et al., 2024; as discussed in Chapter 1).

Furthermore, the evidence in support of the ALH that I presented in this dissertation provides meaningful insight into the reasons behind the performance gains exhibited by linguistically-informed and -augmented LMs (e.g. Xu et al., 2021; Sachan et al., 2021; Zhou

et al., 2020; Zhang et al., 2020c; Wu, Peng, and Smith, 2021; Prange, Schneider, and Kong, 2022, etc.; see the discussion in Chapters 1 and 4), which itself is an important finding in the fields of Linguistics and Machine Learning.

The accomplishment of second objective of this dissertation—a proof-of-concept of the viability of language models over logical forms—paved the way for the practical implementation of such models, and introduced the research program of language modeling over logical forms. In Chapter 1, I divided this goal into two subordinate objectives: demonstrations of the (i) *feasibility* and (ii) *utility* (i.e. downstream applicability) of language models over logical forms. While the GFoLDS model definitively demonstrated the feasibility of language models over logical forms, a demonstration of some aspects of the utility of such models was left to future work. Specifically, GFoLDS' problematic positional encoding module (see Chapter 6), and its inability to take multiple sentences as input and to incorporate CARGs and out-of-vocabulary items (see Chapters 4-5), limit its downstream applicability. However, I suggested feasible modifications to the model's architecture that can plausibly overcome these limitations (see Chapter 7), as discussed in Chapter 1.

8.1 Findings and Contributions

In this section, I summarize the major findings and contributions of each of the main chapters (i.e. Chapters 2-7) of this dissertation.

Chapter 2 provided an experimental paradigm—which has further applications outside of the scope of this dissertation—for evaluating language models' logical reasoning abilities. In this chapter, I found that superficial NLI models learn to treat the external negation prefix "*it is not true that*" as a distractor when initially fine-tuned on common NLI datasets; that DeBERTa (He et al., 2021) and BART (Lewis et al., 2020) models are incapable of learning to inductively generalize the law of the excluded middle, despite extensive fine-tuning; and that the RoBERTa models (Liu et al., 2019) that did manage to grasp the function of the

prefix "*it is not true that*" failed to generalize this pattern to the highly similar prefix "*it is false that*". In addition to motivating the use of language models over logical forms by exposing a major weakness of superficial NLI models, the findings of Chapter 2 provided further support to existing work in the literature (e.g. McCoy, Pavlick, and Linzen, 2019; Chien and Kalita, 2020; Richardson et al., 2020; Niven and Kao, 2019; Naik et al., 2018; Yuan et al., 2023, etc.) indicating that language models leverage shallow heuristics to achieve their remarkable performance on logical reasoning tasks.

In Chapter 3, I introduced the non-neural FoLDS model, a prototype language model over logical forms. FoLDS demonstrated the feasibility of language modeling over logical forms, and outperformed almost all competing superficial approaches on the McRae et al. (2005) property inference database, thereby providing empirical evidence in support of the Accelerated Learning Hypothesis (ALH). Chapter 3 also included further arguments in support of the ALH, positing that the use of logical form inputs yields a syntactic de-noising effect, so that (for example) an active sentence and its passive counterpart are equivalence-classed.

In Chapter 4, I used the limitations of the FoLDS model—namely, its static, complex-valued count-vector embeddings—and an analysis of the existing literature on the use of graph neural networks in NLP to motivate the graph-transformer-based (Wu et al., 2021) GFoLDS model. I then pretrained GFoLDS on automatically-parsed graph representations of logical forms, thereby demonstrating the viability of this model.

Chapter 5 focused on a demonstration of the utility of GFoLDS: I evaluated the model against BERT—both the original models and comparison variants pretrained on the same amount of data as GFoLDS—on the RELPRON (Rimell et al., 2016), SNLI (Bowman et al., 2015), MegaVeridicality V2.1 (White et al., 2018), and McRae et al. (2005) benchmarks. GFoLDS' applicability to these tasks—in particular, the SNLI benchmark—displayed the vastly improved utility of this model over that of the FoLDS model introduced in Chapter 3. As GFoLDS massively outperformed the BERT comparison models on all four evaluation tasks, the results of this chapter additionally provided strong evidence towards the validity of

the ALH. This in turn constitutes strong evidence in support of the use of language models over logical forms as a plausible avenue for continuing the improvement of language models at a more sustainable rate of data consumption than superficial LLMs.

In Chapter 6, I evaluated the scalability of GFoLDS, and found that this model is likely to scale in terms of parameter count and pretraining data. This result is a step forward in the research program of language modeling over logical forms, as it suggests that such models have the potential to compete with superficial LMs at scale. Furthermore, I provided direct empirical evidence towards the validity of the ALH in Chapter 6, demonstrating that GFoLDS maintains near-peak performance on two elementary probing tasks throughout pretraining, while the BERT comparison models do not even begin to improve on the elementary tasks until halfway through the first pretraining epoch. Critically, the results of this experiment also indicated that GFoLDS' built-in elementary linguistic knowledge likely facilitates the model's performance on more complex tasks: GFoLDS immediately began improving on the RELPRON test set, while the BERT models did not begin to see progress until halfway through the first epoch—roughly the same point at which they began improving on the elementary tasks.

However, Chapter 6 also uncovered a severe limitation of the GFoLDS model's positional encoding module, which impairs the model's ability to count long sequences of repeated nodes—which is critical for the negation-cancellation task of Chapter 2—and determine the sentence to which a given node belongs, potentially presenting a serious impediment to GFoLDS' performance on NLI tasks.

In Chapter 7, I used the limitations of GFoLDS uncovered in Chapters 4-6 to inform future directions in the development of language models over logical forms. In particular, I proposed a plausible remedy for the GFoLDS model's flawed positional encoding module, along with architectural modifications intended to overcome its inability to take multiple sentences as input, and to include CARGs and out-of-vocabulary items in its graph representations. I furthermore discussed an additional, yet-to-be-explored application of the current GFoLDS

model—namely, its use for lower-resource languages—and laid out a proposal for the next major phase of the research program of language models over logical forms: graph-to-graph generative models.

8.2 Limitations

I extensively described the limitations of the FoLDS and GFoLDS models in Chapters 2 and 6 (respectively): I instead dedicate this section to a discussion of the limitations of the research and findings of this dissertation in general.

Although the results of Chapter 5 demonstrate that GFoLDS can massively outperform superficial models on lexical and one-to-two-sentence tasks, we cannot definitively extrapolate from these results to assert that this advantage of language models over logical forms will carry over to multi-sentence tasks such as question-answering. It is likewise uncertain whether, after the modifications to GFoLDS proposed in Chapter 7 that are intended to overcome the limitations uncovered in Chapters 4-6, the model will retain the same advantages over superficial LMs that its current incarnation enjoys.

In a similar vein, it is not clear if the benefits of the GFoLDS model—or the Accelerated Learning Hypothesis—will transfer to the graph-to-graph generative model proposed in Chapter 7. Although Prange, Schneider, and Kong's (2022) linguistically-augmented GPT-2 (Radford et al., 2018) model (discussed in Chapters 1 and 4) suggests that the advantages of linguistic-knowledge-injection do in fact carry over to the generative setting, their findings are not guaranteed to transfer to a model purely over logical forms.

This discussion highlights the necessity of further research into language models over logical forms, in order to confirm the viability of such models as a replacement for superficial LMs, as is indicated by the findings of this dissertation.

Bibliography

Abramson, J.; Adler, J.; Dunger, J.; Evans, R.; Green, T.; Pritzel, A.; Ronneberger, O.; Willmore, L.; Ballard, A. J.; Bambrick, J.; Bodenstein, S. W.; Evans, D. A.; Hung, C.-C.; O'Neill, M.; Reiman, D.; Tunyasuvunakool, K.; Wu, Z.; Žemgulytė, A.; Arvaniti, E.; Beattle, C.; Bertolli, O.; Bridgland, A.; Cherepanov, A.; Congreve, M.; Cowen-Rivers, A. I.; Cowie, A.; Figurnov, M.; Fuchs, F. B.; Gladman, H.; Jain, Y. A.; Low, C. M. R.; Perlin, K.; Potapenko, A.; Savy, P.; Singh, S.; Stecula, A.; Thillaisundaram, A.; Tong, C.; Yakneed, S.; Zhong, E. D.; Zielinski, M.; Žídek, A.; Bapst, V.; Kohli, P.; Jaderberg, M.; Hassabis, D.; and Jumper, J. M. 2024. Accurate Structure Prediction of Biomolecular Interactions with AlphaFold 3. *Nature*, 1–3.

Adger, D.; and Harbour, D. 2008. Why Phi. *Phi-theory: Phi-features across Modules and Interfaces*, 1–34.

Aerts, S.; Kitto, K.; and Sitbon, L. 2011. Similarity Metrics within a Point of View. In *Quantum Interaction: 5th International Symposium*, 13–24.

Arvind, V.; Köbler, J.; Rattan, G.; and Verbitsky, O. 2015. On the Power of Color Refinement. In *Fundamentals of Computation Theory: 20th International Symposium*, 339–350.

Asai, A.; and Hajishirzi, H. 2020. Logic-Guided Data Augmentation and Regularization for Consistent Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5642–5650.

Bacciu, D.; Micheli, A.; and Podda, M. 2020. Edge-Based Sequential Graph Generation with Recurrent Neural Networks. *Neurocomputing*, 416: 177–189.

Bacciu, D.; and Podda, M. 2021. GraphGen-Redux: A Fast and Lightweight Recurrent Model for Labeled Graph Generation. In *2021 International Joint Conference on Neural Networks*, 1–8.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186.

Barklund, J. 1994. Bounded Quantifications for Iteration and Concurrency in Logic Programming. *New Generation Computing*, 12: 161–182.

Bassey, J.; Qian, L.; and Li, X. 2021. A Survey of Complex-Valued Neural Networks. *arXiv preprint arXiv:2101.12249*.

Beck, D.; Haffari, G.; and Cohn, T. 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)*, 273–283.

Belinkov, Y.; and Glass, J. 2019. Analysis Methods in Neural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 7: 49–72.

BNC. 2007. The British National Corpus, Version 3. Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium.

Boleda, G.; and Herbelot, A. 2016. Formal Distributional Semantics: Introduction to the Special Issue. *Computational Linguistics*, 42: 619–635.

Borji, A. 2023. A Categorical Archive of ChatGPT Failures. *arXiv preprint arXiv:2302.03494*.

Bos, J. 2011. A Survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding. *Language and Linguistics Compass*, 5: 336–366.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A Large Annotated Corpus for Learning Natural Language Inference. *arXiv preprint arXiv:1508.05326*.

Brachman, R. J.; and Schmolze, J. G. 1989. An Overview of the KL-ONE Knowledge Representation System. *Readings in Artificial Intelligence and Databases*, 207–230.

Bresson, X.; and Laurent, T. 2019. A Two-Step Graph Convolutional Decoder for Molecule Generation. *arXiv preprint arXiv:1906.03412*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, 1877–1901.

Buys, J.; and Blunsom, P. 2017. Robust Incremental Neural Semantic Graph Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1215–1226.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-End Object Detection with Transformers. In *European Conference on Computer Vision*, 213–229.

Caruana, R.; Lawrence, S.; and Giles, C. 2000. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. *Advances in Neural Information Processing Systems*, 13.

Chami, I.; Abu-El-Haija, S.; Perozzi, B.; Ré, C.; and Murphy, K. 2022. Machine Learning on Graphs: A Model and Comprehensive Taxonomy. *Journal of Machine Learning Research*, 23(89): 1–64.

Chaves, R. P.; and Richter, S. N. 2021. Look at That! BERT Can Be Easily Distracted from Paying Attention to Morphosyntax. *Proceedings of the Society for Computation in Linguistics*, 4(1): 28–38.

Chiang, D.; Cholak, P.; and Pillay, A. 2023. Tighter Bounds on the Expressivity of Transformer Encoders. In *Proceedings of the International Conference on Machine Learning*.

Chien, T.; and Kalita, J. 2020. Adversarial Analysis of Natural Language Inference Systems. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, 1–8.

Cho, K. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Choromanski, K. M.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J. Q.; Mohiuddin, A.; Kaiser, L.; Belanger, D. B.; Colwell, L. J.; and Weller, A. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations*.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2021. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*.

Colin, E.; and Gardent, C. 2018. Generating Syntactic Paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 937–943.

Copestake, A. 2009. 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)*, 1–9.

Copestake, A.; Emerson, G.; Goodman, M. W.; Horvat, M.; Kuhnle, A.; and Muszyńska, E. 2016. Resources for Building Applications with Dependency Minimal Recursion Semantics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1240–1247.

Copestake, A.; Flickinger, D.; Pollard, C.; and Sag, I. A. 2005. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation*, 3: 281–332.

Copestake, A. A.; and Flickinger, D. 2000. An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG. In *LREC*, 591–600.

Cramer, B.; and Zhang, Y. 2009. Construction of a German HPSG Grammar from a Detailed Treebank. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks (GEAF)*, 37–45.

Darwiche, A. 2001. Decomposable Negation Normal Form. *Journal of the ACM (JACM)*, 48: 608–647.

DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

Dowty, D. 1989. On the Semantic Content of the Notion of 'Thematic Role'. *Properties, Types and Meaning*, 2.

Dowty, D. 1991. Thematic Proto-Roles and Argument Selection. *Language*, 67: 547–619.

Du, Y.; Guo, X.; Cao, H.; Ye, Y.; and Zhao, L. 2022. Disentangled Spatiotemporal Graph Generative Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 6541–6549.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs. *arXiv preprint arXiv:2012.09699*.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023. Benchmarking Graph Neural Networks. *Journal of Machine Learning Research*, 24(43): 1–48.

Elman, J. L. 1990. Finding Structure in Time. *Cognitive Science*, 14: 179–211.

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

Emirkanian, L.; Da Sylva, L.; and Bouchard, L. H. 1996. The Implementation of a Computational Grammar of French using the Grammar Development Environment. In *COLING: The 16th International Conference on Computational Linguistics*.

Erk, K. 2016. What Do You Know about an Alligator when You Know the Company It Keeps? *Semantics and Pragmatics*, 9: 17–1.

Ettinger, A. 2020. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8: 34–48.

Ferraresi, A.; Zanchetta, E.; Baroni, M.; and Bernardini, S. 2008. Introducing and Evaluating

ukWaC, a Very Large Web-Derived Corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can We Beat Google*, 47–54.

Fine, K. 2014. Truth-Maker Semantics for Intuitionistic Logic. *Journal of Philosophical Logic*, 43: 549–577.

Flam-Shepherd, D.; Wu, T.; and Aspuru-Guzik, A. 2020. Graph Deconvolutional Generation. *arXiv preprint arXiv:2002.07087*.

Fyodorov, Y.; Winter, Y.; and Francez, N. 2000. A Natural Logic Inference System. In *Proceedings of the 2nd Workshop on Inference in Computational Semantics (ICoS-2)*.

Ganea, O.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic Neural Networks. *Advances in Neural Information Processing Systems*, 31.

Godbole, V.; Dahl, G. E.; Gilmer, J.; Shallue, C. J.; and Nado, Z. 2023. Deep Learning Tuning Playbook. Version 1.0.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, 27.

Goodman, M. W. 2019. A Python Library for Deep Linguistic Resources. In *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*. Singapore.

Goodman, M. W. 2020. Penman: An Open-Source Library and Tool for AMR Graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 312–319.

Goyal, N.; Jain, H. V.; and Ranu, S. 2020. GraphGen: A Scalable Approach to Domain-Agnostic Labeled Graph Generation. In *Proceedings of The Web Conference 2020*, 1253–1263.

Graff, D.; and Cieri, C. 2003. English Gigaword Corpus. *Linguistic Data Consortium*.

Granziol, D.; Zohren, S.; and Roberts, S. 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.

Grefenstette, E.; and Sadrzadeh, M. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1394–1404.

Grefenstette, E.; Sadrzadeh, M.; Clark, S.; Coecke, B.; and Pulman, S. 2014. Concrete Sentence Spaces for Compositional Distributional Models of Meaning. In *Computing Meaning*, 71–86. Springer.

Guo, Z.; Zhang, Y.; and Lu, W. 2019. Attention Guided Graph Convolutional Networks for Relation Extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 241–251.

Guo, Z.; Zhang, Y.; Teng, Z.; and Lu, W. 2019. Densely Connected Graph Convolutional Networks for Graph-to-Sequence Learning. *Transactions of the Association for Computational Linguistics*, 7: 297–312.

Habernal, I.; Wachsmuth, H.; Gurevych, I.; and Stein, B. 2018. SemEval-2018 Task 12: The Argument Reasoning Comprehension Task. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, 763–772.

Hajdik, V.; Buys, J.; Goodman, M. W.; and Bender, E. M. 2019. Neural Text Generation from Rich Semantic Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2259–2266.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems*, 30.

Hanin, B. 2018. Which Neural Net Architectures Give Rise to Exploding and Vanishing Gradients? *Advances in Neural Information Processing Systems*, 31.

He, F.; Liu, T.; and Tao, D. 2020. Why ResNet Works? Residuals Generalize. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12): 5349–5362.

He, P.; Liu, X.; Gao, J.; and Chen, W. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*.

Herbelot, A.; and Copestake, A. 2021. Ideal Words: a Vector-Based Formalisation of Semantic Competence. *Künstliche Intelligenz*, 35: 271–290.

Herbelot, A.; and Vecchi, E. M. 2015. Building a Shared World: Mapping Distributional to Model-Theoretic Semantic Spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 22–32.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D. d. L.; Hendricks, L. A.; Welbl, J.; Clark, A.; Hennigan, T.; Noland, E.; Millican, K.; van den Driessche, G.; Damoc, B.; Guy, A.; Osindero, S.; Simonyan, K.; Elsen, E.; Rae,

J. W.; Vinyals, O.; and Sifre, L. 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556*.

Hossain, M. M.; Kovatchev, V.; Dutta, P.; Kao, T.; Wei, E.; and Blanco, E. 2020. An Analysis of Natural Language Inference Benchmarks through the Lens of Negation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9106–9118.

Hosseini, A.; Reddy, S.; Bahdanau, D.; Hjelm, R. D.; Sordoni, A.; and Courville, A. 2021. Understanding by Understanding Not: Modeling Negation in Language Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1301–1312.

Hovda, P. 2009. What is Classical Mereology? *Journal of Philosophical Logic*, 38: 55–82.

Hu, Y.; Shen, H.; Liu, W.; Min, F.; Qiao, X.; and Jin, K. 2021. A Graph Convolutional Network With Multiple Dependency Representations for Relation Extraction. *IEEE Access*, 9: 81575–81587.

Huber, P. J. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1).

Huebner, P. A.; Sulem, E.; Cynthia, F.; and Roth, D. 2021. BabyBERTa: Learning More Grammar With Small-Scale Child-Directed Language. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, 624–646.

Immerman, N. 2012. *Descriptive Complexity*. Springer Science & Business Media.

Jacobsen, M.; Sørensen, M. H.; and Derczynski, L. 2021. Optimal Size-Performance Tradeoffs: Weighing POS Tagger Models. *arXiv preprint arXiv:2104.07951*.

Jang, M.; Kwon, D. S.; and Lukasiewicz, T. 2022. BECEL: Benchmark for Consistency

Evaluation of Language Models. In *Proceedings of the 29th International Conference on Computational Linguistics*, 3680–3696.

Johns, B. T.; and Jones, M. N. 2012. Perceptual Inference through Global Lexical Similarity. *Topics in Cognitive Science*, 4: 103–120.

Jones, C. R.; Chang, T. A.; Coulson, S.; Michaelov, J. A.; Trott, S.; and Bergen, B. 2022. Distrubutional Semantics Still Can't Account for Affordances. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 44.

Kassner, N.; and Schütze, H. 2020. Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7811–7818.

Khot, T.; Sabharwal, A.; and Clark, P. 2018. SciTaiL: A Textual Entailment Dataset from Science Question Answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1): 81575–81587.

Kim, J.-H.; On, K.-W.; Lim, W.; Kim, J.; Ha, J.-W.; and Zhang, B.-T. 2016. Hadamard Product for Low-Rank Bilinear Pooling. *arXiv preprint arXiv:1610.04325*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

Kondor, R. I.; and Lafferty, J. D. 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Proceedings of the 19th International Conference on Machine Learning*, 315–322.

Kubota, Y. 2010. *(In)flexibility of Constituency in Japanese in Multi-Modal Categorial Grammar with Structured Phonology*. Ph.D. thesis, The Ohio State University.

Kubota, Y.; and Levine, R. D. 2020. *Type-Logical Syntax*. MIT Press.

Kuhn, H. W. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2: 83–97.

Larrivée, P. 2016. The Markedness of Double Negation. *Negation and Polarity: Experimental Perspectives*, 177–198.

Laverghetta Jr., A.; and Licato, J. 2022. Developmental Negation Processing in Transformer Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 545–551.

Laverghetta Jr., A.; Nighojkar, A.; Mirzakhalov, J.; and Licato, J. 2021. Can Transformer Language Models Predict Psychometric Properties? In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, 12–25.

LeCun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; and Jackel, L. 1989. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 2.

Leman, A.; and Weisfeiler, B. 1968. A Reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9): 12–16.

Lenci, A. 2018. Distributional Models of Word Meaning. *Annual Review of Linguistics*, 4: 151–171.

Levi, F. W. 1942. Finite Geometrical Systems.

Levy, E.; and Nelson, K. 1994. Words in Discourse: A Dialectical Approach to the Acquisition of Meaning and Use. *Journal of Child Language*, 21: 367–389.

Lewis, D. 1976. General Semantics. In *Montague Grammar*, 1–50. Elsevier.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 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*, 7871–7880.

Li, H. 2022. Language Models: Past, Present, and Future. *Communications of the ACM*, 65(7): 56–63.

Li, Y.; Cui, L.; Yan, J.; Yin, Y.; Bi, W.; Shi, S.; and Zhang, Y. 2023. Explicit Syntactic Guidance for Neural Text Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14095–14112.

Li, Y.; Zemel, R.; Brockschmidt, M.; and Tarlow, D. 2016. Gated Graph Sequence Neural Networks. In *Proceedings of ICLR'16*.

Lim, D.; Robinson, J. D.; Zhao, L.; Smidt, T.; Sra, S.; Maron, H.; and Jegelka, S. 2023. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. In *The Eleventh International Conference on Learning Representations*.

Lin, B. Y.; Chen, X.; Chen, J.; and Ren, X. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2829–2839.

Lin, Z.; Liu, J. Z.; and Shang, J. 2022. Towards Collaborative Neural-Symbolic Graph Semantic Parsing via Uncertainty. In *Findings of the Association for Computational Linguistics: ACL 2022*, 4160–4173.

Liu, N. F.; Schwartz, R.; and Smith, N. A. 2019. Inoculation by Fine-Tuning: A Method for Analyzing Challenge Datasets. 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), 2171–2179.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. 2018. Constrained Graph Variational Autoencoders for Molecule Design. *Advances in Neural Information Processing Systems*, 31.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Lo, C. H.; Cheng, H.; Lam, W.; and Emerson, G. 2023. Functional Distributional Semantics at Scale. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, 423–436.

Lopez, V.; Motta, E.; Uren, V.; and Sabou, M. 2007. State of the Art on Semantic Question Answering.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983*.

Loshchilov, I.; and Hutter, F. 2017. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*.

Mafi, M.; Martin, H.; Cabrerizo, M.; Andrian, J.; Barreto, A.; and Adjouadi, M. 2019. A Comprehensive Survey on Impulse and Gaussian Denoising Filters for Digital Images. *Signal Processing*, 157: 236–260.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Mao, A.; Mohri, M.; and Zhong, Y. 2023. Cross-Entropy Loss Functions: Theoretical Analysis and Applications. In *International Conference on Machine Learning*, 23803–23828.

Marcus, M.; and Moyls, B. 1959. Transformations on Tensor Product Spaces. *Pacific Journal of Mathematics*, 9: 1215–1221.

McCoy, T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3428–3448.

McRae, K.; Cree, G. S.; Seidenberg, M. S.; and McNorgan, C. 2005. Semantic Feature Production Norms for a Large Set of Living and Nonliving Things. *Behavior Research Methods*, 37(4): 547–559.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. GraphiT: Encoding Graph Structure in Transformers. *arXiv preprint arXiv:2106.05667*.

Miller, G. A. 1995. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11): 39–41.

Mitsuishi, Y.; Torisawa, K.; and Tsujii, J. 1998. HPSG-Style Underspecified Japanese Grammar with Wide Coverage. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, 876–880.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4602–4609.

Muennighoff, N.; Rush, A.; Barak, B.; Le Scao, T.; Tazi, N.; Piktus, A.; Pyysalo, S.; Wolf, T.; and Raffel, C. A. 2024. Scaling Data-Constrained Language Models. *Advances in Neural Information Processing Systems*, 36.

Muszynska, E. 2020. *Semantic Chunking*. Ph.D. thesis, University of Cambridge.

Naik, A.; Ravichander, A.; Sadeh, N.; Rose, C.; and Neubig, G. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2340–2353.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, 280–290.

Nan, G.; Guo, Z.; Sekulic, I.; and Lu, W. 2020. Reasoning with Latent Structure Refinement for Document-Level Relation Extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1546–1557.

Nguyen, X.-P.; Joty, S.; Hoi, S.; and Socher, R. 2020. Tree-Structured Attention with Hierarchical Accumulation. In *International Conference on Learning Representations*.

Nickel, M.; and Kiela, D. 2017. Poincaré Embeddings for Learning Hierarchical Representations. *Advances in Neural Information Processing Systems*, 30.

Nie, Y.; Williams, A.; Dinan, E.; Bansal, M.; Weston, J.; and Kiela, D. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4885–4901.

Niven, T.; and Kao, H.-Y. 2019. Probing Neural Network Comprehension of Natural Language Arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4658–4664.

Nocedal, J. 1980. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151): 773–782.

Oepen, S.; and Lønning, J. T. 2006. Discriminant-based MRS Banking. In *LREC*, 1250–1255.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Patalano, A. L.; Wengrovitz, S. M.; and Sharpes, K. M. 2009. The Influence of Category Coherence on Inference about Cross-Classified Entities. *Memory and Cognition*, 37: 21–28.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710.

Piantadosi, S.; and Hill, F. 2022. Meaning without Reference in Large Language Models. In *NeurIPS Workshop on Neuro Causal and Symbolic AI (nCSI)*.

Plenz, M.; and Frank, A. 2024. Graph Language Models. *arXiv preprint arXiv:2401.07105*.

Prange, J.; Schneider, N.; and Kong, L. 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*, 4375–4391.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2018. Language Models are Unsupervised Multitask Learners.

Rae, J. W.; Borgeaud, S.; Cai, T.; Millican, K.; Hoffmann, J.; Song, F.; Aslanides, J.; Henderson, S.; Ring, R.; Young, S.; Rutherford, E.; Hennigan, T.; Menick, J.; Cassirer, A.; Powell, R.; van den Driessche, G.; Hedricks, L. A.; Rauh, M.; Huang, P.-S.; Glaese, A.; Welbl, J.; Dathathri, S.; Huang, S.; Uesato, J.; Mellor, J.; Higgins, I.; Creswell, A.; McAleese, N.; Wu, A.; Elsen, E.; Jayakumar, S.; Buchatskaya, E.; Budden, D.; Sutherland, E.; Simonyan, K.; Paganini, M.; Sifre, L.; Martens, L.; Li, X. L.; Kuncoro, A.; Nematzadeh, A.; Gribovskaya, E.; Donato, D.; Lazaridou, A.; Mensch, A.; Lespiau, J.-B.; Tsimpoukelli, M.; Grigorev, N.; Fritz, D.; Sottiaux, T.; Pajarskas, M.; Pohlen, T.; Gong, Z.; Toyama, D.; de Masson d'Autume, C.; Yujia, L.; Terzi, T.; Mikulik, V.; Babuschkin, I.; Clark, A.;

de Las Casas, D.; Guy, A.; Jones, C.; Bradbury, J.; Johnson, M.; Hechtman, B.; Weidinger, L.; Gabriel, I.; Issac, W.; Lockhart, S., Ed adn Osindero; Rimell, L.; Dyer, C.; Vinyals, O.; Ayoub, K.; Stanway, J.; Bennett, L.; Hassabis, D.; Kavukcuoglu, K.; and Irving, G. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv preprint arXiv:2112.11446*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515.

Richardson, K.; Hu, H.; Moss, L.; and Sabharwal, A. 2020. Probing Natural Language Inference Models through Semantic Fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8713–8721.

Riemann, B. 1854. Über die Hypothesen, Welche der Geometrie zu Grunde Liegen. *Königliche Gesellschaft der Wissenschaften und der Georg-Augustus-Universität Göttingen*, 13(133): 1867.

Rimell, L.; Maillard, J.; Polajnar, T.; and Clark, S. 2016. RELPRON: A Relative Clause Evaluation Data Set for Compositional Distributional Semantics. *Computational Linguistics*, 42(4): 661–701.

Rives, A.; Meier, J.; Sercu, T.; Goyal, S.; Lin, Z.; Liu, J.; Guo, D.; Ott, M.; Zitnick, C. L.; Ma, J.; et al. 2021. Biological Structure and Function Emerge from Scaling Unsupervised

Learning to 250 Million Protein Sequences. *Proceedings of the National Academy of Sciences*, 118(15).

Rogers, A.; Kovaleva, O.; and Rumshisky, A. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8: 842–866.

Roller, S.; Erk, K.; and Boleda, G. 2014. Inclusive yet Selective: Supervised Distributional Hypernymy Detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 1025–1036.

Rosenfeld, A.; and Erk, K. 2022. An Analysis of Property Inference Methods. *Natural Language Engineering*, 1–27.

Sachan, D.; Zhang, Y.; Qi, P.; and Hamilton, W. L. 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*, 2647–2661.

Salman, S.; and Liu, X. 2019. Overfitting Mechanism and Avoidance in Deep Neural Networks. *arXiv preprint arXiv:1901.06566*.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, 593–607.

Shin, S.; Lee, S.-W.; Ahn, H.; Kim, S.; Kim, H.; Kim, B.; Cho, K.; Lee, G.; Park, W.; Ha, J.-W.; and Sung, N. 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*, 5168–5186.

Sparck Jones, K. 1972. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 28: 11–21.

Spearman, C. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1).

Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Steedman, M. 1993. Categorial Grammar. *Lingua*, 90: 221–258.

Su, J.; Ahmed, M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568.

Sullivan, M. 2023. Formal-Logical Distributional Semantics: Applications to Property Inference. In *Workshop on Knowledge Augmented Methods for Natural Language Processing at AAAI 2023*.

Sullivan, M. 2024. It is not True that Transformers are Inductive Learners: Probing NLI Models with External Negation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1924–1945.

Sun, H.; Liu, X.; Gong, Y.; Zhang, Y.; Jiang, D.; Yang, L.; and Duan, N. 2023. Allies: Prompting Large Language Model with Beam Search. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3794–3805.

Sun, Z.; Fan, C.; Han, Q.; Sun, X.; Meng, Y.; Wu, F.; and Li, J. 2020. Self-Explaining Structures Improve NLP Models. *arXiv preprint arXiv:2012.01786*.

Swersky, K.; Sutskever, I.; Tarlow, D.; Zemel, R.; Salakhutdinov, R. R.; and Adams, R. P. 2012. Cardinality Restricted Boltzmann Machines. *Advances in Neural Information Processing Systems*, 25.

Szűcs, P. 2014. On English Topicalization and Left-Dislocation from an Information-Structural Perspective. *Journal of Linguistics*, 49: 413–454.

Talukdar, P. P.; and Crammer, K. 2009. New Regularized Algorithms for Transductive Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 442–457. Springer.

Tang, C.; Zhang, H.; Loakman, T.; Lin, C.; and Guerin, F. 2023. Enhancing Dialogue Generation via Dynamic Graph Knowledge Aggregation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 4604–4616.

Tay, Y.; Dehghani, M.; Rao, J.; Fedus, W.; Abnar, S.; Chung, H. W.; Narang, S.; Yogatama, D.; Vaswani, A.; and Metzler, D. 2022. Scale Efficiently: Insights from Pretraining and Finetuning Transformers. In *International Conference on Learning Representations*.

Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; and Mittal, A. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 809–819.

Tifrea, A.; Becigneul, G.; and Ganea, O.-E. 2018. Poincare Glove: Hyperbolic Word Embeddings. In *International Conference on Learning Representations*.

Tong, Z.; Liang, Y.; Sun, C.; Rosenblum, D. S.; and Lim, A. 2020. Directed Graph Convolutional Network. *arXiv preprint arXiv:2004.13970*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.;

Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Van Valin, R. D. 1999. Generalized Semantic Roles and the Syntax-Semantics Interface. *Empirical Issues in Formal Syntax and Semantics*, 2: 373–389.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all You Need. *Advances in Neural Information Processing Systems*, 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Venhuizen, N. J.; Hendriks, P.; Crocker, M. W.; and Brouwer, H. 2022. Distributional Formal Semantics. *Information and Computation*, 287.

Villalobos, P.; Ho, A.; Sevilla, J.; Besiroglu, T.; Heim, L.; and Hobbhahn, M. 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*.

Vinson, D. P.; and Vigliocco, G. 2002. A Semantic Analysis of Grammatical Class Impairments: Semantic Representations of Object Nouns, Action Nouns and Action Verbs. *Journal of Neurolinguistics*, 15: 317–351.

Waldis, A.; Perlitz, Y.; Choshen, L.; Hou, Y.; and Gurevych, I. 2024. Holmes: Benchmark the Linguistic Competence of Language Models. *arXiv preprint arXiv:2404.18923*.

Wang, B.; Wang, S.; Cheng, Y.; Gan, Z.; Jia, R.; Li, B.; and Liu, J. 2021. InfoBERT: Improving Robustness of Language Models from An Information Theoretic Perspective. In *International Conference on Learning Representations*.

Wang, T.; Wan, X.; and Jin, H. 2020. AMR-To-Text Generation with Graph Transformer. *Transactions of the Association for Computational Linguistics*, 8: 19–33.

Wang, T.; Wan, X.; and Yao, S. 2020. Better AMR-To-Text Generation with Graph Structure Reconstruction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 3919–3925.

Wang, X.; Yeshwanth, C.; and Nießner, M. 2021. Sceneformer: Indoor Scene Generation with Transformers. In *2021 International Conference on 3D Vision (3DV)*, 106–115. IEEE.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Wettig, A.; Gao, T.; Zhong, Z.; and Chen, D. 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*, 2985–3000.

White, A. S.; Rudinger, R.; Rawlins, K.; and Van Durme, B. 2018. Lexicosyntactic Inference in Neural Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4717–4724.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *arXiv preprint arXiv:1704.05426*.

Winter, Y. 2016. *Elements of Formal Semantics: An Introduction to the Mathematical Theory of Meaning in Natural Language*. Edinburgh University Press.

Withagen, R.; De Poel, H. J.; Araújo, D.; and Pepping, G.-J. 2012. Affordances Can Invite Behavior: Reconsidering the Relationship between Affordances and Agency. *New Ideas in Psychology*, 30: 250–258.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; Kaiser, Ł.; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021. Representing Long-Range Context for Graph Neural Networks with Global Attention. *Advances in Neural Information Processing Systems*, 34: 13266–13279.

Wu, Z.; Liu, Z.; Lin, J.; Lin, Y.; and Han, S. 2020. Lite Transformer with Long-Short Range Attention. In *International Conference on Learning Representations*.

Wu, Z.; Peng, H.; and Smith, N. A. 2021. Infusing Finetuning with Semantic Dependencies. *Transactions of the Association for Computational Linguistics*, 9: 226–242.

Xia, M.; Artetxe, M.; Zhou, C.; Lin, X. V.; Pasunuru, R.; Chen, D.; Zettlemoyer, L.; and Stoyanov, V. 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)*, 13711–13738.

Xu, K.; Wu, L.; Wang, Z.; Feng, Y.; Witbrock, M.; and Sheinin, V. 2018. Graph2seq: Graph to Sequence Learning with Attention-Based Neural Networks. *arXiv preprint arXiv:1804.00823*.

Xu, Z.; Guo, D.; Tang, D.; Su, Q.; Shou, L.; Gong, M.; Zhong, W.; Quan, X.; Jiang, D.; and Duan, N. 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), 5412–5422.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *International Conference on Machine Learning*, 40–48. PMLR.

Yasunaga, M.; Bosselut, A.; Ren, H.; Zhang, X.; Manning, C. D.; Liang, P. S.; and Leskovec, J. 2022. Deep Bidirectional Language-Knowledge Graph Pretraining. *Advances in Neural Information Processing Systems*, 35: 37309–37323.

Yin, Y.; Meng, F.; Su, J.; Zhou, C.; Yang, Z.; Zhou, J.; and Luo, J. 2020. A Novel Graph-Based Multi-Modal Fusion Encoder for Neural Machine Translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3025–3035.

Young, I. T.; and Van Vliet, L. J. 1995. Recursive Implementation of the Gaussian Filter. *Signal Processing*, 44(2): 139–151.

Yuan, Z.; Hu, S.; Vulić, I.; Korhonen, A.; and Meng, Z. 2023. Can Pretrained Language Models (Yet) Reason Deductively? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 1447–1462.

Zadeh, L. A. 1965. Fuzzy Sets. *Information and Control*, 8: 338–353.

Zamaraeva, O.; Allegue, L. S.; and Gómez-Rodríguez, C. 2024. Spanish Resource Grammar Version 2023. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 15093–15104.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 793–803.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020a. Graph-BERT: Only Attention is Needed for Learning Graph Representations. *arXiv preprint arXiv:2001.05140*.

Zhang, S.; Dinan, E.; Urbanek, J.; Szlam, A.; Kiela, D.; and Weston, J. 2018. Personalizing Dialogue Agents: I Have a Dog, Do You Have Pets Too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2204–2213.

Zhang, S.; Ning, Q.; and Huang, L. 2022. Extracting Temporal Event Relation with Syntax-guided Graph Transformer. In *Findings of the Association for Computational Linguistics: NAACL 2022*, 379–390.

Zhang, X.; Bosselut, A.; Yasunaga, M.; Ren, H.; Liang, P.; Manning, C. D.; and Leskovec, J. 2022. GreaseLM: Graph REASoning Enhanced Language Models. In *International Conference on Learning Representations*.

Zhang, Y.; Guo, Z.; Teng, Z.; Lu, W.; Cohen, S. B.; Liu, Z.; and Bing, L. 2020b. Lightweight, Dynamic Graph Convolutional Networks for AMR-to-Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2162–2172.

Zhang, Z.; Wu, Y.; Zhao, H.; Li, Z.; Zhang, S.; Zhou, X.; and Zhou, X. 2020c. Semantics-Aware BERT for Language Understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9628–9635.

Zhou, H.; Young, T.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Commonsense Knowledge Aware Conversation Generation with Graph Attention. In *IJCAI*, 4623–4629.

Zhou, J.; Zhang, Z.; Zhao, H.; and Zhang, S. 2020. LIMIT-BERT: Linguistics Informed Multi-Task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4450–4461.

Zhu, M. 2004. Recall, Precision and Average Precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30): 6.

Zhu, Y.; Du, Y.; Wang, Y.; Xu, Y.; Zhang, J.; Liu, Q.; and Wu, S. 2022. A Survey on Deep Graph Generation: Methods and Applications. In *The First Learning on Graphs Conference*.

Zong, C.; Xia, F.; Li, W.; and Navigli, R. 2021. Dependency-driven Relation Extraction with Attentive Graph Convolutional Networks. 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)*, 4458–4471.