On Translation as a Symmetry of Language

ON TRANSLATION AS A SYMMETRY OF LANGUAGE

A thesis presented

by

CAMILO BROWN-PINILLA

Presented to

The Department of Applied Mathematics

in partial fulfillment of the requirements for the degree with honors of

Bachelor of Arts

HARVARD COLLEGE

Cambridge, Massachusetts

March, 2026

DEDICATION

A mi mamá,
Martha Lucia Pinilla,
quien trabajó
para que yo pudiera
disfrutar de la riqueza que es
aprender.

To my mother
Martha Lucia Pinilla,
who worked
so that I could
enjoy the bliss that is
to learn.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to offer my deepest gratitude to my advisor, Professor Melanie Weber. In the classroom, your excitement is infectious. In the lab, your expertise is inspiring and your guidance is second to none. I distinctly remember the feeling of wonder I always had leaving your lectures, amazed by the elegance of the geometric perspective and its seemingly endless applications. Since then, I have gotten the opportunity to work closely with you and channel that curiosity into this thesis. Learning from you has shaped my perspectives and interests in a way that will likely be permanent, and I could not be more grateful for that opportunity.

I would also like to sincerely thank Professor Stuart Shieber for the many conversations we have had on grammars, structure in language, and the philosophy of AI. As a student and then a teaching fellow for your class on Natural Language Processing, much of what you have taught me serves as the foundation upon which all of this work rests. Thank you.

ABSTRACT

Geometric Deep Learning (GDL) unifies diverse machine learning architectures under the principle of symmetry. By designing models to respect the invariances inherent in data, GDL achieves improved sample efficiency and robustness without relying on scale alone. Modern Natural Language Processing (NLP), by contrast, treats text as an unstructured sequence of tokens, relying on massive scale to implicitly discover linguistic regularities. In this thesis, we bridge these two paradigms by observing that meaning is an invariant of cross-lingual translation and that this invariance can be formalized and exploited in the GDL sense. Concretely, we use Abstract Meaning Representations (AMRs) as a language-independent canonical form to which sentences in any language are mapped, and embed these graphs into a space whose geometry reflects semantic structure. Because the canonical form abstracts away from surface-level linguistic variation, the resulting embeddings are translation-invariant by construction. We then inject these embeddings into pre-trained LLMs via Soft Prompt Tuning, conditioning translation on a geometric semantic signal using only four learned prefix tokens. Evaluated on translation across English paired with German, Spanish, Italian, and Chinese using the COMET metric, our augmented systems achieve scores matching or exceeding conventional LoRA fine-tuning while requiring roughly one-eighth the learnable parameters. These results suggest that explicitly encoding symmetry as an inductive bias offers a principled, parameter-efficient complement to the dominant paradigm of scale.

Chapter 1

INTRODUCTION

«Ningún problema tan consustancial con las letras y con su modesto misterio como el que propone una traducción.»

"No problem is as consubstantial with literature and its modest mystery as that posed by translation."

Jorge Luis Borges, *Las versiones homéricas*

Within the past five years, the term "artificial intelligence" has exploded in relevance. Indeed, the technology labeled as AI has become central to conversations in areas spanning education, innovation, and even war. This shift was effectively mobilized by a particular type of system called the "Large Language Model" (LLM) which, by being trained to predict the next word in a sequence, can output remarkably fluent text.

The success of modern LLMs is due largely in part to their incredible size. Empirical trends known as scaling laws (*1*) reveal that performance scales as a power-law with model and dataset size, prompting us to aggressively and confidently scale them in return for increased capability. While this approach has certainly worked very well in practice, there are limits to scale. With models approaching parameter counts in the *trillions* (*2*), we must come to terms with the fact that the supply of data to train them on is not infinite. As studies begin to project that we might soon run out of data (*3*), we must begin to ask: *is scale alone really the answer?*

To poke at this question, we zoom out and examine the implications of the fact that we are even able to learn in high dimensions in the first place. According to the *curse of dimensionality*, the number of data points needed to learn a function is exponential in the dimension of the data. Yet, modern LLMs learn on several-

thousand-dimensional representations of text and exhibit remarkable generalization. How could this be?

Part of the answer lies in *geometric structure*. Far from being random samples from a high-dimensional space, data often lie on low-dimensional manifolds embedded in that space. Because of this, much of our being able to learn effectively lies in the ability for models to capture the underlying regularity of data.

Thus, to investigate what can be done to curb our reliance on scale, we ask:

What geometric structure does language have?

Quite the loaded question. To begin to make sense of it, we turn to the aptly-named field of Geometric Deep Learning(GDL) which, broadly speaking, prescribes a principled tonic against the curse of dimensionality by *explicitly* designing model classes $\mathcal{F}$ to respect the underlying structure of data.

Inspired by Felix Klein's 1872 *Erlangen Program*, GDL is built on the language of group theory in an attempt to unify the "zoo" of machine learning practice under the principle of *symmetry* (*4*). For example, let us suppose we are trying to build a model that predicts the acidity of molecules. How do we go about designing $\mathcal{F}$? By reasoning in terms of symmetries, we observe that our prediction should not depend on the 3D orientation or position of the molecule. In other words, $\mathcal{F}$ should be *invariant* under actions from $SE(3)$, the group of rigid transformations in $\mathbb{R}^3$. By constructing $\mathcal{F}$ as the composition of $SE(3)-$Invariant components, we guarantee respect for this regularity, entirely eliminating our need to estimate this property from data.

Indeed, this approach has worked wonders in terms of sample-efficiency and robustness, becoming the default approach for learning on physical data. However, it also requires knowledge of the underlying symmetry group and a suitable representation of it. Is it possible for us to extend this philosophy to natural language systems?

To offer an answer to this question, we must begin with symmetries. Abstractly, one can think of a symmetry as a transformation that, when applied to an object, leaves some fundamental property of that object unchanged. In the case of the molecule, it was clear that altering its orientation had no effect on its acidity, and hence, in the context of chemical properties, rigid transformations are symmetries. What, then, are the symmetries of language?

Certainly, the *meaning* of a piece of text is among its most fundamental properties. Then, as we just saw, we can simply rephrase this as asking what transformations we can apply to a text that leave its meaning unchanged. Approaching the question in this way, one answer falls out almost immediately: *translation*. After all, the entire point of cross-lingual translation is to send a sentence in language A to language B such that its meaning is preserved.

In this thesis, taking cross-lingual translation to be a symmetry of language, we shall show that the answer to whether the GDL philosophy can be extended to natural language systems is *yes*, if only empirically. Further, we show that doing so results in a learnable-parameter-efficient system that rivals and even beats the performance that the unstructured approach to language modeling offers. Along the way, we offer exposition on the history of Natural Language Processing and the principles of Geometric Deep Learning. We hope that this novel perspective on Natural Language Processing might add to the excitement around the intersection of geometry and language and open the door to further consideration of structured language modeling.

Contributions

To summarize, this thesis contributes the following:[1]

- A novel, symmetry-first approach to language modeling inspired by GDL.

[1]The code for this project can be found at https://github.com/Weber-GeoML/translation-invariance

- An empirical demonstration that this perspective results in a learnable-parameter-efficient system and competitive machine translation pipeline.
- Thorough exposition on the relevant backgrounds and their connections.

Chapter 2

BACKGROUND

This chapter provides the technical foundations for the 3 central pieces that overlap in this thesis: Natural Language Processing (NLP), Geometric Deep Learning, and the Abstract Meaning Representation. We begin with a brief overview of the evolution of NLP to the current state-of-the-art, leading to a detailed description of the Transformer. We then introduce Geometric Deep Learning, the motivating framework behind this thesis, giving formal descriptions of the notions of symmetry and invariance that will be central to chapters 3 and 4. Finally, we describe the Abstract Meaning Representation, the semantic formalism we use to bridge NLP and GDL, ending with a discussion of related works.

2.1 Natural Language Processing

2.1.1 From Rules to Representations

The history of NLP can roughly be organized into three distinct eras, each defined by its dominant paradigm: rule-based in the beginning, statistical in the middle, and neural currently.

Rule-Based Systems

The rule-based era of NLP began roughly in the 1950s and continued into the 1980s. Unlike today, the dominant approach to NLP during this time was almost entirely linguistic in nature with minimal computation. The most representative systems of this time are *grammars*: formal systems consisting of a very large, handcrafted set of rules modeling the syntax of language, defining which sentences are valid in language and how their different components organize. One of the most famous

systems of this time is Chomsky's Context-Free Grammar (*5*), which formalized the hierarchical structure of sentences as production rules over symbols.

Context-free grammars and their extensions powered early systems for parsing, question answering, and machine translation, and their influence on the field cannot be overstated. The very notion that language has a hierarchical, tree-like structure that can be captured by formal rules remains foundational to linguistics today. However, rule-based systems suffered from a fundamental brittleness. Natural language is rife with ambiguity, exception, and context-dependence, and no handcrafted rule set could hope to cover the full diversity of human expression. Extending these systems to new domains or languages required enormous manual effort, and even then, coverage remained incomplete. However, despite their limitations in fitting *natural* languages, grammars are still foundational to *formal* languages. In particular, context-free grammars are used to define the syntax of programming languages and to support parsing in compilers and interpreters.

Statistical Approaches

Around the 1990s, the limitations posed by rule-based systems and the increasing availability of computational resources lead to the development of the statistical era of NLP. The core principles underlying these systems are quite simple. Indeed, the motivation for and success of statistical approaches is perhaps entirely summarized by the linguist J. R. Firth's now famous quote: "You shall know a word by the company it keeps".

Using largely only word co-occurence statistics, statistical systems made it possible for us to move from the arduous task of defining the rules of language to instead *learning* them from data. Statistical models such as n-grams, Hidden Markov Models (*6*) , and Probabilistic Context-Free Grammars replaced rigid rule-sets with probability distributions estimated over large text corpora, lending some "softness" to parsing languages by assigning low probability to unanticipated or ungrammatical

sentences instead of failing outright.

Machine translation was among the tasks that benefited the most from this paradigm shift. Though crude by today's standards, the IBM models (*7*) revolutionized machine translation, dramatically outperforming their rule-based predecessors by treating translation as a statistical task. This marked a paradigm shift to the rule of *data over design* that persists today.

While statistical models were a great improvement over their predecessors, they were not without their own limitations. Most pressingly, they suffered from the *curse of dimensionality*: as the vocabulary size and context window grew, the number of parameters to estimate exploded exponentially, and, on top of this, the data required to reliably estimate them grew in kind. An n-gram model, for instance, must estimate probabilities for *all* possible sequences of n words, the vast majority of which will never appear in even a very large corpus. This data sparsity problem meant that statistical models struggled to generalize beyond the exact patterns observed in their training data. Various smoothing techniques were developed to redistribute probability mass to unseen events (*8*), but these were ultimately palliative rather than curative.

A more fundamental issue was that statistical models, operating over discrete symbols, had no mechanism for capturing the notion that two words might be *similar*: the words "cat" and "kitten" were as unrelated as "cat" and "parliament" in the eyes of an n-gram model.

Neural Approaches

As with the rise of statistical approaches, the still expanding access to data and computation in the early 2000s made neural networks a feasible approach to language processing. By transforming symbolic language into dense *vector representations*, neural approaches were able to overcome the fundamental limitations of their predecessors. No longer was "cat" an arbitrary index in a vocabulary. Instead, it was

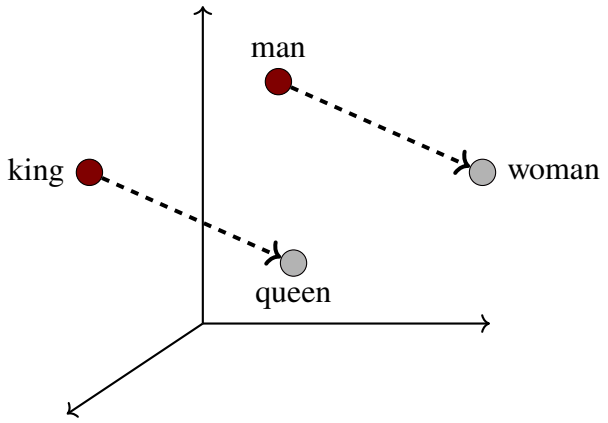

Figure 2.1: An illustration of the classic Word2Vec vector analogy. Analogies like these indicate the geometry of the representation space encodes meaningful semantic information. In particular, this analogy implies the existence of a *direction* in the embedding space that encodes gender.

transformed into a nicely manipulable algebraic object inside a larger space whose geometry meaningfully encodes semantic relationships.

The earliest neural language models date to 2003 with Bengio et al.'s seminal work learning distributed representations of words with neural networks (*9*). While this work provided the conceptual breakthrough that would eventually bring us to the neural age, it was intractable at the time.

It was not until a decade later in 2013 when Mikolov's Word2Vec (*10*) would successfully scale the neural approach to training over billion-token scale corpora, approaching the order of magnitude today's language models train over. Beyond demonstrating the scalability of the neural approach, Mikolov et al. demonstrated the remarkable algebraic and geometric properties learned embeddings possess. Most notably, the famous word analogy

$$\textsc{King} - \textsc{Man} + \textsc{Woman} \approx \textsc{Queen}$$

is a remarkable example of this property, pointing to the existence of a wealth of human recognizable, abstract concepts encoded in the *geometry* of representations

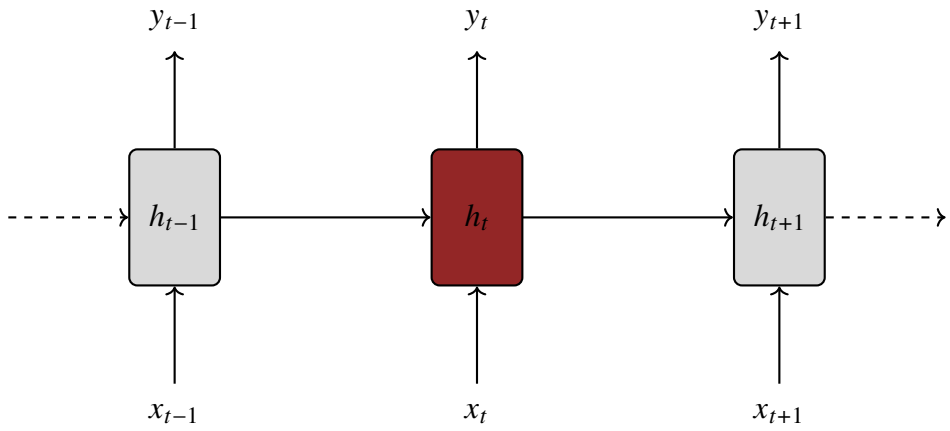

Figure 2.2: The RNN architecture. RNN computation cannot be parallelized due to *recurrence*: each hidden state h_t depends on the previous state h_{t-1}. While more complicated, LSTMs share this recurrence.

[1].

2.1.2 Attention

Having proved the viability of neural networks for language tasks, Word2Vec kick-started the neural age in earnest. The recurring thread of increased access to computational resources popularized more complicated neural architectures. In particular, Recurrent Neural Networks (RNNs) (*12*) and Long Short-Term Memory Networks (LSTMs) (*13*) served as very popular platforms for a variety of natural language tasks, processing text as a *recurrent sequence* of representations.

While the compute available was certainly enough to make these models feasible, the inability to parallelize computation (owing to the recurrent processing design of RNNs and LSTMs) prevented further scaling, leading to an eventual plateau in performance. Even beyond the parallelization bottleneck, RNNs and LSTMs suffered from a fundamental representational limitation. In the dominant *encoder-decoder* (or *sequence-to-sequence*) framework (*14*, *15*), an input sequence is processed token-by-token by an encoder network, which must compress the entire

[1] Indeed, in modern LLMs this phenomenon very much still persists, resulting in the formulation of what is now called the *Linear Representation Hypothesis* (*11*), which postulates that abstract concepts exist as linear directions in a model's representation space.

input into a single fixed-dimensional hidden state often called the *context vector*. The decoder then generates the output sequence conditioned solely on this vector. For short sequences, this design works well. However, as sequence length grows, the fixed-size context vector becomes an information bottleneck: the encoder must pack an increasingly large amount of information into a representation of fixed capacity, inevitably losing fine-grained detail about earlier tokens.

The solution to this representational bottleneck was the introduction of *attention* by Bahdanau et al. in 2014 (*16*). In the spirit of Geometric Deep Learning, we can think of attention as encoding an *inductive bias*[2] about how language is processed. That is, hypothesizing about how we communicate and understand language, it is clear that the information we convey does not depend uniformly across all the words in a sentence; certain parts are more important than others. For example, consider the sentences:

"The animal didn't cross the street because it was too tired." (2.1)

"The animal didn't cross the street because it was too wide." (2.2)

In both, we understand that an animal is not crossing the street for some reason. However, swapping "tired" for "wide" changes entirely the way we process and thus understand the sentence. In (2.1), we understand that tiredness is a condition experienced by living things and therefore understand that "it" refers to the animal. In (2.2), however, our interpretation is different: we understand that this wideness is probably attributed to the road and so confidently resolve "it" to refer to the street rather than the animal.

Through this example, we see that the way in which words interact influences interpretation and meaning *dynamically*. This is precisely what attention encodes. By reformulating a representation to be a dynamic, weighted sum of the parts of

[2]To be more precise, an inductive bias generally *constrains* a function class, whereas attention expands representational capacity. In that sense, is not quite an inductive bias, but rather a reasonable prior on how we wish representations to interact.

sentence, attention is able to model the way in which context shapes the way we process language.

Formally, suppose that at time t we have a sequence of T tokens $\{x_i\}^T$ that we wish to use to decode token x_{T+1}. Further, suppose that each token x_i has a hidden state (vector representation) h_i. Attention produces *context vector*

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i$$

that is a weighted combination of all the hidden states. Crucially, the attention weights $\alpha_{t,i}$ are given by a *learned function* of some notion of compatibility between each hidden state and the model's current state. Intuitively, this captures the example we presented above, allowing the model to process language in a dynamic way depending on context.

Attention rapidly became a ubiquitous component in neural NLP. Luong et al. (*17*) explored several variants of the compatibility function, establishing dot-product attention as a modular mechanism that could be integrated into a variety of architectures. Importantly, the success of attention revealed a broader principle: explicitly allowing a model to *selectively route information* based on content, rather than relying on fixed computational paths, leads to more expressive and scalable representations. This principle would prove to be far more consequential than its initial role as an augmentation to recurrent networks.

2.1.3 The Transformer

After overcoming the representational bottleneck, the next natural question was to ask whether recurrence was necessary. This brought us to the *Transformer* (*18*), the now ubiquitous architecture that powers modern LLMs. As the title "Attention is All You Need" of the original paper implies, the Transformer dispensed with recurrence entirely, instead relying *solely* on attention as its core sequence-processing mechanism.

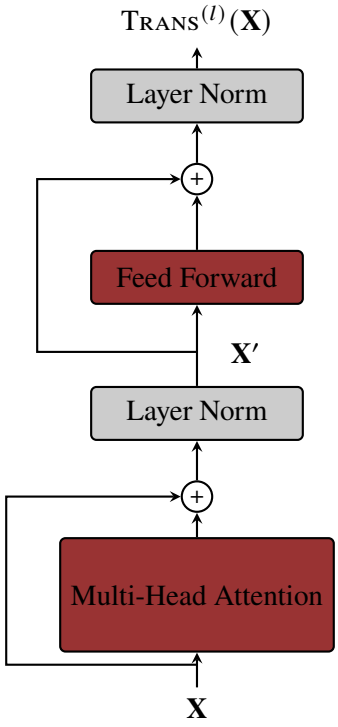

Figure 2.3: The l^{th} Transformer layer $\textsc{Trans}^{(l)}$

At its core, the Transformer can be thought of as being composed of L Transformer layers $\textsc{Trans}^{(l)}$, each given by the composition

$$\textsc{Trans}^{(l)}(\mathbf{X}) = \text{LN}(\mathbf{X}' + \text{FFN}(\mathbf{X}')), \quad \mathbf{X}' = \text{LN}(\mathbf{X} + \text{MHA}(\mathbf{X})) \tag{2.3}$$

where $\mathbf{X} \in \mathbb{R}^{T \times d}$ is the matrix of T token representations of dimension d output by the previous layer (or by initial embedding matrix $\textsc{Emb}$ at $l = 1$), MHA is *multi-headed self-attention*, FFN is a position-wise feed-forward network, and LN denotes layer normalization. We now describe each of these components beginning with self-attention, the basic computational component of multi-head attention.

Self-Attention

In Section 2.1.2, we introduced attention as a mechanism for computing a context-dependent, weighted combination of hidden states. In the encoder–decoder frame-

work, this allowed the decoder to selectively attend to relevant parts of the encoder's output. The Transformer's key conceptual leap was to apply this mechanism *within* a single sequence, enabling each token to attend to every other token in the same input. This is known as *self-attention*.

To operationalize this, the Transformer reformulates attention in terms of three learned projections: *queries*, *keys*, and *values*. Suppose we have a sequence of T token representations $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^\top \in \mathbb{R}^{T \times d}$. We project these into queries, keys, and values via learned weight matrices:

$$Q = \mathbf{X}W_Q, \quad K = \mathbf{X}W_K, \quad V = \mathbf{X}W_V \tag{2.4}$$

where $W_Q, W_K \in \mathbb{R}^{d \times d_k}$ and $W_V \in \mathbb{R}^{d \times d_v}$. The intuition behind this formulation is natural: a query represents what a token is "looking for", a key represents what a token "offers", and a value represents the information a token carries. Self-attention then computes

$$\textsc{Attention}(Q, K, V) = \textsc{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.5}$$

where the softmax is applied row-wise. Notice the similarity to the formulation from Section 2.1.2; here too, we compute a weighted sum of representations (the values), where the weights are determined by a compatibility function (the dot product between queries and keys). The crucial difference is that the queries, keys, and values are all derived from the *same* input sequence, allowing every token to attend to every other.

One subtlety deserves comment. The dot product $QK^\top$ is scaled by $\frac{1}{\sqrt{d_k}}$. This normalization is necessary because as d_k grows, the expected magnitude of the dot products grows proportionally, pushing the softmax into regions of extremely small gradients. By rescaling, we ensure that the attention weights remain well-behaved regardless of the projection dimension.

Multi-Head Attention

A single application of attention computes one set of weights and aggregates information accordingly. However, language is rich with structure at many levels: syntactic dependencies, semantic roles, coreference relations, and more. A single attention distribution may not be able to capture all of these relationships simultaneously.

Multi-head attention addresses this by running H independent attention operations in parallel, each with its own set of projections:

$$\text{HEAD}_i = \text{ATTENTION}(Q_i, K_i, V_i), \quad Q_i = \mathbf{X}W_{Q_i}, \quad K_i = \mathbf{X}W_{K_i}, \quad V_i = \mathbf{X}W_{V_i} \tag{2.6}$$

where $W_{Q_i}, W_{K_i} \in \mathbb{R}^{d \times d_k}$ and $W_{V_i} \in \mathbb{R}^{d \times d_v}$, with $d_k = d_v = d/H$. The outputs of all heads are then concatenated and projected back to dimension d:

$$\text{MHA}(\mathbf{X}) = \text{CONCAT}(\text{HEAD}_1, \ldots, \text{HEAD}_H)\, W_O \tag{2.7}$$

where $W_O \in \mathbb{R}^{d \times d}$ is a learned output projection. Each head is free to attend to different parts of the sequence in different ways, and the output projection learns how to integrate this information. In practice, different heads are indeed observed to learn qualitatively distinct attention patterns. Some track syntactic dependencies, others attend to adjacent tokens, and others capture longer-range relationships (*19*). This mirrors the linguistic intuition that understanding a sentence requires simultaneously tracking many types of relationships present in that sentence.

Position-Wise Feed-Forward Network

Following multi-head attention, each token representation is independently processed by a feed-forward network:

$$\mathrm{FFN}(\mathbf{x}) = W_2\,\sigma(W_1\mathbf{x} + b_1) + b_2 \tag{2.8}$$

where $W_1 \in \mathbb{R}^{d \times d_{\mathrm{ff}}}$, $W_2 \in \mathbb{R}^{d_{\mathrm{ff}} \times d}$, and σ is a nonlinearity. This is applied identically and independently to each position in the sequence (hence "position-wise"), and can be thought of as a per-token, learned transformation that processes the information aggregated by attention. Typically, $d_{\mathrm{ff}} = 4d$, giving the network a substantial expansion in which to perform nonlinear computation before projecting back to d dimensions.

Residual Connections and Layer Normalization

Returning to equation (2.3), two additional components complete the definition of a Transformer layer. The first is the *residual connection*, visible in the additive terms $\mathbf{X} + \mathrm{MHA}(\mathbf{X})$ and $\mathbf{X}' + \mathrm{FFN}(\mathbf{X}')$. By adding the input of each sub-layer to its output, residual connections allow gradients to flow directly through the network, making it feasible to train architectures with many layers (*20*). One can think of each subsequent Transformer layer as learning a *refinement* to the current representation rather than an entirely new one.

The second is *layer normalization* (LN) (*21*), which normalizes each token's representation to have zero mean and unit variance across the feature dimension. This stabilizes the scale of activations throughout the network and is essential for reliable training at depth.[3]

Positional Encoding

Having dispensed with recurrence entirely, the Transformer has no inherent notion of the order in which tokens appear. However, order is clearly essential to language:

[3]The placement of layer normalization relative to the sub-layer varies across implementations. The original Transformer (*18*) 2017 applies LN after the residual connection ("Post-LN"), as shown in equation (2.3). Much of the subsequent literature instead applies LN before the sub-layer ("Pre-LN") (*22*), which has been shown to improve training stability. Most modern LLMs use Pre-LN or variants such as RMSNorm(*23*)

"the cat sat on the mat" and "the mat sat on the cat" contain exactly the same tokens but very different meanings. To resolve this, the Transformer injects positional information by adding a *positional encoding* to the initial token embeddings.

Letting Λ denote the vocabulary of tokens, the initial embedding layer $\textsc{Emb} : \Lambda \to \mathbb{R}^d$ maps tokens to their initial representations. The input to the first Transformer layer is not $\textsc{Emb}(x)$ alone but rather $\textsc{Emb}(x) + \mathrm{PE}$, where $\mathrm{PE} \in \mathbb{R}^{T \times d}$ encodes the position of each token in the sequence. The original Transformer defines these encodings using sinusoidal functions of varying frequency:

$$\mathrm{PE}_{t,2i} = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \mathrm{PE}_{t,2i+1} = \cos\left(\frac{t}{10000^{2i/d}}\right) \tag{2.9}$$

where t is the position and i indexes the dimension. The key property of this scheme is that for any fixed offset k, the encoding at position $t + k$ can be expressed as a linear function of the encoding at position t, allowing the model to easily learn to attend to relative positions.[4]

The Encoder-Decoder Architecture

With these components defined, we can describe the full Transformer architecture as originally proposed (*18*). The model consists of an *encoder* and a *decoder*, each being a stack of Transformer layers.

The encoder processes the input sequence $x = (x_1, \ldots, x_T)$ to produce contextualized representations:

$$\textsc{Enc}(x) = \textsc{Trans}_{\text{ENC}}^{(L)} \circ \cdots \circ \textsc{Trans}_{\text{ENC}}^{(1)} \circ (\textsc{Emb}(x) + \mathrm{PE}) \tag{2.10}$$

The decoder generates the output sequence autoregressively, one token at a time. At each step t, it produces a probability distribution over the next token conditioned

[4]Modern Transformers often use more recent and effective positional encodings like *Rotary Positional Embeddings (RoPE)* (24)

on the tokens generated so far and the encoder's output. Crucially, the decoder's self-attention is *masked* so that position t can only attend to positions $\leq t$, preserving the autoregressive property. In addition, the decoder contains *cross-attention* layers in which the queries come from the decoder's representations but the keys and values come from the encoder's output, allowing the decoder to selectively draw on the input sequence at each generation step.

2.1.4 The Scaling Age

While the original encoder-decoder Transformer was designed for sequence-to-sequence tasks like translation, subsequent work demonstrated that the removing the encoder and building decoder-only models resulted in a remarkably versatile architecture. This is the design underlying the GPT family of models (*25*, *26*) and, indeed, most modern LLMs including the Qwen and Gemma models we use as base systems in Chapter 4.

The decoder-only design enabled self-supervised pre-training *at scale*, defining what has become the dominant paradigm in NLP. By training on the simple objective of next-token prediction over massive text corpora, these models learn rich, general-purpose representations of language that transfer to virtually any downstream task. The Transformer's support for parallel computation was the critical enabler. It allowed us to distribute training across thousands of GPUs, scaling systems to models with hundreds of billions of parameters trained on datasets approaching the scale of the web.

It is this conceptually simple architecture and training process that leaves us today in the age of Large Language Models (LLMs). The results have been extraordinary. Systems like GPT-4 (*27*) and their contemporaries exhibit remarkable capabilities across translation, reasoning, code generation, and many other tasks, driven almost entirely by the combination of the decoder-only Transformer architecture and extreme scale. However, it is precisely this success that brings us back

to the motivating questions of this thesis.

The Limits of Scale; or, Towards a Geometric Perspective

As previewed in the introduction, while scale has been transformative, it is not alone the answer. The curse of dimensionality tells us that the data required to learn a function grows exponentially with the dimension of the space in which it operates. And yet, models operating in spaces of dimension d in the tens of thousands learn representations that generalize remarkably well. As we argued, the resolution to this paradox lies in *structure*: real data occupy low-dimensional manifolds within the ambient space, and models succeed to the extent that they exploit this structure, whether explicitly by design or implicitly through sufficient data.

While the Transformer took attention and pushed it to its limits, it still leaves significant structure on the table. In particular, it treats its input as an unstructured, flat sequence of tokens, discarding any hierarchical, relational, or any other syntactic structure that might be present in language, forcing the model to learn these relationships from data. For tasks like machine translation, where *meaning* must be preserved across languages, this means the model must *rediscover* cross-lingual semantic regularities entirely from data—a feat that requires enormous amounts of parallel text and computation, and that may never fully succeed for low-resource language pairs. In this thesis, we consider how to address this issue through the geometric perspective.

2.2 Geometric Deep Learning

This discussion of explicitly modeled structure (or lack thereof) brings us to Geometric Deep Learning (GDL) (*4*), the relatively new perspective that seeks to unify and advance machine learning through the lens of structure. Rather than relying on scale to *implicitly* discover latent structure, GDL accounts for explicitly by identify-

ing the *symmetries* of a problem and building models that respect this symmetry *by construction*. To properly introduce this perspective, a few definitions are in order.

2.2.1 Groups, Actions, and Predictions

Definition. A **group** $(G, *)$ is a set G equipped with a binary operation $* : G \times G \to G$ that obeys the following properties:

1. (*Closure*) If $g, h \in G$ then $g * h \in G$
2. (*Associativity*) For $g, h, l \in G$,
$$(g * h) * l = g * (h * l)$$
3. (*Identity*) There is a special element $e \in G$ satisfying
$$e * g = g = g * e$$
for all $g \in G$.
4. (*Inverse*) For each element $g \in G$ there is an element $g^{-1} \in G$ such that
$$g g^{-1} = e = g^{-1} g$$

It is important to note that commutativity is not required so that, in general, $g * h \neq h * g$. Groups that satisfy this property are called *abelian*. From now on, we will refer to $(G, *)$ simply by G, and write gh to mean $g * h$.

Groups are very simple and yet extraordinarily fundamental objects. One particularly nice way to introduce them is as the set of symmetries of an object. Consider the humble square. What transformations can we apply that leave it unchanged? We can rotate it by any multiple of 90°. We can also flip it along the vertical axis. Of course, we can also do nothing to it. In fact, we can compose any number of these transformations together and still get the same square back. These transformations,

together with composition, precisely make up the group D_8: the dihedral group of order 8.

The following notion connects abstract groups to the concrete data we wish to model.

Definition. Given a set A and group G, an **action** on A is a map $\rho : G \times A \to A$ written $\rho(g, a) = g.a$ satisfying

1. (*Identity*) If e is the identity in G, then $e.a = a$ for all $a \in A$
2. (*Compatibility*) for all pairs $g_1, g_2 \in G$ and elements $a \in A$,

$$g_1.(g_2.a) = (g_1 g_2).a$$

Returning to the square, the dihedral group D_8 acts on the set of vertex configurations of the square. More generally, symmetry groups arise naturally whenever we can identify transformations of our data that should not affect our predictions. The following definition formalizes this notion of a symmetry-respecting transformation which is central to GDL.

Definition. Suppose G acts on X and Y and let $f : X \to Y$. We say f is

1. $G-$*invariant* if $f(g.x) = f(x)$ for all $x \in X$ and $g \in G$
2. $G-$*equivariant* if $f(g.x) = g.f(x)$ for all $x \in X$ and $g \in G$

These two definitions are similar, but their distinction is important. Intuitively, a function is $G-$invariant if transforming the input leaves the output unchanged, whereas a function is $G-$equivariant if transforming the input transforms the output in an analogous way.

Already, we see the GDL perspective taking place. By identifying symmetry groups that act on our data and constructing modules that respect these symmetries through invariance or equivariance, we can operationalization our desire to build

models that respect the underlying structure (that is, the symmetries) of our data. This brings us back to the concrete example given in the introduction of predicting the acidity of a molecule. *A priori*, we know that rotating any molecule does not affect its acidity. We also know that the special Euclidean group $SE(3)$ describes these transformations. Thus, we know that a robust acidity prediction model should be $SE(3)-$Invariant. On the other hand, a function that predicts the position of each atom after a force is applied should be *equivariant*; rather than leaving position unchanged, rotating the molecule should rotate the predicted positions accordingly.

With these definitions in place, the GDL approach to architecture design can be stated concisely:

1. **Identify** the symmetry group G that characterizes the domain.
2. **Design** the function class $\mathcal{F}$ so that every $f \in \mathcal{F}$ is G-invariant (for global predictions) or G-equivariant (for local/per-element predictions).
3. **Compose** equivariant layers to build deep networks, noting that the composition of equivariant functions is itself equivariant.

By restricting the function class to respect the known symmetry, we dramatically reduce the effective hypothesis space. The result is improved *sample efficiency* and *generalization*, both of which have been demonstrated extensively in practice (*4*, *28*, *29*).

2.2.2 The Graph Neural Network and GDL as a Unifying Perspective

As a perspective, the power of the GDL framework lies in its revealing of many well-known architectures as instances of the same blueprint, differing only in their choice of domain and symmetry group. We briefly give some examples, with particular attention to the graph neural network (GNN), which will appear centrally later.

Convolutional Neural Networks (CNNs).

Consider an image classification task. Translating the image by a few pixels should not change the classification. The relevant symmetry group is the group of discrete translations on a 2D grid. CNNs, which include layers consisting of convolutional kernels, are precisely the class of neural networks that are equivariant to this group: the convolution operation commutes with translation. A convolutional layer maps a translated input to a translated feature map, and invariance for classification is achieved by a final global pooling layer. This connection was formalized by Cohen and Welling (*28*), who generalized CNNs to arbitrary groups.

Message-Passing Graph Neural Networks (MP-GNNs)

A graph $G = (V, E)$ is a set of nodes V connected with edges E. Graphs are arguably amongst the simplest mathematical objects and, perhaps because of this, they are extraordinarily powerful tools for describing any number of entities and the relationships between them. As we shall see in section 2.3, graphs will be our bridge between GDL and NLP.

Consider the task of producing a vector representation of a graph via some embedding model $f : G \to \mathbb{R}^d$. Given a graph $G = (V, E)$, we can represent it via an adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$ without any loss of information. However, this presentation implicitly assigns an arbitrary ordering to the nodes of the graph, which themselves are naturally unordered. That is, we can permute A with a permutation matrix P and have an equivalent presentation $A' = PAP^T$ of our graph G. Here we see our problem. For general f, we would have that $f(A) \neq f(A')$, a condition we clearly do not want to hold. Thus, we require the f be *permutation-invariant*.

The solution to creating such an f is the formulation of *message passing*. The core idea is that we view embedding as an iterative process, where at each step i we compute an embedding $h_v^{(i+1)}$ for each node $v \in V$ by computing a "message" $m_v^{(i)}$

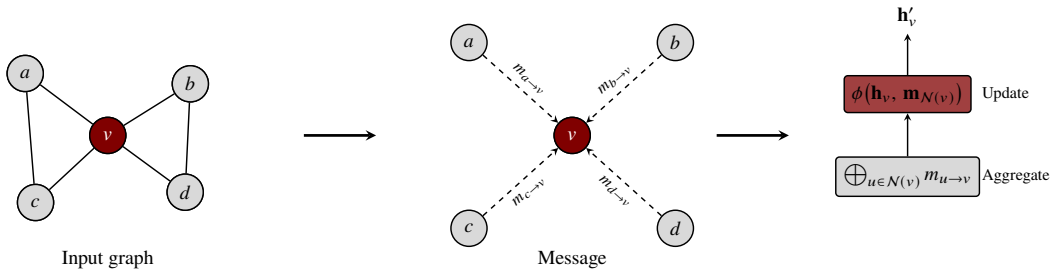

Figure 2.4: The message passing formulation. An embedding for node v is calculating by passing and aggregating messages from its neighbors a, b, c, d.

based on the embeddings $h_u^{(i)}$ of v's neighbors $u \in \mathcal{N}(v)$ and combining it with v's previous embedding $h_v^{(i)}$. Symbolically,

$$m_v^{(i)} = \textsc{Aggregate}^{(i)}(\{h_u^{(i)} \mid u \in \mathcal{N}(v)\})$$
$$h_v^{(i+1)} = \textsc{Update}^{(i)}(h_v^{(i)}, m_v^{(i)})$$

Where $\textsc{Aggregate}^{(i)}$ and $\textsc{Update}^{(i)}$ are functions parameterized by neural networks, and the superscripts emphasize our ability to define different functions at different layers (iterations). This also presents a natural way to incorporate node features $X \in \mathbb{R}^{|V|\times d}$ in our embedding by initializing $h_v^{(0)} = X_v$, allowing us to greatly boost the expressivity and specificity of our resulting embeddings.

Notice that since the message is a set function, message passing GNNs are permutation-*equivariant* by design, i.e. that $f(PAP^T) = Pf(A) \in \mathbb{R}^{|V|\times d}$, meaning that embedding a permuted adjacency matrix is equivalent to permuting the resulting embeddings. Further, they can be made *invariant* by simply pooling together the individual node embeddings into a single vector graph embedding. In both cases, we achieve our desired outcome of not having node order affect our embedding in any meaningful way.

Transformers

Perhaps surprisingly, the Transformer architecture from Section 2.1.3 also fits within the GDL framework. Set Attention (the Transformer's attention mechanism without

positional encodings) is permutation equivariant: it treats the input as an unordered set and permuting the input permutes the output in the same way. The addition of positional encodings breaks this symmetry, effectively informing the model that the input has sequential structure. This can be viewed as a deliberate “symmetry breaking” step that introduces an inductive bias appropriate for sequential data.

Approximate symmetries

In many real-world domains, symmetries hold only approximately. Physical systems may exhibit near-symmetries that are broken by small perturbations. For example, consider a ball resting on the top of a hill. As time progresses, the potential energy of the ball remains constant. However, treating weather as a stochastic process, consider a sudden burst of wind that sends the ball rolling down the hill. Such a perturbation is an example of spontaneous symmetry breaking. From the GDL perspective, one approach to modeling such approximate symmetries is through "soft" constraints like regularization, and the question of approximate symmetries is very much an active research area.

As we shall argue in Chapter 3, translation is in reality only an approximate symmetry of natural language. However, our approach does *not* align with the typical approaches to approximate symmetries. Rather, our approach will be to treat our symmetry as an exact one, defining a system that, modulo data quality, enforces invariance exactly via input canonicalization.

2.3 The Abstract Meaning Representation

In order give a proper treatment of translation as a symmetry of language, it is necessary to prescribe some structure to language to make precise what we mean by *meaning*. To this end, we make use of the linguistic formalism known as the *Abstract Meaning Representation*.

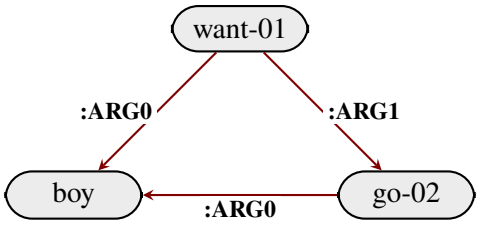

Figure 2.5: AMR graph for the sentence "The boy wants to go".

Introduced by Banarescu et al., the Abstract Meaning Representation (AMR) (*30*) is a linguistic formalism that models the *meaning* of an utterance as a directed, acyclic graph. In an AMR graph, nodes represent *concepts* (entities, events, properties, and other semantic units), while labeled, directed edges represent the *relations* between those concepts. The result is a structured representation of *who did what to whom*, capturing the underlying *meaning* of an utterance in a way that abstracts away syntax and surface form.

As an example, consider the sentence "The boy wants to go." Its AMR graph has a root node `want-01` corresponding to the wanting event. This node has two outgoing edges: an `ARG0` edge pointing to a `boy` node (the entity that wants), and an `ARG1` node pointing to a `go-02` node (the thing that is wanted). In turn, the `go-02` node has its own `ARG0` edge pointing back to the boy, encoding the fact that the boy is the entity that will go. See Figure 2.5 for an illustration.

Several AMR properties are worth emphasizing. First, notice that AMRs abstract away from syntax entirely, presenting meaning in the simplest and most abstract form possible. This is by construction: the sentences "The boy wants to go", "The boy desires to leave", "Leaving is what the boy wants to do", etc. all mean the same thing and would therefore all map to the same (or very similar) AMR graph(s). Second, AMR concepts are drawn from a fixed ontology of concepts. Events are represented using PropBank framesets (*31*), which define standardized argument structures for predicates and their senses (e.g., `go-02` is a specific sense of the verb “to go"). This provides a consistent, fixed vocabulary of relations and concepts used across

different sentences. Third, the set of relations (edge labels) is relatively small. The most common are the numbered `ARG` roles corresponding to the arguments of a predicate as defined by its PropBank frame, but other general relations like `location, time, manner`, and `polarity` exist, among others.

While AMRs are fundamentally graphs, they are commonly presented as strings using PENMAN notation (*32*), a human-readable format that presents the graph via a rooted traversal. That is, PENMAN chooses a root node and then follows a depth-first traversal order to serialize the graph. For example, the AMR for "The boy wants to go" is written as

```
(w / want-01
   :ARG0 (b / boy)
   :ARG1 (g / go-02
            :ARG0 b))
```

where the letters `w,b,g` are variable names and the '/' character separates variables from their respective concepts. This provides a natural way to interface AMRs with text-based systems like LLMs, as we shall see in our survey of related work.

It is important to note that neither an AMR graph nor its PENMAN serialization are necessarily unique. For graphs in particular, which concept is to be used as the root node is a choice made by the annotator. While it is advised to be the central predicate in the sentence, there is some ambiguity. In practice, however, the set of valid graphs describing a sentence are extremely similar, placating this issue.

2.3.1 Parsing AMRs

Parsing natural language into AMR graphs is itself a challenging and unsolved tasks. Human-parsed AMRs are expensive and time-consuming to create, making

the development of reliable and accurate parsing machines critical for machine learning applications.

Early AMR parsers were largely rule-based or relied on transition-based algorithms (*33*), but the field has since shifted to neural approaches treating AMR parsing as a sequence-to-sequence prediction problem. The state-of-the-art parsers belong to the StructBART family (*34*) which modify the BART pre-trained sequence-to-sequence language model to generate AMRs from text. Crucially, this family of parsers supports multilingual parsing from German, Spanish, Italian, and Chinese into AMR, which we shall see is essential for our purposes. It is important to note that parsing from non-English languages degrades performance. However, the StructBART family still provides very strong parsing ability with an accuracy score of 82.9% for multilingual models and 85.9% for the English model, giving us confidence that parsing systems are sufficiently robust to be used at scale.

2.3.2 As an Interlingua

Continuing the thread of multilingual parsing, it is essential we emphasize that AMRs were constructed to model English *specifically*. Many linguistic phenomena that appear in non-English languages are difficult or impossible for AMRs to capture. Gendered nouns in Romance languages, for instance, carry semantic information that AMRs cannot naturally encode.

Despite these limitations, there has been a significant amount of research investigating using AMRs across languages; that is, as an *interlingua*. Xue et al. (*35*) showed early on that English AMRs often overlapped with their Czech and Chinese counterparts, laying the foundation for future work considering English AMRs as a unified semantic space. Damonte and Cohen (*36*) pushed this further, showing that effective parsers could be trained to overcome the structural difference between English and Italian, German, Spanish, and Chinese, producing meaningful English AMRs for all of these languages.

These foundations have spawned several branches of multilingual AMR research, including many works considering multilingual → AMR parsing and AMR → multilingual parsing. Wein and Schneider (*37*) present one of the most complete syntheses so far, demonstrating that while AMR does not extend perfectly to other languages, it can still be very useful cross-lingually. Following this, our ability to reliably map sentences into a shared semantic space using AMRs will be a central component to this thesis and motivates our use of this system.

2.4 AMR Applications and Related Work

Having introduced the AMR formalism, we now turn surveying its role in the broader NLP landscape. AMRs have been applied to a wide variety of tasks, and the question of how best to integrate the information they offer into neural models—particularly LLMs—is an active area of research with direct relevance to this work.

2.4.1 General Downstream Applications

As a structured meaning representation, AMRs have proved useful across a diverse set of NLP tasks. Liu et al. (*38*) were among the first to apply AMR to text summarization, proposing a system that creates a document-level graph and selects a summarizing subgraph. More recently, Qiu et al. (*39*) use AMRs to generate robust and coherent factually inconsistent summaries to improve summarization evaluation. In information extraction, Zhang and Ji (*40*) demonstrated that AMR-derived knowledge graphs can significantly improve biomedical event extraction. For a comprehensive overview of downstream applications, we direct the reader to the survey by Wein and Optiz (*41*) which reviews over 100 papers employing AMRs for various downstream tasks.

2.4.2 *Integrating AMRs with LLMs*

A recurring challenge in the works above is the *modality gap*: AMRs encode meaning as a graph, but modern language models process sequences of tokens. How to bridge this gap (and, in particular, how to preserve the structural information that make AMRs useful) has been a central question in recent work. The approaches taken can be roughly organized along a spectrum from purely text-based integration to fully graph-native methods.

The simplest strategy is to linearize the AMR graph into its PENMAN string and feed it directly to an LLM. Jin et al.(*42*) explore this idea, appending AMR representations to prompts and asking the model to reason over them in what they term graph-based chain-of-thought prompting. However, they find that naively linearizing graphs generally does not help and can even hurt performance. Yao et al. (*43*) slightly modify this approach by further augmenting the prompt with pseudocode for algorithmically interpreting AMRs, but find similar results. These findings are perhaps unsurprising, as it has been shown that while LLMs have a good handle on semantic information, this does not transfer to their ability to parse or generate AMRs (*44*).

Building from these negative results, subsequent approaches massaged linearized graph tokens in different ways to make linearization viable. Zhang et al.(*40*) converted AMRs into their natural language description and used this to augment original prompts, finding this approach did boost performance across a wide variety of tasks. This result suggests that it is not the *information* in AMRs that LLMs struggle with, but rather the *format* in which it is presented.

Most relevant to our work is the approach taken by Kamel et al. (*45*), who introduce SAFT (Structure-Aware Fine-Tuning), a method that injects graph topology into pre-trained LLMs without architectural changes. SAFT computes direction-sensitive positional encodings from the magnetic Laplacian(*46*) of a transformed AMR graph and projects them into the LLM's embedding space via a learned MLP,

adding structural inductive bias to the token representations during fine-tuning. Evaluated on AMR-to-text generation, SAFT achieves a new state-of-the-art score with gains scaling with graph complexity. This work confirms our motivating principle that explicitly encoding graph structure yields substantial improvements over linearization alone.

Traversing these results, we echo our motivating theme of the importance of *structure*: presenting AMRs in a purely linearized way discards their topology and results in poor performance, while reintroducing structural elements brings us to the state of the art. However, while Kamel's approach successfully exploits *positional* structure (using direction-sensitive spectral graph embeddings), it crucially overlooks another piece of graph structure which we address in Chapter 3: *permutation invariance*. That is, SAFT still operates over a linearized sequence of graph tokens, making it sensitive to the choice of traversal order. Our GNN-based approach, by contrast, directly embeds the graph using message-passing networks that are permutation-invariant by construction, more completely capturing the structure of the data. Further, while we also consider AMRs as an augmentation to LLMs, we differ significantly from Kamel's work not only in terms of the methods we use, but in our motivation and perspective and in the application setting we consider.

2.4.3 AMR-Augmented Machine Translation

The application of AMR to machine translation has a notable, if still relatively short, history. Song et al. (*47*) were the first to systematically study AMR's usefulness for neural machine translation. Working in the pre-Transformer era, they proposed a dual encoder architecture that processes both the source sentence and its AMR graph using graph recurrent networks, combining the two representations for English-to-German translation. Their results showed significant improvements over a strong attention-based sequence-to-sequence baseline, establishing that the explicit semantic structure provided by AMR is complementary to the information

already captured by neural encoders.

Li and Flanigan (*48*) updated this line of work for the Transformer era. They proposed a novel hybrid architecture that augments the standard encoder-decoder Transformer with a Heterogeneous Graph Transformer (*49*) to encode source-sentence AMR graphs. Their system combines the sequential representation produced by the standard Transformer encoder with the structural representation produced by the graph encoder, allowing the decoder to attend over both during generation. Crucially, they demonstrated improvements in both high-resource and low-resource settings across two language pairs, confirming that AMR-derived structure provides benefits even when parallel data is abundant and especially when it is scarce.

Our work differs from both Song et al. and Li and Flanigan in several important respects. First, rather than modifying the encoder architecture of an encoder-decoder translation model, we consider augmenting decoder-only modern LLMs and pursue a modular approach: a frozen GNN embedder produces a semantic embedding that is injected into the LLM via soft prompt tuning (Chapter 4). This design allows us to augment any pre-trained LLM without architectural changes, making the approach more practical and generalizable. Second, we frame the problem explicitly through the lens of Geometric Deep Learning, treating cross-lingual translation as a symmetry of meaning and designing our embeddings to be invariant under this symmetry. This perspective is, to our knowledge, novel in the context of AMR-augmented translation. Third, while previous work used AMR only on the source side of a single language pair, our use of multilingual AMR parsing allows us to produce translation-invariant representations across multiple languages, enabling evaluation across English paired with German, Spanish, Italian, and Chinese.

Now, with the relevant background and related work properly introduced, we continue to the description of our geometric approach to language modeling. Following the GDL design philosophy outlined in Section 2.2.1 and having identified translation as our symmetry, we will show how to design a class of functions that

is invariant to this (approximate) symmetry. Finally, in Chapter 4, we shall show the benefit of this approach by using our invariant function to augment LLM performance on the task most relevant to our symmetry.

Chapter 3

TRANSLATION-INVARIANT EMBEDDINGS

Having introduced the necessary background, we are now ready to discuss our approach to the central component of this thesis: considering a geometric approach to language modeling by treating cross-lingual translation as a *symmetry of language* and designing models to produce representations of language that are invariant to this symmetry.

What exactly do we mean by invariance under translation? Informally, this captures a very intuitive notion: translating a sentence preserves the meaning of that sentence. Therefore, the embeddings we learn should not change when the language the sentence is presented in changes.

Already, we can see how one might formalize the idea of embeddings being invariant under translation. One can imagine that there is a group $\{\text{English} \rightarrow \text{Spanish}, \text{Spanish} \rightarrow \text{English}, \dots\}$ that acts on sentences by translating them accordingly, and that an embedding model is invariant if it produces the same embedding for any sentence, regardless of the language it is presented in.

In reality, language is not this nice. Consider the sentences *te amo* and *te quiero* in Spanish. The commonly accepted translation for both of these in English is "I love you". However, in Spanish, these two sentences have different meanings in the sense that "*amo*" conveys a more intense (usually romantic) type of love than "*quiero*" does. Should we translate *te amo* to "I love you romantically" then? If so, why not "I love you intensely" instead? Is it even possible to preserve the meaning of a sentence when you translate it? What even is the "meaning" of a sentence? Unfortunately there are no clear answers to these questions. From this example, however, it is clear that while intuitively translation is very much a symmetry a meaning, in reality it is much better described as an *approximate* symmetry of meaning as, while successful

translation forgets some nuances of the original phrase, it certainly preserves the vast majority of the essential information conveyed.

The point here is that language is quite the tricky substrate to work with and resists formalization. However, while we may not be able to formally enforce translation-invariance, we can certainly approximate it in a meaningful way. The key observation here is that while it may be difficult to design our embedding function such that we are able to control its *output*, it is much easier to control its *input*. In fact, by *canonicalizing the input*, we can make any embedding model truly invariant. In this chapter, we will do exactly this to demonstrate how to construct representations of meaning that are invariant to cross-lingual translation.

3.1 Method

3.1.1 Input Canonicalization

Recall the Abstract Meaning Representation (AMR) introduced in chapter 2, which represents the "meaning" of a sentence as a directed graph. One very useful fact for us is that, in theory, modulo parser error and the limitations of AMRs discussed in Section 2.3, sentences with the same meaning map to the same AMR, *regardless of language*. By mapping sentences to their AMRs and using AMRs as a canonical input format, these graphs give us a clear path to translation invariance.

This approach is very intuitive and works well in practice, but it is important we recapitulate the shortcomings noted in Section 2.3. Perhaps the most pressing is the fact that AMRs are specifically designed with the structure of English in mind. Along these lines, parsing text to AMRs is a challenge, and parsing non-English text to AMRs is even more challenging. However, as we noted before, there has been extensive research demonstrating that AMRs can be used to effectively model other languages, and there are parsing systems that provide excellent text $\rightarrow$ AMR parsing performance across many language. To this end, we will be using the state-of-the-art

StructBART family of parsers, which will allow us to consider English, Spanish, Italian, German, and Chinese.

Thus, in line with the literature, our course of action will be to treat English AMRs as an *interlingua* and map sentences across languages to their AMR graph, producing sentence representations that are *translation invariant up to the performance of the parser*. To see this, suppose $x \in X$ is an input sentence, $f : X \to \mathbb{R}^d$ is a function that embeds sentences, and $\rho : X \to X$ is a function that translates x to some other language. When X is the space of sentences in string format, we are unable to control f and therefore have

$$f(\rho(x)) \neq f(x)$$

dashing our desire for invariance. However, when we move from X to A, the space of sentences in AMR form, the story is much different. Treating A as an interlingua, $x^* \in A$ corresponds to $x \in X$ *modulo language* so that ρ acts trivially on A. Thus for $f^* : A \to \mathbb{R}^d$, we have

$$f^*(\rho(x^*)) = f^*(x^*)$$

exactly enforcing translation invariance whenever the parser correctly sends $x \mapsto x^*$. When discussing our results in Chapter 5, we shall show that, in practice, the parsers we use perform well enough such that embeddings of parallel sentences are *highly* aligned and are meaningfully separated from non-parallel sentences.

Here, we take a moment to remark at the simplicity of our method. Formalizing language is *hard* and designing an invariant architecture is even harder. However, by shifting the problem from architecture design to input formatting, we sidestep these problems entirely.

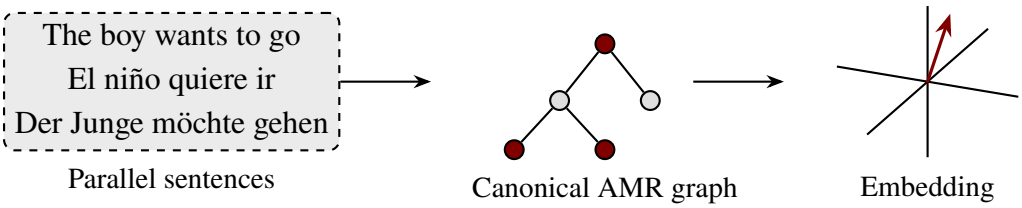

Figure 3.1: The input canonicalization process. Parallel sentences are sent to their canonical AMR graph and then embedded, producing a representation that is invariant to cross-lingual translation.

3.1.2 Embedding AMRs

Once we have canonicalized the input by transforming sentences into graphs, the next step is to embed these graphs to produce our translation-invariant embeddings. The main consideration here is that graphs are very structured objects. Thus, as introduced in Section 2.2.2, we use Message-Passing Graph Neural Networks (MP-GNNs) as our neural architecture.

GNN Architectures

As we have introduced it, the MP-GNN formulation is very general, and allows us to define many different architectures that all share the key permutation inductive bias we require. Here, we introduce the particular GNNs we consider.

In searching for the best architecture to embed AMR graphs, one key property to keep in mind is that in addition to labels describing the semantic role of each node, AMRs also have labels *on the edges* of the graph describing the semantic relationship between two nodes. However, the general MP-GNN formulation does *not* explicitly allow edge features. This motivates us to group architectures by their support for edge features and introduce the *graph lift* to accommodate architectures that lack support for edge features.

Edge Feature Support

◦ **Graph Isomorphism Network with Edge Features** (GINE): Originally intro-

duced without edge support as the Graph Isomorphism Network (GIN) in (*50*). The name comes from its connection to the k-Weisfieler-Lehman (k-WL) test, an algorithm which solves the fundamental problem of graph isomorphism testing for a broad class of graphs. Theorems by (*51*) and (*50*) prove that the general formulation of a k layer MP-GNN is at most as powerful as the $k-$WL test in terms of its representational capacity. Following this, GIN(E) is the minimal GNN with this capacity, defined by

$$h_v^{(i)} = \text{MLP}\left((1+\epsilon^{(i)})h_v^{(i-1)} + \sum_{u \in \mathcal{N}(v)} \sigma(h_u^{(i-1)} + \mathbf{e}_{vu})\right)$$

Where MLP is the standard multilayer perceptron, σ is an elementwise nonlinearity (the ReLU in our case), and $\mathbf{e}_{vu}$ is the edge feature between node v and u embedded into the appropriate dimension, and $\epsilon^{(k)}$ is a learnable scalar.

- **Graph Attention Network** (GAT): Introduced by Veličković(*52*), GAT is, as the name implies, inspired by *attention* (*16*) which has been instrumental in the success of recent advances in NLP. In particular, the message m_v is calculated by paying attention to the neighbors:

$$m_v = \sum_{u \in \mathcal{N}(v)} \alpha_{v,u} h_u$$

Where $\alpha_{v,u}$ is a weight capturing how much node v attends to node u. In the original paper, the weights are defined by

$$\alpha_{v,u} = \frac{\exp(a^T \sigma[W_s h_v + W_t h_u + W_e \mathbf{e}_{vu}])}{\sum_{v' \in \mathcal{N}(v) \cup \{v\}} \exp(a^T \sigma[W_s h_v + W_t h_{v'}])}$$

Where a is a trainable attention vector, W_s, W_t, W_e are trainable matrices (target, source, and edge projectors, respectively) and σ is a variant of the ReLU named the "Leaky ReLU".

- **GraphGPS**: Introduced by (*53*) to bridge Graph Transformers (which lack the permutation equivariance inductive bias) and MP-GNNs (which suffer from bottlenecks known as *over-smoothing* and *over-squashing* (*54*)) by combining them

into a hybrid architecture. Each GPS layer roughly has the following form:

$$\text{MLP} \circ \textsc{GlobalAttention} \circ \textsc{LocalMP-GNN}(\mathbf{X} + PE(\mathbf{X}))$$

Where $\mathbf{X}$ are the node features and $PE(\cdot)$ is a *positional encoding* which assigns some notion of position to each node (necessary for Transformers). In particular, we use random walk positional encodings (*55*), GINE for the MP-GNN (allowing us to utilize edge features), and a linear-complexity variant of attention for the global attention component.

No Edge Feature Support

- **Graph Convolutional Networks** (GCNs): Introduced by (*56*) have strong performance across tasks and are commonly used as a strong baseline GNN. They are defined by

$$h_v^{(i)} = \sigma\left(W^{(i)} \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{h_u}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)}}\right)$$

- **GraphSAGE**: Introduced by (*57*) is a framework that was designed for inductive reasoning on large graphs (producing embeddings for nodes not seen at training time). It is defined by

$$h_v^{(i)} = \sigma\left(W^{(i)} \cdot \textsc{Concat}(h_v^{(i-1)}, m_v^{(i)})\right)$$

  This framework supports many permutation invariant $\textsc{Aggregate}$ functions to produce m_v. We in particular use mean aggregation: $m_v = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} h_u$

Graph Directionality

One detail we have glossed over until now is the fact that the standard MP-GNN formulation treats graphs as *undirected* when passing messages. This is an issue because the AMR graphs we seek to embed are explicitly *directed*. Modeling directionality with GNNs is a topic that has been addressed extensively in the

literature. The most promising and compatible solution is Rossi et al.'s formulation of the Directed GNN (Dir-GNN) (*58*).

As formulated, Dir-GNN is a very natural generalization of the standard MP-GNN framework, allowing *any* GNN architecture to be easily modified to become a Dir-GNN. Recall that in the standard MP-GNN framework, the embedding of a node v is calculated as the aggregation of the node's previous state and a "message" computed from the states of all its neighbors. At a high level, Dir-GNN simply splits the message computed from *all* the neighbors of v into two messages: one from outgoing neighbors of v and one from incoming neighbors.

Formally, let $v \in V$ be a node in the graph and consider incoming edges $(u,v) \in E$ and outgoing edges $(v,u) \in E$. At iteration i, Dir-GNN computes the $i+1$th embedding $h_v^{(i+1)}$ as

$$
\begin{aligned}
m_{v,\leftarrow}^{(i)} &= \textsc{Aggregate}^{(i)}(\{h_u^{(i)} \mid (u,v) \in E\}) \\
m_{v,\rightarrow}^{(i)} &= \textsc{Aggregate}^{(i)}(\{h_u^{(i)} \mid (v,u) \in E\}) \\
h_v^{(i+1)} &= \textsc{Update}^{(i)}(h_v^{(i)}, m_{v,\leftarrow}^{(i)}, m_{v,\rightarrow}^{(i)})
\end{aligned}
$$

Notice the similarities to the MP-GNN formulation in Section 2.2.2; all Dir-GNN does is split the message $m_v^{(i)}$ into incoming messages $m_{v,\leftarrow}^{(i)}$ and outgoing messages $m_{v,\rightarrow}^{(i)}$. Further, notice that this formulation allows for the outgoing and incoming messages to be given different weight. In practice, we weight both directions equally.

The Graph Lift

As mentioned previously, edge labels are crucial components of AMRs. As such, we need some way of making architectures that do not support edge features compatible with AMRs. Our solution is to introduce an operation we refer to as a "graph lift", which converts multirelational graphs (graphs with edge features) into standard graphs by "lifting" edge relations into nodes.

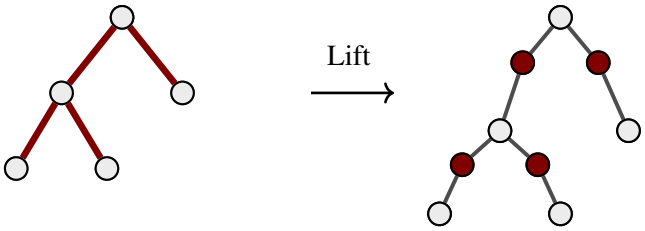

Figure 3.2: An illustration of the graph "lift", lifting edge features (in red) to nodes.

Formally, we define a multirelational graph $M = (V, E_R)$ where V is a set of nodes and E_R is a set of relational edges: i.e., if there is an edge of type τ between $u, v \in V$, then $(u, \tau, v) \in E_R$. Define $\textsc{lift} : (V, E_R) \to (V', E)$ by adding τ to the set of nodes V' and adding $(u, \tau), (\tau, v)$ to the set of edges E for every multirelational edge $(u, \tau, v) \in E_R$. See figure 3.2 for an illustration of the process.

With this operation, we can create "lifted" AMRs that contain the same information as the standard, multirelational version, but that are compatible with all GNNs.

Graph Tokenization

Before moving on to describe how AMR embeddings are trained, we describe an important step of the process we have glossed over until now: *tokenization*. In the case of AMR graphs, the node and edge features we have been discussing are *strings*, corresponding to the semantic role of a node and the semantic relationship between nodes, respectively. However, to use them as inputs to a GNN, node and edge features need to be numeric vectors.

Since there is a finite and predetermined vocabulary from which these labels come from, it makes sense to associate each label to some arbitrary, unique integer ID that is then embedded into $\mathbb{R}^d$ by a randomly initialized embedding matrix. This process is referred to as *tokenization*, and is the same way by which we transform text into sequences of tokens that language models can process.

3.1.3 Training Protocol

Now, with the architectures and data settled, we turn to the objective used to train the models. Here, we recall our goal of producing low-dimensional embeddings that meaningfully represent the "meaning" of a sentence in any language, as determined by its AMR graph. While the architecture encodes our inductive bias (permutation equivariance), the training objective is what determines the actual information that our embeddings will encode.

In deciding on a training objective, we must make precise what exactly we mean by a "meaningful representation". While our input canonicalization strategy enforces invariance $f(\rho(x)) = f(x)$, it does not enforce *separation* $f(x) = f(y) \iff x = y$. Concretely, we need an objective that pressures the function we learn to be injective to avoid representational collapse. While objectives like graph reconstruction or masked auto-encoding (*59*) may seem like natural choices, in our experiments we found these objectives led to severe representational collapse, leading to a lack of separability as embeddings were clustered is a low-dimensional subspace. Instead, we need an objective that explicitly optimizes for separation between different classes of objects. For this, we turn to the contrastive loss.

Contrastive Loss

The issue with reconstruction-type objectives (which includes masked auto-encoding) for training embedding models is that they operate at the *individual* graph level. Given a single graph, they optimize the model to embed that graph in such a way that some property of the original graph can be decoded. Without incorporating any information an how different graphs relate to each other, the model has no pressure to separate semantically distinct sentences. Instead, this allows graphs to be projected into a low-dimensional, dense subspace whose geometry captures almost no information on how different graphs relate.

What we need, then, is an objective that explicitly optimizes for separation by pulling together examples that should be similar and pushing apart examples that should be different. This is precisely the idea behind contrastive learning, a paradigm that has driven major advances in self-supervised representation learning across vision (*60*), language (*61*), and multimodal settings (*62*). Crucially, our setting provides a natural source of supervision for contrastive learning. Recall our access to a corpus of parallel pairs of sentences in different languages. Since our input canonicalization maps each sentence to its AMR, a parallel sentence pair $(x_{\text{en}}, x_{\text{tgt}})$ yields a pair of AMR graphs $(a_{\text{en}}, a_{\text{tgt}})$ that, if our interlingua assumption holds, represent the same meaning. We therefore want $f(a_{\text{en}})$ and $f(a_{\text{tgt}})$ to be close in embedding space, while graphs corresponding to non-parallel sentences should be far apart.

We formalize this using the symmetric InfoNCE objective (*63*). Given a batch of B parallel AMR pairs $\{(a^i_{\text{en}}, a^i_{\text{tgt}})\}_{i=1}^{B}$, we embed each graph to obtain ℓ_2-normalized representations

$$z^i_{\text{en}} = \frac{f(a^i_{\text{en}})}{\|f(a^i_{\text{en}})\|}, \quad z^i_{\text{tgt}} = \frac{f(a^i_{\text{tgt}})}{\|f(a^i_{\text{tgt}})\|}$$

We then compute the $B \times B$ similarity matrix

$$S_{ij} = \frac{(z^i_{\text{en}})^\top z^j_{\text{tgt}}}{\tau} \tag{3.1}$$

where $\tau > 0$ is a temperature parameter controlling the sharpness of the distribution. The contrastive loss is then given by averaging two cross-entropy terms:

$$\mathcal{L}_{\text{con}} = \frac{1}{2}\left[\frac{1}{B}\sum_{i=1}^{B} -\log \frac{\exp(S_{ii})}{\sum_{j=1}^{B}\exp(S_{ij})} + \frac{1}{B}\sum_{j=1}^{B} -\log \frac{\exp(S_{jj})}{\sum_{i=1}^{B}\exp(S_{ij})}\right] \tag{3.2}$$

The intuition here is straightforward. The first term treats each English embedding as an anchor and asks it to identify its parallel target-language partner among all target-language embeddings in the batch. The second term reverses the roles, anchoring from the target side. The diagonal entries of S correspond to positive pairs (parallel translations that share meaning) while all off-diagonal entries serve

as negatives. Symmetrizing the loss ensures that the alignment pressure is applied in both directions, preventing the learned space from developing a directional bias.

The off-diagonal entries provide what are known as *in-batch negatives*: graphs that happen to co-occur in the same minibatch but are not translations of one another. This is an attractive feature of the contrastive formulation, as it provides a large number of negatives at no additional computational cost. However, because these negatives are randomly sampled from the training set, they tend to be structurally dissimilar from the anchor, in a sense making the in-batch negatives *easy* examples. As a result, the model can achieve low loss simply by learning coarse-grained distinctions without learning to discriminate between semantically close but non-equivalent sentences.

Hard Negative Mining

To push the model beyond these coarse distinctions, we supplement in-batch negatives with *hard negatives*: pairs of AMR graphs that are structurally similar but do not correspond to translations of one another. The idea is to force the model to distinguish between graphs that share significant substructure but differ in meaning, pressuring the learned representations toward finer-grained semantic separation.

To mine these hard negatives, we use a metric on AMRs known as the smatch score, which we introduce in Section 3.2. For now, it suffices to understand that the smatch score gives a score in $[0, 1]$ that corresponds to how much two graphs overlap. Thus, given two non-parallel graphs, a high smatch score indicates the two graphs are structurally similar, giving us a *hard* example we can use to push the model towards fine-grained separation.

There is, however, a subtlety. Non-parallel graphs with very high structural overlap may in fact be near-paraphrases and therefore false negatives; including these would corrupt the training signal. To guard against this, we calibrate an upper bound on the smatch scores we are willing to accept. To do so, we sample a large

number of known positive pairs and compute the smatch for each. Taking the 90th percentile of this distribution of scores as our upper bound θ_{high}, we reason that non-parallel pairs exceeding this threshold are likely semantically equivalent and should be excluded. We then sample a large pool of candidate pairs consisting of non-parallel AMR graphs, compute their smatch scores, and retain pairs whose smatch falls within a window $[\theta_{\text{low}}, \theta_{\text{high}})$, where θ_{low} ensures a minimum level of structural similarity. We choose 0.3 as a somewhat arbitrary but intuitive threshold for minimum overlap.

During training, we maintain a *negative bank* of hard negatives that is refreshed at the start of each epoch. Negatives are sampled with probability proportional to their smatch score, so that structurally harder examples appear more frequently during training. Each sampled negative graph is embedded through the frozen GNN to produce a bank of embeddings $\{z^k_{\text{hard}}\}_{k=1}^{N_{\text{hard}}}$. We then augment the similarity matrix by concatenating the in-batch similarities with the similarities to the hard negative bank, yielding an expanded matrix of size $B \times (B + N_{\text{hard}})$:

$$\tilde{S}_{ij} = \begin{cases} S_{ij} & j \leq B \\ (z^i_{\text{en}})^\top z^{j-B}_{\text{hard}} \,/\, \tau & j > B \end{cases} \tag{3.3}$$

The cross-entropy labels remain unchanged so that hard negatives appear only as additional distractors in the softmax denominator. This forces the model to rank the true positive above not only the random in-batch negatives but also the structurally similar hard negatives, sharpening the learned representations and producing an embedding space in which graphs are meaningfully separated in accordance with their semantic overlap. See Chapter A for a discussion of the data used for training.

3.2 Evaluation

Let us return again to our goal of producing embeddings of sentences that (1) respect the topological structure of the sentence and (2) are meaningfully separated

in accordance with their semantic similarity. We have shown how GNNs can help us accomplish (1) and introduced the objective that is designed to maximize (2). But, with a family of GNNs and two data presentation formats, how do we decide which recipe is best?

3.2.1 The Smatch Score

Given that AMRs are constructed to represent meaning, let us begin by thinking how we can use them to quantify how similar two sentences are in meaning. First, we can decompose two AMR graphs into two sets of triples (u, τ, v) corresponding to all the relationships τ between all the nodes u, v in the graphs. Then, designating one graph as the candidate as the other as the gold, we can search for an optimal alignment[1] of both graphs and count the number of matching triples M in both graphs. Finally, letting G be the number of triples in the gold graph and C be the number of triples in the candidate graph, we can compute the precision, recall, and F_1 score of our matching:

$$P = \frac{M}{C}, \quad R = \frac{M}{G}$$

$$F_1 = \frac{2PR}{P+R}$$

yielding a score in $[0, 1]$ that we interpret as how much two AMR graphs overlap; that is, how similar the two underlying sentences are in meaning.

What we have described is referred to as the *smatch* score, the literature-standard metric for comparing two AMR graphs (*64*). While this metric is principled, accepted, and intuitive, it is not perfect, and it is important we thoroughly discuss its setbacks.

The most fundamental limitation is that smatch treats all triples equally. A mismatch on a minor modifier (say, a `:location` or `:time` relation edge), is

[1] As it turns out, the optimal alignment problem is NP-hard, making exact computation intractable. In practice, smatch computation uses a hill-climbing approximation of the optimal alignment, adding some noise to the result.

penalized exactly the same as a mismatch on the root predicate, which may change the entire meaning of the sentence. Consider "The boy wants to go to school" and "The boy wants to go to the park." These sentences differ in a single concept node, and smatch will reflect this as a high score. This is reasonable. On the other hand, consider "The boy wants to go" and "The boy wants to eat." Here too, only one concept differs and the rest of the graph matches perfectly, resulting in an identical smatch score as the previous example. However, it is clear that the first pair of sentences are semantically more related than the second pair are despite the identical smatch scores.

This uniform weighting is especially problematic in the presence of negation. Adding a single `:polarity -` triple to an AMR flips the meaning of the entire sentence: "The boy wants to go" and "The boy does not want to go" are semantically opposite, yet their AMR graphs differ by exactly one triple out of several. For a sufficiently large graph, this single-triple difference will be nearly invisible to smatch.

A subtler failure mode arises from argument swaps. The sentences "The dog chased the cat" and "The cat chased the dog" contain exactly the same set of concepts and relations; only the arguments of the predicate "chase" are exchanged. Since most of the triples are shared between the two graphs, smatch will return a high score. And yet, *who is chasing whom* is precisely the kind of distinction a semantic metric should capture.

From these examples we see that though the smatch score measures how much structure two graphs share, it cannot distinguish between structure that is semantically central and structure that is peripheral. A single triple can carry the difference between a sentence and its negation, between an agent and a patient, or between two entirely different events, and smatch has no mechanism to reflect this.

Despite these limitations, smatch remains the standard metric for AMR comparison for good reason: it is fast to compute, well-understood, and correlates well with human judgments of graph quality in the aggregate (*64*). The pathological

cases described above, while real, tend to be the exception rather than the rule when comparing graphs sampled randomly from a corpus. Thus, for our purposes, smatch provides a coarse but reliable measure of semantic similarity that is sufficient both for evaluating embeddings and, as mentioned above, for mining hard negatives for training. With that, we can introduce the main metric we use to evaluate embeddings.

3.2.2 Semantic-Geometric Consistency (SGC)

If our embedding function f successfully captures the meaning of sentences as expressed through their AMR graphs, then the pairwise geometry of sentences in the embedding space should reflect the structural differences between their AMR graphs. Defining the cosine similarity $sim(x, y) = \frac{x^T y}{\|x\|\|y\|}$, let s_i, s_j refer to a pair of AMR graphs and define

$$S_{ij} = smatch(s_i, s_j), \quad E_{ij} = cos(f(s_i), f(s_j))$$

Using the Mantel correlation ρ we define the *semantic-geometric consistency* (SGC) evaluation as

$$SGC(f) = \rho(S, E)$$

Producing a score in $[-1, 1]$ whose interpretation is clear: the higher the score, the better the relationship between pairs of embeddings reflects the overlap of their source AMRs, and hence the better the geometry of our embedding space captures semantics—precisely what we are looking to do! As such, we use SGC as our main metric to evaluate embedding recipes.

Our motivation here for using the Mantel test as opposed to the more standard Pearson's or Kendall's is that we are not evaluating linear or ranking correlation. Instead, we are comparing two notions of distance over the same set of objects, which is exactly what the Mantel test is meant for.

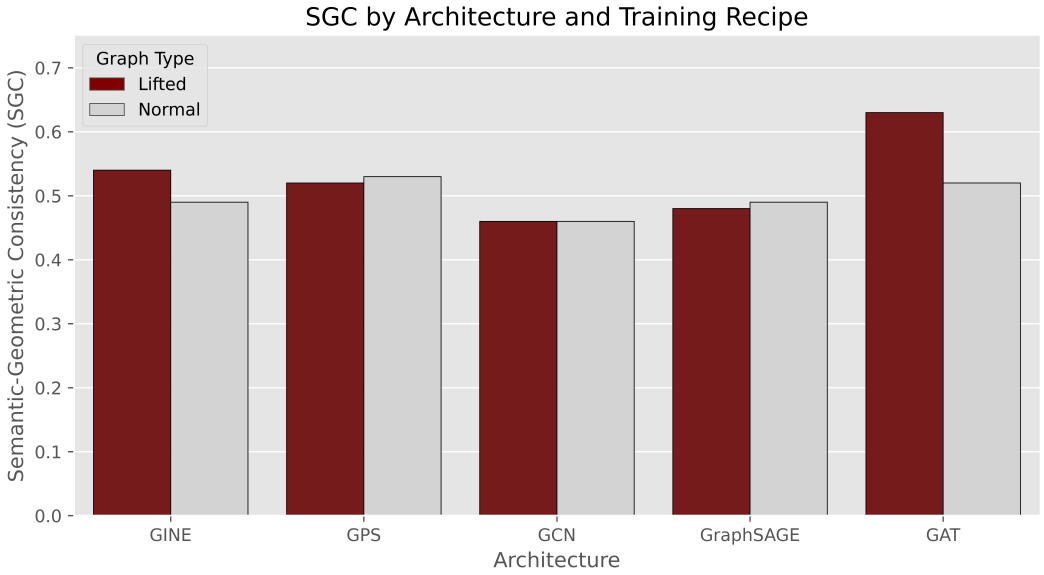

Figure 3.3: Embedding recipes evaluated on Semantic-Geometric Consistency (SGC).

Finally, having introduced our methodology, we present the result of evaluating SGC on all the recipes discussed in Figure 3.3. We see that the GAT architecture trained on lifted graphs is a clear winner with an SGC score of 0.63, indicating strong alignment between the geometry of the representation space and meaning in AMR space.

And so, with this recipe in hand, we have accomplished our goal of using AMRs to create translation-invariant representations of sentences that *respect topological and semantic structure*. With that, we withhold further discussion of these results to Chapter 5 and proceed to Chapter 4, where we discuss our approach to applying this geometric perspective to natural language processing to real tasks.

Chapter 4

STRUCTURALLY AUGMENTED MACHINE TRANSLATION

In the previous chapter, we approached Natural Language Processing through the lens of Geometric Deep Learning. By identifying translation between languages as an approximate symmetry of meaning, we developed representations designed to respect this symmetry. We now turn to investigating how these representations can be leveraged downstream.

A central reason GDL succeeds in practice is that encoding known structure as an inductive bias restricts the model class to functions consistent with this structure, thereby improving sample efficiency and generalization. We therefore need a downstream task where it is essential for representations to behave predictably under our symmetry. Machine translation is a particularly natural testbed: it demands that meaning be preserved while surface form changes across languages. In this chapter, we will demonstrate how translation-invariant semantic embeddings can be used as an auxiliary component to augment neural machine translation systems.

4.1 Method

Modern machine translation has increasingly shifted to using large language models, which have consistently achieved state-of-the-art results on standard benchmarks (*65*, *66*). We therefore take LLMs as our base systems and ask whether their translation capabilities can be improved by injecting external geometric structure in the form of our translation-invariant embeddings.

4.1.1 Task and Baselines

We elect to evaluate models on WMT24++ (*66*), chosen since it is the most recent, standard evaluation dataset that includes all our translation directions (English↔{German,

Spanish, Italian, Chinese}). This dataset consists of 998 pairs of parallel sentences across all the languages we consider. Sentences are collected across literary, news, social, and speech domains, and are translated by humans.

The metric we use is the reference-based system COMET (*67*). We describe this system in section 4.2.

Finally, our goal of augmenting pre-trained LLMs presents a natural baseline. We select Qwen 3 (*68*) and Gemma 3 (*69*) as our base systems. These models are chosen for their strong multilingual support and comparable size and performance, which allows us to test whether our approach generalizes across architectures. We select the 4 billion parameter versions of these models, which presents us with the best balance in the tradeoff between computational tractability and performance.

To understand how much our semantic embeddings improve MT performance, we compare each augmented system against both its stock, pre-trained counterpart and its fine-tuned counterpart. This enables us to disentangle the contribution of our geometric augmentation from gains achievable through standard task-specific fine-tuning.

4.1.2 Integration Mechanism

Our goal of augmenting LLMs with our semantic embeddings raises a design question: *how* should we integrate these embeddings into an LLM? One approach would be to incorporate them during pre-training, allowing the model to learn from the outset how to exploit the semantic structure of its inputs. This would be especially compelling in low-resource settings, where scarce parallel data may prevent a model from discovering cross-lingual regularities on its own. However, pre-training billion-parameter models from scratch is prohibitively expensive, and tying our evaluation to a single pre-training run would make it difficult to isolate the contribution of the embeddings themselves.

Instead, we pursue a more controlled and practical strategy: fine-tuning pre-

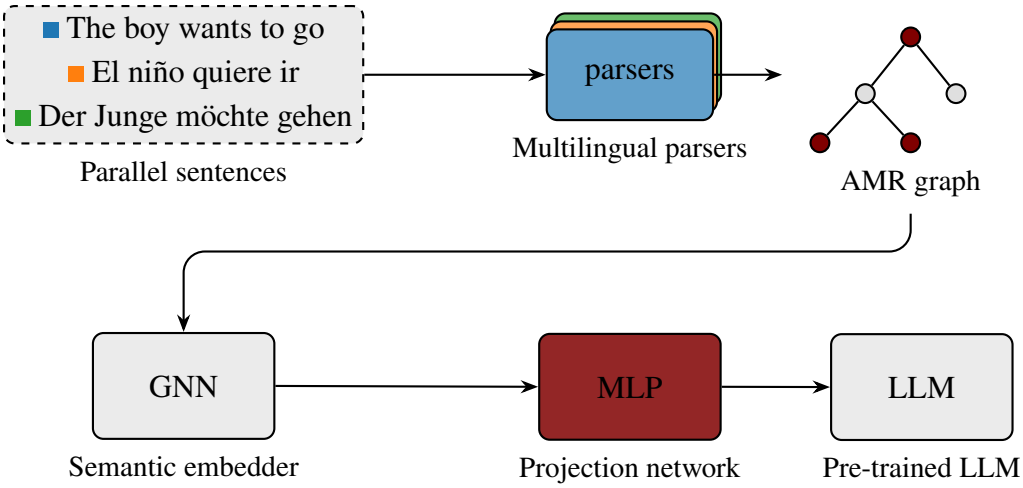

Figure 4.1: The integration pathway. A trainable projection network augments a pre-trained LLM with AMR-derived semantic embeddings.

trained LLMs with our embeddings as an auxiliary input. This setting poses a sharper test of our representations. A pre-trained model already possesses rich internal representations of language; the question is whether externally supplied geometric structure can provide complementary information that the model has not already captured. If translation-invariant embeddings improve performance even when layered onto a strong existing system, this constitutes stronger evidence for their utility than improvements from training a weaker model from scratch. We leave structurally-augmented pre-training as a direction for future work.

Soft Prompt Tuning

Even within the fine-tuning regime, there are many possible integration strategies. At a high level, what we require is a method to project our semantic embedding into the LLM's representational space. To make precise what we mean by the LLM's representational space, we can think of a model as follows.

Suppose Λ^* is the space of possible sequences of tokens from our vocabulary Λ. Then an LLM $f_{LLM} : \Lambda^* \to \Delta^{|\Lambda|}$ is a function mapping sequences of tokens to a

probability distribution over the next token, given by

$$f_{LLM} = \textsc{Unembed} \circ \textsc{Embed}$$

Where Unembed maps the sequence representation learned by Embed to a probability distribution $\Delta^{|\Lambda|}$ over our vocabulary that we then sample from. Thus, zooming into Embed, suppose our LLM is composed of L Transformer layers. Then

$$\textsc{Embed} = \textsc{Trans}^{(L)} \circ \cdots \circ \textsc{Trans}^{(1)} \circ \textsc{Emb}$$

where $\textsc{Emb} : \Lambda \to \mathbb{R}^d$ is the embedding matrix sending tokens to their *initial* d dimensional representation that is then refined by the L Transformer layers.

Thus the "representational space" of an LLM is really a sequence of maps that iteratively refines the initial representation given by Emb. Recall from chapter 2 that each Transformer layer itself is a composition of many maps. This gives us many choices for where exactly along the LLM's computational trace to project our semantic embeddings into.

What, then, should we choose? At the input level, one can concatenate or additively combine the semantic embedding with the token representations produced by Emb, so that the geometric signal is present from the very first layer. At intermediate layers, one could inject information into the residual stream at some layer l, or introduce cross-attention modules that allow the Transformer's hidden states to attend over the semantic embedding. At the output level, one could use the semantic embedding to bias or reweight the logits produced by Unembed, intervening only at the final prediction step.

A priori, it is unclear which choice is best. However, since each layer processes information given by the preceding layer, a natural choice is an *input level intervention*: projecting our semantic embeddings into the model's input embedding space in $\mathbb{R}^d$, the space spanned by the Emb matrix. This ensures the information present in the semantic embedding is able to be incorporated at every step of the LLM's computation and also gives the LLM the flexibility to learn *how* exactly to

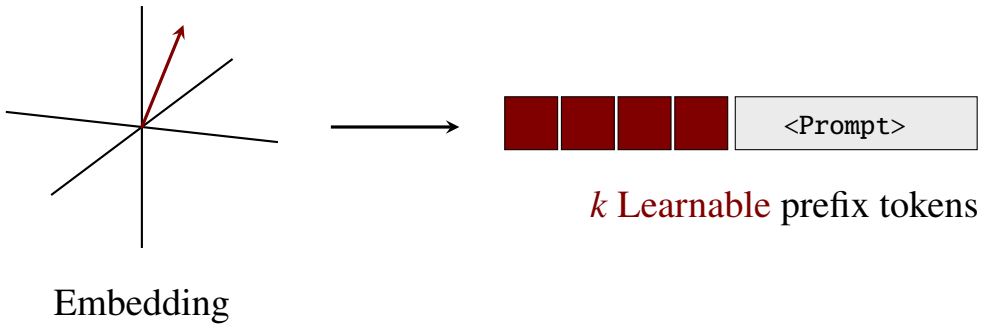

Figure 4.2: Soft Prompt Tuning (SPF). The semantic embedding is projected into a set of k learnable prefix tokens that are prepended to the LLM's translation prompt.

incorporate this information. This brings us to *Soft Prompt Tuning* (SPT)(*70*), the integration method we will use.

The core idea behind SPT is simple. A model's prompt serves to condition the output. How do we optimize this conditioning? By making part of the prompt *learnable* (i.e., "soft"), we allow gradient descent to optimize the conditioning signal. Since autoregressive LLMs only attend to previous tokens in the input sequence, prepending this "soft prompt" to the actual prompt given to the model ensures the model is able to incorporate information from this optimized conditioning in its processing of the prompt.

As Su et al. discuss(*71*), adapting this paradigm to inject graph-level information (our semantic embeddings) into the model is simple. Freezing the GNN backbone that produces our semantic embeddings, we can define a simple projection network $\textsc{Proj} : \mathbb{R}^s \to \mathbb{R}^{k \times d}$ that takes our semantic embedding in $\mathbb{R}^s$ and sends it to k virtual tokens of dimension d— our "soft" prompt. Thus, letting $(\cdot \parallel \cdot)$ denote the concatenation operator, x be a sequence of tokens, and g be our semantic embedding, our integrated model becomes

$$f_{HYB}(x) = \textsc{Unemb} \circ \textsc{Trans}^{(L)} \circ \cdots \circ \textsc{Trans}^{(1)} \circ (\textsc{Proj}(g) \| \textsc{Emb}(x))$$

defining a hybrid architecture that incorporates our GNN-derived semantic embeddings g into any LLM's computations. In words, the source sentence is parsed to

an AMR, embedded by the GNN, projected into k prefix tokens, and concatenated with the tokenized prompt before being processed by the LLM.

Since g is supposed to correspond to the "meaning" of a sentence modulo language, we take it to be the embedding of the AMR for the *source side* sentence that is to be translated. In practice, we fix learnable token count $k = 4$, chosen by tuning over $k \in \{1, 2, 4, 8, 16\}$ on performance on a test set. We note that the system seems to be insensitive to k, in that performance does not change significantly as k changes..

4.1.3 Training Protocol

Loosely speaking, our injection pathway is a composition GNN $\rightarrow$ projection network $\rightarrow$ LLM of three trainable modules. We begin by describing how the first module is trained.

Certainly the projection network must be trained to align the GNN's and LLM's representational spaces. This leaves us with the choice of whether to fine-tune the LLM on top of the semantic augmentation or to leave it frozen. We consider both options, allowing us to observe the effect of our geometric conditioning in isolation and to consider our method as a lightweight component to push the performance of the standard fine-tuning paradigm even further.

Scaling the AMR Embedder

In chapter 3, we described the process of using GNNs to embed AMR graphs into translation-invariant semantic embeddings. To select and tune the best GNN architecture, we used a subset of the training set we describe here to sweep over the various architectures and training recipes. Based on our Semantic Geometric Consistency (SGC) metric, we selected GAT trained on lifted graphs as our best system.

To integrate our embedder into our machine translation pipeline, we must scale the data it is trained on to ensure it is robust to the various domains and languages it will encounter at test time. To do so, it is necessary to create a large dataset of 4 million "silver" (machine generated) graphs for training. These graphs are generated by using the StructBART (*34*) neural parsers to convert a subset of the ParaCrawl multilingual dataset (*72*) into AMRs. We give a detailed discussion of how this was accomplished in Appendix A.

The purpose of the embedder is to convert AMRs into vector representations of meaning. As such, once the embedder is trained on our dataset, we leave it frozen during Soft Prompt Tuning to retain the geometric signal it gives us.

Training the Projection Network

Soft Prompt Tuning gives us a sort of hybrid LLM where the embeddings from our projection network form part of the input. Thus, to train it, we simply train the hybrid LLM with the standard next-token-prediction objective and let the gradients flow backwards through the projection network.

We define the projection network as a simple MLP, taking in a vector from the GNN's embedding space and projecting it to a set of k vectors in the LLM's embedding space. In particular, we define a one-layer MLP with a hidden dimension of 128 and use the GeLU as the activation function (*73*).

Fine-tuning

The last trainable module in our pathway is the LLM itself. As mentioned above, rather than training it from scratch, we instead elect to fine-tune it. In particular, we use Low-Rank Adapter fine-tuning (LoRA) (*74*).

Typical fine-tuning picks some set of modules in the network to tune (e.g., attention layers, the unembedding layer, the whole network, etc.) and tunes each individual parameter in those modules. While not as expensive as pre-training,

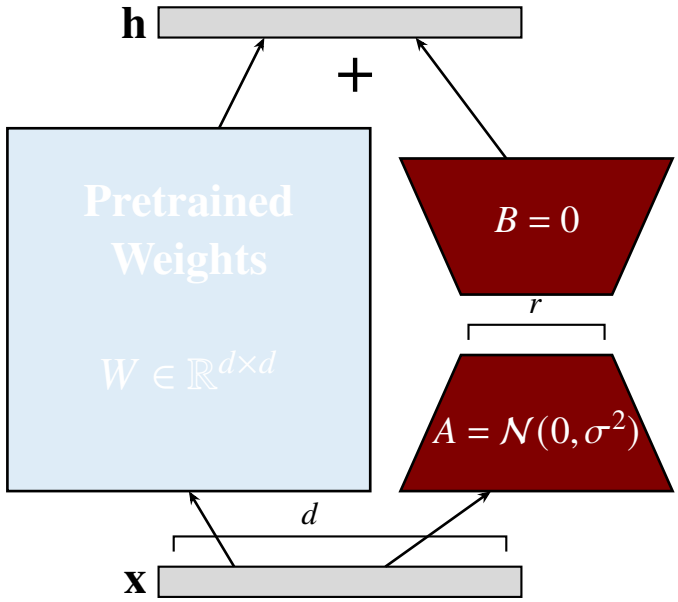

Figure 4.3: LoRA. Low rank adapters are fitted to weights and tuned to save computation.

fine-tuning in this style can still often be intractable. LoRA works by decomposing a layer $X \in \mathbb{R}^{d \times d'}$ into a *low-rank* update. Rather than modifying X directly, LoRA freezes the pre-trained weights and introduces a low-rank perturbation, so that the effective weight matrix becomes

$$X + \Delta X = X + BA,$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d'}$, with rank $r \ll \min(d, d')$. This reduces the number of trainable parameters per adapted layer from $d \cdot d'$ to $r(d + d')$, which for small r represents a dramatic reduction in memory and compute cost. See figure 4.3 for an illustration.

In practice, we fix $r = 16$ and apply adapters to the attention layers. Attention layers are the typical choice for LoRA tuning since tuning them results in performance gains comparable to full fine-tuning (*74*). This leaves us with ≈ 8.5 million trainable parameters, which is a substantial decrease from the 4 billion parameters in the network.

It is important to note that it is possible to inject our semantic embeddings via

SPT by training the projection network *and* fine-tuning the base LLM. We elect to consider both SPT with the LLM frozen and with fine-tuning the LLM to isolate the effect of our augmentation and test how much we can improve performance past standard fine-tuning.

4.2 Evaluation

COMET

Having described our method for injecting translation-invariant semantic embeddings into pre-trained LLMs, we now turn to the question of evaluation: how do we measure whether our augmentation actually improves translation quality?

The standard approach in machine translation has historically been to evaluate systems using BLEU (*75*), a metric that scores translations by counting n-gram overlaps between the machine output and a set of human reference translations. While BLEU has the advantages of being simple, fast, and widely understood, it suffers from operating entirely at the surface level; two translations that are semantically equivalent but use different word choices or syntactic structures will receive a low BLEU score, even though a human evaluator would judge them equally correct. This is particularly problematic in our setting, where the entire motivation is that meaning should be preserved across surface-level variation.

For these reasons, we adopt COMET (*67*) as our primary evaluation metric. COMET is a neural, reference-based evaluation framework that scores translations by leveraging cross-lingual language representations. Given a source sentence, a machine translation, and a human reference translation, COMET produces a score in [0,1] reflecting how faithfully the machine output captures the meaning of the source. A score close to 1 indicates a high-quality translation, while a score close to 0 indicates output no better than random chance. Absolute COMET scores vary considerably across language pairs and domains, with the original paper

reporting typical system-level averages in the range of 0.4–0.6 depending on the language pair (*67*). At the competitive WMT shared tasks, top-performing systems on well-resourced language pairs (e.g., English–German) achieve COMET scores of approximately 0.87–0.90 while more challenging or lower-resource directions yield lower scores.

At its core, COMET is built on top of a pre-trained cross-lingual encoder (XLM-RoBERTa (*76*)) that maps all three inputs (source, hypothesis, and reference) into a shared multilingual embedding space. These representations are then combined and passed through a regression head that is trained on human quality judgments.

This training paradigm gives COMET a decisive advantage over surface-level metrics. In particular, COMET achieves significantly higher correlation with human judgment than BLEU across a wide range of language pairs and domains (*67*, *77*). COMET's excellent correlation with human judgement, consistent use in the literature, and scalability makes it the perfect candidate for our use.

Chapter 5

RESULTS

Up to now, we have described the process of our approach to applying the symmetry-first principle of Geometric Deep Learning to Natural Language Processing. Our approach led us to the creation of a two stage pipeline in which we embed canonical input graphs to enforce symmetry and then augment machine translation systems with this geometric signal. This leaves us with two central questions:

1.) Do our embeddings capture semantics in a geometrically meaningful way?

2.) Does injecting this geometric signal improve translation ability?

In this chapter, we demonstrate that the answer to both of these questions is *yes*.

5.1 Embedding Analysis

5.1.1 AMR Embeddings Align With Semantic Structure

At the end of Chapter 3, we presented a plot of the result of evaluating our architecture sweep using our Semantic Geometric Consistency (SGC) metric. We saw that the Graph Attention Network (GAT) trained on lifted graphs was a clear winner, achieving an SGC score of 0.63, indicating strong alignment with the geometry of its embedding space and the semantic structure of embedded sentences.

Across architectures, training on lifted graphs improves performance on average, but this statistic is skewed by GINE and especially by GAT, which saw a 0.11 increase in SGC score when training on lifted graphs. This is likely the result of the attention mechanism GAT possesses. Without the lift, edge features are used as modifiers of attention scores between neighboring concept nodes. When we lift these edges to nodes, however, they become first-class entities themselves. This allows the model to capture richer information flow along the graph as concept

Architecture	Graph Type	
	Lifted	Normal
GINE	0.54	0.49
GPS	0.52	<u>0.53</u>
GCN	0.46	0.46
GraphSAGE	0.48	0.49
GAT	**<u>0.63</u>**	0.52

Table 5.1: Semantic-Geometric Consistency (SGC) scores by architecture and graph representation. *Lifted* denotes graphs where edge relations are promoted to nodes; *Normal* denotes standard multirelational graphs. The best score in each column is <u>underlined</u>; the overall best is **bolded**.

nodes can incorporate information from edge nodes, which themselves incorporate information from concept neighbors, allowing the model to very flexibly capture the dynamics between AMR concepts and relations.

Notably, while GPS also has an attention component, its performance does not when trained on lifted graphs. The key difference between GPS attention and GAT attention is that GAT attention is *local*; each node attends only to its 1-hop neighbors. On the other hand, GPS computes attention *globally*. Beyond attentional differences, GPS consists of a GINE backbone, which is explicitly designed to incorporate edge feature information. We hypothesize that global attention combined GINE's edge feature incorporation is sufficient to model concept-relation dynamics such that lifting becomes redundant, resulting in a diluted signal that does not boost the expressivity of the resulting representation.

5.1.2 AMR Embedding Geometry Captures Semantics

The SGC score gave us a quantitative look at how embedding geometry captures semantic structure as defined by AMR graph topology. Despite the limitations of the smatch score, it is still robust enough for us to claim this alignment is still a good proxy for how well embedding geometry aligns with semantics.

On top of this quantitative view, we can also take a very informative qualitative

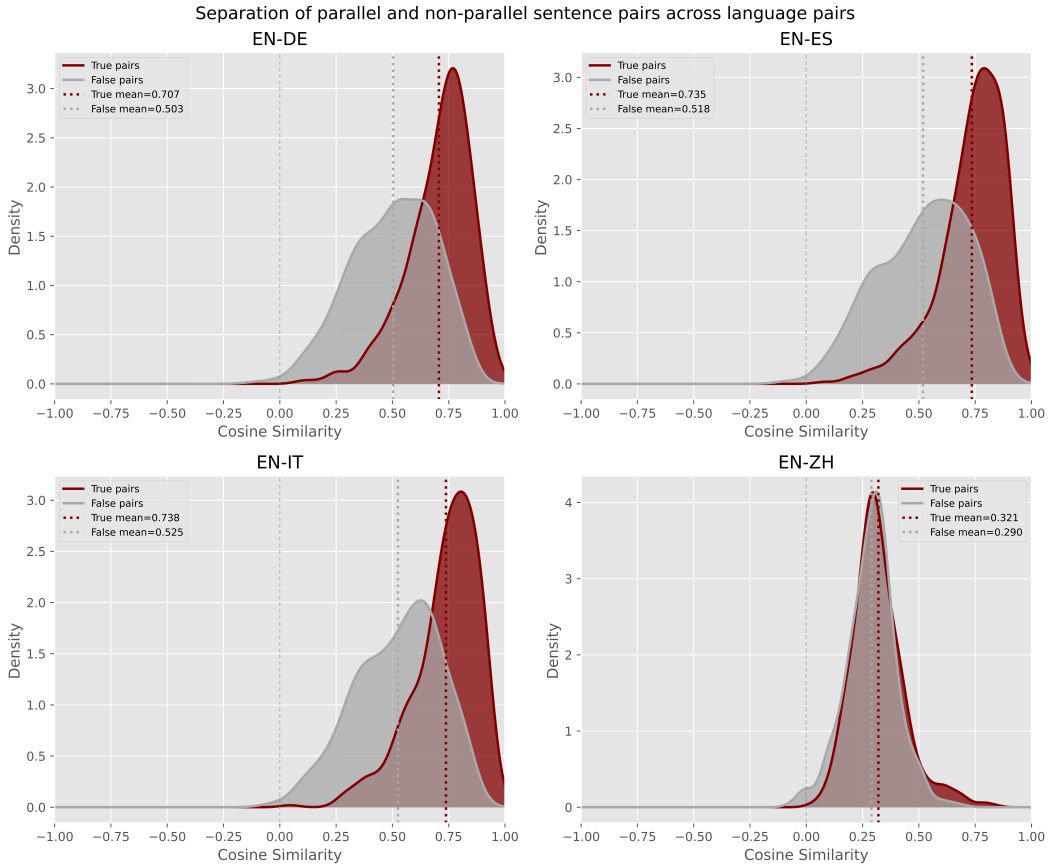

Figure 5.2: Distributions of alignment between parallel (true) and non-parallel (false) pairs of sentence embeddings. In every language pair except English-Chinese, true pairs are highly aligned while false pairs are meaningfully separated, indicating that AMR embedding geometry captures semantic information

view at how well embedding geometry captures semantics. In particular, given a set of parallel sentence pairs and non-parallel pairs across the languages we consider, we can analyze the distribution of the cosine similarities of examples in these two classes. By comparing alignment across these two classes, we get a clear sense how well they are *separated* in the embedding space.

In Figure 5.2, we see that parallel sentences pairs are highly aligned across every language pair except for English-Chinese. In the German, Spanish, and Italian setting, the distributions have significant mass and low variance around the maximal similarity of 1, with an average cosine similarity of 0.727. On the other hand, the similarity distributions for false pairs has a much higher variance and a

lower mean similarity of 0.515. This results in a meaningful similarity gap of 0.212 across these three language pairs, indicating that AMR embedding geometry does indeed meaningfully capture semantics. Further, based on the very high alignment between parallel pairs of sentences, we see our embeddings capture our approximate symmetry to a high degree.

Despite the gap, there is still a region of substantial overlap between the two distributions, pointing to the existence of a highly non-trivial class of parallel graphs with low alignment and non-parallel graphs with high alignment. The cause for this region of overlap is not clear, but we hypothesize it has to do with our interlingua assumption which we shall analyze more closely in Section 5.1.3. That is, while AMR can function as an interlingua for downstream applications, automatic parsers often produce parallel graphs that do not align as much as one would like. As such, the structural signal the embedder receives is rather noisy, with parallel graphs that do not align enough and non-parallel graphs that align too much. Indeed, the results we see are consistent with this line of thinking. Non-Chinese parallel graphs have a moderately high degree of overlap, whereas English-Chinese parallel graphs have almost no overlap (Figure 5.3), resulting in an almost entirely noisy signal that then leads to the representational collapse we see. While we see clearly that English AMRs are ill-suited for Chinese, the complete parser failure makes it unclear how much of this is due to the purely linguistic structural difference between the two languages.

5.1.3 Evaluating the Interlingua Assumption

We can evaluate our interlingua assumption directly by inspecting the distribution of smatch scores of parallel sentences across our language pairs. What we see in Figure 5.3 confirms our suspicion about the the incompatability between English and Chinese. Unlike the other language pairs, that all have very similar smatch score distributions with high mean, smatch scores for parallel English-Chinese sentences

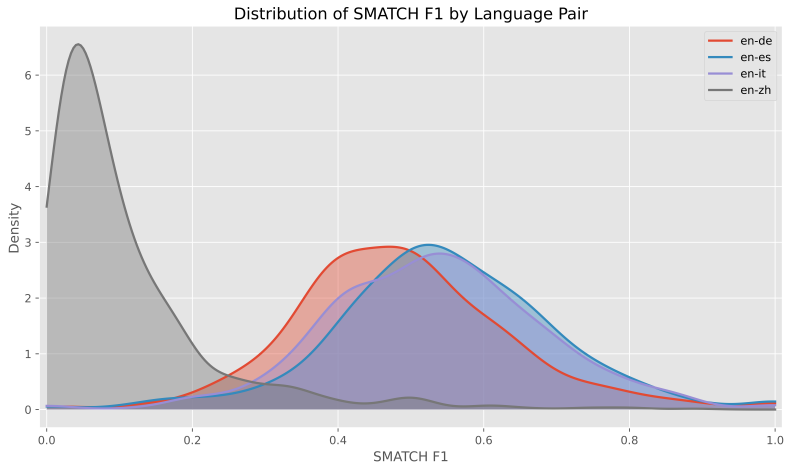

Figure 5.3: The distribution of smatch scores for parallel sentences across language pairs. English-Chinese sentences have a significantly lower smatch scores than other language pairs.

are heavily clustered around 0, with median score of 0.07. From this, it clear that the cause of representational collapse for this language pair is likely due to the clear violation of our interlingua assumption. For every other pair, however, we see the assumption holds for the most part, which we see reflected in the success of their embeddings. From this, we conjecture that embedding separability is causally linked to parallel overlap.

While structural overlap between parallel graphs is certainly an axis that can be improved, it remains unclear whether the performance we observe is due to the expressivity of AMR as a formalism or whether it is due to parser performance. Certainly, however, parser performance can be improved. Notably, exploring LLMs as AMR parsers remains underexplored. While Ettinger et al. (*44*) found that LLMs parsed with effectively 0% accuracy, this result is from 2023, which is far too old to be reliable given the rapid pace of progress in AI. Indeed in 2025, Han (*78*) found a vastly different result, with models able to parse with accuracy up to 80%. We leave further exploration of this direction to future work.

5.2 Translation Performance

A Note on Missing Scores

Before we begin our discussion, we must explain the missing entries in the baseline columns of tables 5.1 and 5.2. While COMET is widely accepted as being reliable, it expects as an input the tuple (`source_sentence, translated_sentence, gold_translation`). In our evaluation pipeline, we compute the translated sentence by prompting the model to produce a translation. In the case of the baseline models, the model often outputs significant distractor text composed of reasoning tokens such as "Okay, the user wants me to...", "Let's break down this sentence:", and "Here are some options:", among other examples. These artifacts significantly bias the resulting score the model produces. This is especially problematic when translating from other languages into English, as the presence of coherent English reasoning traces alongside the translation substantially boosts the COMET score. For these reasons, we omit the baseline scores in the X→English direction.

To address this problem, we attempted extensive prompt engineering and aggressive text filtering pipelines. Despite our best efforts, these models were explicitly trained to be verbose, helpful assistants, and we were unable to prevent them from outputting distractor text. Notably, this problem was entirely absent from fine-tuned and soft-prompted models, as we trained them explicitly to provide only the relevant translation. We recognize that these baseline scores are a crucial comparison point. However, based on the similarity of scores across both directions for the fine-tuned and soft-prompted systems, we argue it is reasonable to expect baseline X→English scores to be a in small neighborhood of English→X scores.

5.2.1 Geometric Signal Improves Translation Ability

Moving to the downstream applications of our semantic embeddings, we report the COMET scores of our augmented systems in Tables 5.1 and 5.2. As Figure 5.4

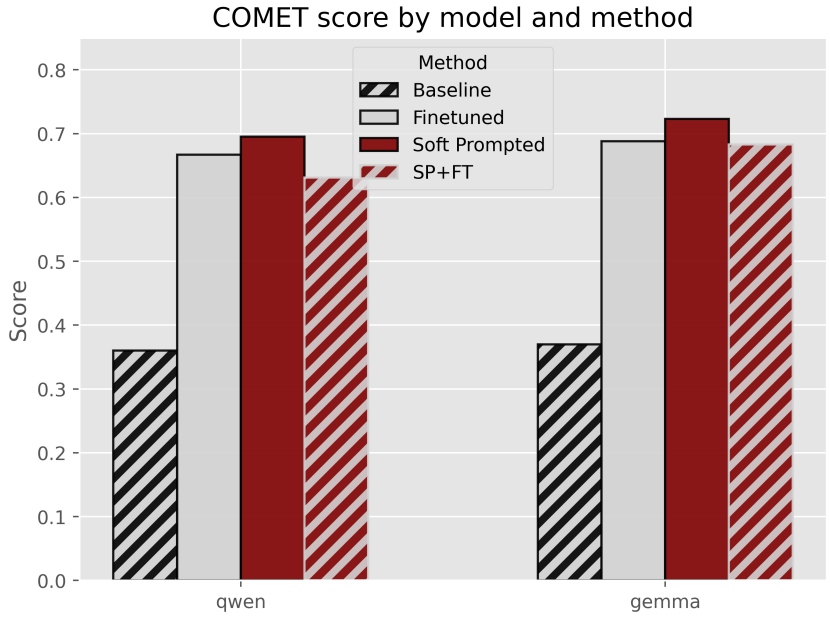

Figure 5.4: Evaluation COMET score of machine translation systems averaged across language pairs. Injecting our invariant embeddings results in superior or equal performance to conventional fine-tuning.

summarizes, injecting our embeddings via Soft Prompt Tuning results in superior performance to conventional LoRA fine-tuning across both architectures. However, this aggregate masks a far more interesting story; the two models benefit from the geometric signal in almost perfectly complementary directions. Understanding this asymmetry requires attending both to the structure of the embeddings themselves and to the substantial differences in how each base model was trained.

To contextualize the asymmetry, it is worth reviewing the pre-training regimes of the two base systems we consider. Qwen 3 (4B) was developed by the Qwen team at Alibaba Cloud and was pre-trained on approximately 36 trillion tokens spanning 119 languages and dialects (*68*). Alibaba's position as a Chinese technology company means that the training data almost certainly contains a very large proportion of high-quality Chinese text relative to Western-developed models.

Gemma 3 (4B), developed by Google DeepMind, was pre-trained on approximately 4 trillion tokens (*69*), roughly one-ninth the volume of Qwen's training set

Table 5.1: COMET scores on WMT24++ using Qwen 3 (4B).

Dir.	Base	FT	SPT	SPT+FT
en→de	0.362	**0.669**	0.657	0.612
en→es	0.366	**0.703**	0.673	0.639
en→it	0.364	**0.704**	0.683	0.624
en→zh	0.360	0.640	**0.662**	0.641
de→en	/	0.634	**0.731**	0.614
es→en	/	0.683	**0.712**	0.644
it→en	/	0.673	**0.723**	0.644
zh→en	/	0.625	**0.719**	0.627
en→x avg.	0.363	**0.679**	0.669	0.629
x→en avg.	/	0.654	**0.721**	0.632
total avg.	0.363	0.667	**0.695**	0.631

Table 5.2: COMET scores on WMT24++ using Gemma 3 (4B).

Dir.	Base	FT	SPT	SPT+FT
en→de	0.364	0.682	**0.748**	0.696
en→es	0.371	0.717	**0.771**	0.714
en→it	0.368	0.721	**0.774**	0.718
en→zh	0.359	0.659	**0.754**	0.647
de→en	/	0.673	**0.685**	0.664
es→en	/	**0.703**	**0.703**	0.699
it→en	/	**0.702**	0.699	0.694
zh→en	/	0.640	**0.645**	0.630
en→x avg.	0.366	0.695	**0.762**	0.694
x→en avg.	/	0.680	**0.683**	0.672
total avg.	0.366	0.688	**0.723**	0.683

Base: stock pre-trained model; *FT*: LoRA fine-tuned; *SPT*: Soft Prompt Tuning with LLM frozen; *SPT+FT*: both combined.

at the same parameter count. To compensate for this difference, Gemma 3 is trained with knowledge distillation from larger teacher models, inheriting representational quality that would otherwise require substantially more data. The model is trained with a tokenizer that is specifically designed to be more balanced for non-English languages, with particular improvements in the encoding efficiency of Chinese, Japanese, and Korean text. Despite covering over 140 languages, the 4B variant's substantially smaller token budget means that per-language depth is necessarily lower than Qwen's. These differences set the stage for interpreting the directional asymmetry we observe.

Directional asymmetry

The most striking pattern in our results is that Qwen and Gemma benefit from soft prompt injection in almost opposite translation directions. For Qwen, SPT produces a 10.2% average improvement on X → English, while English → X performance is slightly lower than fine-tuning. For Gemma, the pattern reverses: SPT lifts English → X performance by 9.6% on average, while X → English barely moves.

In both cases, the effect size in the favored direction is substantial while the non-favored direction shows near-zero or slightly negative deltas. This duality suggests that the geometric signal addresses different bottlenecks in the two models.

The natural first explanation for directional variation is parse quality. In English → X directions, the source AMR is parsed from English using the highest-quality StructBART parser, producing clean embeddings. In X → English directions, the source AMR is parsed from a non-English sentence using a multilingual parser of lower accuracy, producing noisier embeddings.

However, while Gemma's results are consistent with this prediction, Qwen's directly contradict it, gaining almost exclusively in the direction where the embeddings are noisier. We conclude that parse quality is a relevant factor but cannot alone explain the asymmetry.

Instead, we hypothesize that the directional asymmetry reflects where each model's translation bottleneck lies (source comprehension versus target generation) and that the prefix tokens are most useful when they address the active bottleneck rather than reinforce an already-strong capability. This hypothesis is speculative but parsimonious with our observations.

Consider Qwen first. Despite its massive multilingual pre-training corpus, Qwen's LoRA-finetuned baseline shows notably weaker X → English scores (average 0.654) than English → X scores (average 0.679). One interpretation is that Qwen generates target-language text competently, likely benefiting from the sheer volume of its 36-trillion-token training set, but struggles more to extract meaning from non-English source input. The AMR prefix tokens then provide a language-agnostic semantic summary of the source sentence. Thus, even though the non-English AMR parses are noisier, this signal addresses a genuine deficit. On the other hand, in English → X directions, the prefix tokens are largely redundant with information the model has already extracted from the input, and the additional tokens may introduce noise into the attention computation without contributing new information.

For Gemma, the bottleneck appears to lie elsewhere. Gemma's LoRA-finetuned X → English scores are already reasonably strong (average 0.680), suggesting robust multilingual comprehension. What Gemma lacks relative to Qwen is raw generative depth in non-English languages. Trained on 4 trillion tokens compared to Qwen's 36 trillion, Gemma has substantially less exposure to the distributional patterns of German, Spanish, Italian, and Chinese text that are necessary for fluent generation. In English → X directions, the high-quality English-side AMR embedding provides a strong semantic scaffold that constrains generation. In effect, the prefix tokens give Gemma a compressed "check" against which it can verify that its generation in the target language preserves the intended meaning. Finally, for X → English, Gemma already generates English fluently and comprehends the source well enough that the noisier non-English AMR adds little.

To summarize our hypothesis, we postulate that Qwen's bottleneck is *comprehension* of non-English input, and so even noisy AMR embeddings help by providing a language-agnostic semantic anchor. Gemma's bottleneck is *generation* in non-English output, and so clean AMR embeddings help by scaffolding the target-side production. Along these lines, we conjecture that geometric signal is most valuable where the model is weakest.

We note that this hypothesis could be tested more directly in future work. One approach would be to probe each model's internal representations to measure source-language comprehension quality independently of generation, for instance by extracting hidden-state representations of non-English inputs and evaluating their alignment to English representations of the same content. Another approach would be to examine the attention patterns over the prefix tokens. If the bottleneck hypothesis is correct, we would expect Qwen to attend more heavily to prefix tokens when processing non-English sources, and Gemma to attend more when generating non-English targets.

The en→zh anomaly

One result demands special attention. For Gemma, SPT on en → zh scores 0.754 while FT scores 0.659. This 9.5-point gap that is massive by MT evaluation standards. For Qwen, SPT also outperforms FT on en → zh, though not as drastically, which is quite surprising in light of our embedding analysis in Section 5.1.3.

If the embedding carries negligible semantic information, what is the prefix token doing? We hypothesize a *mode switch* effect: even a semantically degenerate embedding occupies a distinct region of the LLM's input embedding space and may function as a task indicator rather than a meaning carrier. The prefix tokens signal to the model that a translation task involving Chinese is being performed, potentially activating latent generation capabilities that LoRA fine-tuning on 1 million parallel sentences does not fully unlock. This interpretation is consistent with the literature on soft prompting, which has demonstrated that even randomly initialized continuous prompts can provide task-steering effects (*70*).

This hypothesis is testable. Replacing the AMR-derived embeddings with fixed random vectors of the same dimension would isolate the mode-switch contribution from any residual semantic content. If the random-vector baseline matches SPT performance on en → zh, the task-indicator interpretation is confirmed; if SPT retains an advantage, even the collapsed embeddings carry information that random vectors do not. We leave this experiment to future work but flag en → zh as the most surprising result in our evaluation and one that warrants further investigation.

Injection improves X→English translation

Across both architectures, the single most consistent result is that soft prompt injection improves translation into English. As discussed above, this likely ties back to our interlingua assumption: AMRs are best fit to English, and English parsers achieve the highest accuracy. When translating into English, the target language

aligns naturally with the formalism in which the geometric signal is expressed, and the resulting embeddings carry richer information. Notably, however, the magnitude of this improvement varies dramatically across architectures. Qwen's average X $\rightarrow$ en performance increases by 10.2%, whereas Gemma's increases by only 0.4% on average. Under the bottleneck hypothesis discussed above, this is expected as the geometric signal helps Qwen overcome a comprehension deficit on non-English sources, whereas Gemma, which already comprehends non-English input reasonably well, gains little from the additional signal in this direction.

SPT+FT Degrades Performance

In nearly every translation direction and for both models, combining soft prompt tuning with LoRA fine-tuning (SPT+FT) produces worse results than either method alone. This is a surprising and counterintuitive finding that bears directly on how our method should be used in practice.

We offer two non-mutually-exclusive explanations. The first is *optimization interference*. The projection network and LoRA adapters are trained simultaneously. LoRA adapts the attention layers' behavior, potentially learning to downweight or reinterpret the prefix token positions, while the projection network is trained to produce prefix tokens that are informative to the *original* attention patterns. The two optimizations may push in competing directions, with LoRA effectively routing around the prefix signal while the projection network chases a moving target.

The second is *redundancy and capacity waste*. Both SPT and LoRA are trained on the same parallel data. If LoRA fine-tuning captures much of the cross-lingual signal that the geometric embeddings provide, the prefix tokens become redundant. But they still consume four sequence positions and require attention at every layer, imposing a constant overhead on a model that has already internalized the relevant information through weight adaptation. For a well-finetuned model, this overhead may outweigh any residual benefit the prefix tokens provide.

A natural follow-up would be a staged training protocol in which the projection network is trained to convergence with the LLM frozen, and then the projection weights are themselves frozen while LoRA fine-tuning is applied. This would decouple the two optimization targets and avoid the interference we hypothesize. We leave this exploration to future work.

Chapter 6

CONCLUSION

Motivated by the possibility we might be approaching a scaling plateau, we looked to Geometric Deep Learning for inspiration and asked: *what sorts of symmetries does natural language have?*

We arrived at a simple, but ultimately effective answer: *translation*. However, while this identification is intuitive, we lack the tools to formalize it. Indeed, we saw through example that this symmetry is only an approximate one. Despite these challenges, we argued that this action is close enough to a proper symmetry and treated it as such. Following this perspective, we demonstrated how to shift the work of symmetry-enforcement away from the challenging setting of architecture design to a much simpler input design problem, using AMRs to produce embeddings in a space whose geometry captures the quotiented semantics this symmetry describes. We then pushed this perspective to its empirical end, defining hybrid architectures that integrated this geometric signal into modern LLMs. In the end, using machine translation as a testbed, we found that this perspective really did result in something meaningful. Using just four prefix vectors derived from noisy signal was enough to beat the expressivity of an additional 8 million parameters in the majority of settings.

6.1 Limitations

While fruitful, our geometric approach to NLP was certainly not without its limitations. The most pressing is the parsing bottleneck. As we saw with the complete failure of English-Chinese parsing, sending text to its faithful AMR presentation is a highly non-trivial task whose outcome is highly predictive of the quality of the resulting embeddings. In a sense, this failure is itself a testament to the coherence

of our framework; when canonical form degrades and structure is lost, downstream quality reflects this honestly.

Beyond the difficulty of the task, the computational resources required are not at all negligible. While the projection network is relatively small at 1M parameters, the GNN backbone is rather large at 100M parameters thanks to the large AMR concept vocabulary. On top of this, the parsing network itself is comprised of over 400M parameters. Together with the engineering complexity of stitching these pieces together, the complexity our method demands cannot be overlooked and is a significant limitation to our approach. Despite these setbacks, our perspective is effective, and the severity of the limitations scales inversely with base model size.

6.2 Future Work

Our discussion of limitations brings us naturally to several directions for future work. The most relevant direction to our work is the improvement of the canonical representation. While the StructBART parsers we used are state-of-the-art, their performance is certainly not a ceiling. As mentioned in Chapter 5, modern LLMs are a promising alternative to specialized parsers and have the potential to meaningfully boost the reliability and coverage of AMRs across many languages. Beyond just increasing AMR quality, it remains interesting to consider alternative canonical forms such as the Universal Meaning Representation and Universal Conceptual Cognitive Annotation (*79*, *80*), both of which were designed explicitly as interlinguas.

Further than just improvements to our particular method, the question of *where* structure exists in natural language and *how* one might incorporate it as a design principle is a deeply exciting one. As discussed in Chapter 4, incorporating structure *ab initio* via some form of structural pre-training is an exciting idea with the potential to accommodate much more structure than just this particular symmetry. Indeed, on a more speculative note, this principle can extend beyond the single modality

setting. That is, just as our language has structure, so too does the world we observe and describe with language. How can we describe and jointly model this structure? The possibilities for geometric approaches to problems in machine learning seem endless.

As we conclude our work, we hope this thesis will add to the growing body of support for the beauty that is the geometric approach to learning. As we have discussed, the dominant approach to language has been through massive scale—and not without reason. Scale has certainly worked wonders, and we by no means refute its effectiveness. However, we do claim that it leaves significant structure on the table and present our findings in support of that claim.

We are taken back 150 years ago to the words of the young Felix Klein, who proposed that geometry is the study of invariants. Through a series of approximations, we have taken these words to heart and applied this philosophy to study the incredible object that is our language. We hope that by taking seriously the question of the nature of this mysterious substrate's geometry, we might open the door, however incrementally, to a structured understanding of how meaning is organized and preserved across the wonderful diversity of human expression.

Appendix A

COMPILING AND CREATING DATA

In chapters 3 and 4, we introduced the method by which we could use the Abstract Meaning Representation formalism to treat cross-lingual translation as a symmetry of meaning, produce embeddings that are invariant to this symmetry, and use these embeddings to augment the performance of pre-trained LLMs on machine translation. As with any machine learning pipeline—and certainly as the Geometric Deep Learning philosophy would tell us—the data we used was arguably the most important part of the process. In this chapter, we will give a detailed description of the process of collecting and processing this data.

A.1 Natural Language Datasets

We begin with the natural language datasets used. This leads us naturally to discussing AMR datasets, which were primarily derived from the natural language datasets used.

A.1.1 WMT24++

We use WMT24++ (*81*) as the evaluation dataset for determining the performance of all the machine translation systems we consider. WMT24++ extends the original WMT24 General MT test set (*66*) from 9 English-paired languages to 55 languages and dialects by collecting new human-written references and post-edits. Each language pair shares the same 998 English source texts drawn from four domains: literary (206 paragraphs), news (149), social (531), and speech (111). All translations were produced by professional human translators. Notably, each source text is *paragraph level* as opposed to sentence level, making translation a more comprehensive and contextually demanding task. We use this dataset for evaluation

because it is the most recently created and highest-quality dataset containing the language pairs we consider. Additionally, it served as the primary evaluation set at the WMT24 conference.

A.1.2 ParaCrawl

ParaCrawl (*72*) is a web-scale parallel corpus made to cover European languages. We use release v9, the final release of the project, which covers all 23 official EU languages paired with English as well as a bonus English-Chinese corpus. We use two non-overlapping subsets: one for fine-tuning MT systems, and the other as the primitive we run parsers over to create the AMR datasets used for training the GNNs and hybrid systems in Chapters 3 and 4, respectively. Parallel sentences in ParaCrawl are extracted using the open-source Bitextor pipeline(*82*), which performs web crawling, text extraction, document alignment, sentence alignment, and sentence pair filtering. Filtering is carried out using Bicleaner, a classifier that scores each sentence pair by the likelihood that the two sentences are mutual translations, and noisy pairs are discarded. Release v9 further improves quality by introducing neural cleaning for the first time.

The scale of the corpus varies considerably across language pairs. For the pairs relevant to this work, ParaCrawl v9 contains approximately 278M parallel sentence pairs for English-German, 269M for English-Spanish, and 97M for English-Italian, while the bonus English-Chinese corpus is substantially smaller at approximately 15M pairs. While sentence counts vary drastically, since we sample the same number of sentences randomly and web-scraped data is very diverse, we do not expect any issues arising from these discrepancies.

We use ParaCrawl as our main training set because it is the largest and most well-maintained multilingual parallel corpus available, and it has been a primary data source for training multilingual translation models since its inclusion in WMT shared tasks beginning in 2018. In particular, we sample 1M sentences per language

Table A.1: ParaCrawl v9 subsets used in this work. "Fine-tuning" denotes data used for LoRA fine-tuning of LLMs (Chapter 4); "AMR data" denotes the disjoint subset parsed into silver AMRs for training the GNN embedder and hybrid system (Chapters 3 and 4). Sentence counts are per language pair. All pairs are with English. Token counts are only reported for natural language data.

Language Pair	**Use**	**Sentences**	**Approx. Tokens**
en–de	Fine-tuning	1,000,000	29M
en–es	Fine-tuning	1,000,000	30M
en–it	Fine-tuning	1,000,000	33M
en–zh	Fine-tuning	1,000,000	20M
en–de	AMR data	500,000	-
en–es	AMR data	500,000	-
en–it	AMR data	500,000	-
en–zh	AMR data	500,000	-

pair for LLM fine-tuning and a disjoint set of 500K sentences per language pair as the AMR dataset primitive. See Table A.1 for a summary.

A.2 Gold AMR Data

By Gold AMR data, we mean data exclusively created and verified by human annotators. Similarly, silver data refers to datasets containing AMRs created by machine parsing systems.

A.2.1 AMR 3.0

To date, AMR 3.0 (LDC2020T02) (*83*) is the largest source of gold AMR data, containing 59,255 English sentences annotated with AMR graphs. It is the primary dataset used to train machine parsers and other AMR-related systems, including the StructBART parsers we use (*34*). The corpus draws from a diverse set of domains, including DEFT and BOLT discussion forums, broadcast conversation, Wall Street Journal and Xinhua newswire, proxy reports, weblogs, LORELEI situation frames, Aesop's Fables, and Wikipedia. The data is split into 55,635 training, 1,722 development, and 1,898 test sentences. We combine the test and development sets

Table A.2: Domain distribution of the AMR 3.0 corpus (LDC2020T02). We use the combined test and dev split to evaluate GNN embedders (Chapter 3)

Domain	**Train**	**Dev**	**Test**	**Total**
BOLT DF MT	1,061	133	133	1,327
Broadcast conversation	214	0	0	214
Weblog and WSJ	0	100	100	200
BOLT DF English	7,379	210	229	7,818
DEFT DF English	32,915	0	0	32,915
Aesop's Fables	49	0	0	49
Guidelines AMRs	970	0	0	970
LORELEI	4,441	354	527	5,322
2009 Open MT	204	0	0	204
Proxy reports	6,603	826	823	8,252
Weblog	866	0	0	866
Wikipedia	192	0	0	192
Xinhua MT	741	99	86	926
Total	**55,635**	**1,722**	**1,898**	**59,255**

into a single test set to increase size and coverage. Table A.2 provides a detailed breakdown by domain. For our purposes, we use the test split to evaluate GNN embedders in Chapter 3. This lets us evaluate how our systems perform on the "true" (gold) distribution of AMR data, and also serves as an expected performance floor, as the test-set distribution is quite dissimilar from the distribution underlying the data the system sees at inference time. In particular, the inference-time distribution closely matches the distribution underlying the silver training set, as both are disjoint subsets of the same corpus (ParaCrawl).

A.3 Silver AMR Data

While silver data is inherently noisier than gold annotations, the StructBART family of parsers we employ achieves a state-of-the-art smatch score of 82.9 on AMR 3.0 (*34*), giving us confidence that the resulting graphs are of sufficient quality for our purposes.

A.3.1 WMT24++-AMR

In order to run our hybrid MT system–and therefore to evaluate it–it is necessary to provide the AMR corresponding to the source sentence that is to be translated. Therefore, we parse the WMT24++ natural language evaluation set into WMT24++-AMR to evaluate our hybrid systems. To support evaluation of translation between any pair of languages, we parse every direction in the dataset. That is, we obtain 998 English, Spanish, German, Italian, and Chinese AMRs, respectively.

A.3.2 ParaCrawl-AMR

As mentioned in Section 6.1.2, we reserve a disjoint subset of 500K sentence pairs per language pair from ParaCrawl v9 as the primitive for constructing silver AMR training data. We parse the English side of each pair using the StructBART English parser to obtain the corresponding AMR graph, yielding approximately 2M English AMR graphs. To create cross-lingual training signal, we also parse the non-English side of each pair using the respective multilingual StructBART parser, producing an additional 2M graphs across Spanish, German, Italian, and Chinese. In total, ParaCrawl-AMR comprises approximately 4M silver AMR graphs. These graphs serve two roles: training the GNN embedder (Chapter 3) and training the projection network in our hybrid MT system (Chapter 4). Crucially, this set is entirely disjoint from the 1M-per-pair subset used for LLM fine-tuning, ensuring no data leakage between the AMR and fine-tuning pipe

BIBLIOGRAPHY

1. J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, *CoRR* **abs/2001.08361**, arXiv: 2001.08361, (https://arxiv.org/abs/2001.08361) (2020).

2. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*, en, 2025, (2026; https://ai.meta.com/blog/llama-4-multimodal-intelligence/).

3. P. Villalobos, A. Ho, J. Sevilla, T. Besiroglu, L. Heim, M. Hobbhahn, *arXiv preprint arXiv:2211.04325* (2022).

4. M. M. Bronstein, J. Bruna, T. Cohen, P. Velickovic, *CoRR* **abs/2104.13478**, arXiv: 2104.13478, (https://arxiv.org/abs/2104.13478) (2021).

5. N. Chomsky, en, *Information and Control* **2**, 137–167, ISSN: 00199958, (2026; https://linkinghub.elsevier.com/retrieve/pii/S0019995859903626) (June 1959).

6. L. E. Baum, T. Petrie, en, *The Annals of Mathematical Statistics* **37**, 1554–1563, ISSN: 0003-4851, 2168-8990, (2026; https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-37/issue-6/Statistical-Inference-for-Probabilistic-Functions-of-Finite-State-Markov-Chains/10.1214/aoms/1177699147.full) (Dec. 1966).

7. P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer, *Computational Linguistics* **19**, ed. by J. Hirschberg, 263–311, (2026; https://aclanthology.org/J93-2003/) (1993).

8. S. F. Chen, J. Goodman, presented at the 34th Annual Meeting of the Association for Computational Linguistics, pp. 310–318, (2026; https://aclanthology.org/P96-1041/).

9. Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, en.

10. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, presented at the Advances in Neural Information Processing Systems, ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger, vol. 26, (https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf).

11. K. Park, Y. J. Choe, V. Veitch, *The Linear Representation Hypothesis and the Geometry of Large Language Models*, arXiv:2311.03658 [cs], July 2024, (2026; http://arxiv.org/abs/2311.03658).

12. D. E. Rumelhart, J. L. McClelland, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations* (MIT Press, 1987), pp. 318–362, ISBN: 978-0-262-29140-8, (2026; https://ieeexplore.ieee.org/document/6302929).

13. S. Hochreiter, J. Schmidhuber, *Neural computation* **9**, 1735–1780 (1997).

14. I. Sutskever, O. Vinyals, Q. V. Le, *Sequence to Sequence Learning with Neural Networks*, arXiv:1409.3215 [cs], Dec. 2014, (2026; http://arxiv.org/abs/1409.3215).

15. K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, arXiv:1406.1078 [cs], Sept. 2014, (2026; http://arxiv.org/abs/1406.1078).

16. D. Bahdanau, K. Cho, Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv:1409.0473 [cs], May 2016, (2026; http://arxiv.org/abs/1409.0473).

17. M.-T. Luong, H. Pham, C. D. Manning, *Effective Approaches to Attention-based Neural Machine Translation*, arXiv:1508.04025 [cs], Sept. 2015, (2026; http://arxiv.org/abs/1508.04025).

18. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, *CoRR* **abs/1706.03762**, arXiv: 1706.03762, (http://arxiv.org/abs/1706.03762) (2017).

19. K. Clark, U. Khandelwal, O. Levy, C. D. Manning, presented at the Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, ed. by T. Linzen, G. Chrupała, Y. Belinkov, D. Hupkes, pp. 276–286, (2026; https://aclanthology.org/W19-4828/).

20. K. He, X. Zhang, S. Ren, J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015, (2026; http://arxiv.org/abs/1512.03385).

21. J. L. Ba, J. R. Kiros, G. E. Hinton, *Layer Normalization*, arXiv:1607.06450 [stat], July 2016, (2026; http://arxiv.org/abs/1607.06450).

22. R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T. Liu, *CoRR* **abs/2002.04745**, arXiv: 2002.04745, (https://arxiv.org/abs/2002.04745) (2020).

23. B. Zhang, R. Sennrich, *CoRR* **abs/1910.07467**, arXiv: 1910.07467, (http://arxiv.org/abs/1910.07467) (2019).

24. J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, Y. Liu, *RoFormer: Enhanced Transformer with Rotary Position Embedding*, arXiv:2104.09864 [cs], Nov. 2023, (2026; http://arxiv.org/abs/2104.09864).

25. A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, en.

26. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, *Language*

Models are Few-Shot Learners, arXiv:2005.14165 [cs], July 2020, (2026; http://arxiv.org/abs/2005.14165).

27. OpenAI *et al.*, *GPT-4 Technical Report*, arXiv:2303.08774 [cs], Mar. 2024, (2026; http://arxiv.org/abs/2303.08774).

28. T. S. Cohen, M. Welling, *Group Equivariant Convolutional Networks*, arXiv:1602.07576 [cs], June 2016, (2026; http://arxiv.org/abs/1602.07576).

29. M. Weiler, M. Geiger, M. Welling, W. Boomsma, T. S. Cohen, presented at the Advances in Neural Information Processing Systems, vol. 31, (2026; https://proceedings.neurips.cc/paper/2018/hash/488e4104520c6aab692863cc1dba45af-Abstract.html).

30. L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, presented at the Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, ed. by A. Pareja-Lora, M. Liakata, S. Dipper, pp. 178–186, (2025; https://aclanthology.org/W13-2322/).

31. M. Palmer, D. Gildea, P. Kingsbury, *Computational Linguistics* **31**, 71–106, (2026; https://aclanthology.org/J05-1004/) (2005).

32. W. C. Mann, en.

33. C. Wang, N. Xue, S. Pradhan, presented at the Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ed. by R. Mihalcea, J. Chai, A. Sarkar, pp. 366–375, (2026; https://aclanthology.org/N15-1040/).

34. Y.-S. Lee, R. Fernandez Astudillo, T. L. Hoang, T. Naseem, R. Florian, S. Roukos, presented at the NAACL-HLT 2022: The 2022 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies.

35. N. Xue, O. Bojar, J. Hajič, M. Palmer, Z. Urešová, X. Zhang, presented at the Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), ed. by N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis, pp. 1765–1772, (2026; https://aclanthology.org/L14-1332/).

36. M. Damonte, S. B. Cohen, presented at the Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), ed. by M. Walker, H. Ji, A. Stent, pp. 1146–1155, (2026; https://aclanthology.org/N18-1104/).

37. S. Wein, N. Schneider, *Computational Linguistics* **50**, 419–473, (2026; https://aclanthology.org/2024.cl-2.1/) (June 2024).

38. F. Liu, J. Flanigan, S. Thomson, N. Sadeh, N. A. Smith, *Toward Abstractive Summarization Using Semantic Representations*, arXiv:1805.10399 [cs], May 2018, (2026; http://arxiv.org/abs/1805.10399).

39. H. Qiu, K.-H. Huang, J. Qu, N. Peng, presented at the Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), ed. by K. Duh, H. Gomez, S. Bethard, pp. 594–608, (2026; https://aclanthology.org/2024.naacl-long.33/).

40. Z. Zhang, H. Ji, presented at the Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ed. by K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou, pp. 39–49, (2026; https://aclanthology.org/2021.naacl-main.4/).

41. S. Wein, J. Opitz, presented at the Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, ed. by Y. Al-Onaizan,

M. Bansal, Y.-N. Chen, pp. 6856–6875, (2026; https://aclanthology.org/2024.emnlp-main.390/).

42. Z. Jin, Y. Chen, F. Gonzalez, J. Liu, J. Zhang, J. Michael, B. Schölkopf, M. Diab, presented at the, Version Number: 1, (2026; https://arxiv.org/abs/2405.01502).

43. P. Yao, K. Guzhva, D. Barbosa, Version Number: 1, (2026; https://arxiv.org/abs/2407.04067) (2024).

44. A. Ettinger, J. D. Hwang, V. Pyatkin, C. Bhagavatula, Y. Choi, presented at the, Version Number: 2, (2026; https://arxiv.org/abs/2310.17793).

45. R. Kamel, F. Guerranti, S. Geisler, S. Günnemann, Version Number: 2, (2026; https://arxiv.org/abs/2507.13381) (2025).

46. S. Furutani, T. Shibahara, M. Akiyama, K. Hato, M. Aida, en, presented at the Machine Learning and Knowledge Discovery in Databases, ed. by U. Brefeld, E. Fromont, A. Hotho, A. Knobbe, M. Maathuis, C. Robardet, pp. 447–463, ISBN: 978-3-030-46150-8.

47. L. Song, D. Gildea, Y. Zhang, Z. Wang, J. Su, *Transactions of the Association for Computational Linguistics* **7**, ed. by L. Lee, M. Johnson, B. Roark, A. Nenkova, 19–31, (https://aclanthology.org/Q19-1002/) (2019).

48. C. Li, J. Flanigan, presented at the Proceedings of the 2nd Workshop on Deep Learning on Graphs for Natural Language Processing (DLG4NLP 2022), ed. by L. Wu, B. Liu, R. Mihalcea, J. Pei, Y. Zhang, Y. Li, pp. 12–21, (https://aclanthology.org/2022.dlg4nlp-1.2/).

49. S. Yao, T. Wang, X. Wan, presented at the Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ed. by D. Jurafsky, J. Chai, N. Schluter, J. Tetreault, pp. 7145–7154, (https://aclanthology.org/2020.acl-main.640/).

50. K. Xu, W. Hu, J. Leskovec, S. Jegelka, *How Powerful are Graph Neural Networks?*, arXiv:1810.00826 [cs], Feb. 2019, (2026; http://arxiv.org/abs/1810.00826).

51. C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, M. Grohe, *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks*, arXiv:1810.02244 [cs], Nov. 2021, (2026; http://arxiv.org/abs/1810.02244).

52. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, *Graph Attention Networks*, arXiv:1710.10903 [stat], Feb. 2018, (2026; http://arxiv.org/abs/1710.10903).

53. L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, D. Beaini, *Recipe for a General, Powerful, Scalable Graph Transformer*, arXiv:2205.12454 [cs], Jan. 2023, (2026; http://arxiv.org/abs/2205.12454).

54. W. L. Hamilton, en.

55. V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, X. Bresson, en (2022).

56. T. N. Kipf, M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, arXiv:1609.02907 [cs], Feb. 2017, (2026; http://arxiv.org/abs/1609.02907).

57. W. L. Hamilton, R. Ying, J. Leskovec, *Inductive Representation Learning on Large Graphs*, arXiv:1706.02216 [cs], Sept. 2018, (2026; http://arxiv.org/abs/1706.02216).

58. E. Rossi, B. Charpentier, F. D. Giovanni, F. Frasca, S. Günnemann, M. Bronstein, *Edge Directionality Improves Learning on Heterophilic Graphs*, arXiv:2305.10498 [cs], Nov. 2023, (2026; http://arxiv.org/abs/2305.10498).

59. Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, J. Tang, *GraphMAE: Self-Supervised Masked Graph Autoencoders*, arXiv:2205.10803 [cs], July 2022, (2026; http://arxiv.org/abs/2205.10803).

60. T. Chen, S. Kornblith, M. Norouzi, G. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, arXiv:2002.05709 [cs], July 2020, (2026; http://arxiv.org/abs/2002.05709).

61. T. Gao, X. Yao, D. Chen, *SimCSE: Simple Contrastive Learning of Sentence Embeddings*, arXiv:2104.08821 [cs], May 2022, (2026; http://arxiv.org/abs/2104.08821).

62. A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, *Learning Transferable Visual Models From Natural Language Supervision*, arXiv:2103.00020 [cs], Feb. 2021, (2026; http://arxiv.org/abs/2103.00020).

63. A. v. d. Oord, Y. Li, O. Vinyals, *Representation Learning with Contrastive Predictive Coding*, arXiv:1807.03748 [cs], Jan. 2019, (2026; http://arxiv.org/abs/1807.03748).

64. S. Cai, K. Knight, presented at the Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ed. by H. Schuetze, P. Fung, M. Poesio, pp. 748–752, (2026; https://aclanthology.org/P13-2131/).

65. T. Kocmi, E. Artemova, E. Avramidis, R. Bawden, O. Bojar, K. Dranch, A. Dvorkovich, S. Dukanov, M. Fishel, M. Freitag, T. Gowda, R. Grundkiewicz, B. Haddow, M. Karpinska, P. Koehn, H. Lakougna, J. Lundin, C. Monz, K. Murray, M. Nagata, S. Perrella, L. Proietti, M. Popel, M. Popović, P. Riley, M. Shmatova, S. Steingrímsson, L. Yankovskaya, V. Zouhar, presented at the Proceedings of the Tenth Conference on Machine Translation, ed. by B. Haddow, T. Kocmi, P. Koehn, C. Monz, pp. 355–413, ISBN: 979-8-89176-341-8, (https://aclanthology.org/2025.wmt-1.22/).

66. T. Kocmi, E. Avramidis, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, M. Freitag, T. Gowda, R. Grundkiewicz, B. Haddow, M. Karpinska,

P. Koehn, B. Marie, C. Monz, K. Murray, M. Nagata, M. Popel, M. Popović, M. Shmatova, S. Steingrímsson, V. Zouhar, presented at the Proceedings of the Ninth Conference on Machine Translation, ed. by B. Haddow, T. Kocmi, P. Koehn, C. Monz, pp. 1–46, (https://aclanthology.org/2024.wmt-1.1/).

67. R. Rei, C. Stewart, A. C. Farinha, A. Lavie, en, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Conference Name: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2685–2702, (2026; https://aclanthology.org/2020.emnlp-main.213) (2020).

68. A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K.-P. Yang, L. Yu, L.-C. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S.-Q. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y.-C. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, Z. Qiu, presented at the, (2026; https://www.semanticscholar.org/paper/d2d84d56f730f81d276a02b48d5d44db5bde0b4a).

69. Gemma Team *et al.*, Version Number: 1, (2026; https://arxiv.org/abs/2503.19786) (2025).

70. B. Lester, R. Al-Rfou, N. Constant, en, presented at the Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3045–3059, (2026; https://aclanthology.org/2021.emnlp-main.243).

71. G. Su, H. Wang, J. Wang, W. Zhang, Y. Zhang, J. Pei, Version Number: 1, (2026; https://arxiv.org/abs/2510.21131) (2025).

72. M. Bañón, P. Chen, B. Haddow, K. Heafield, H. Hoang, M. Esplà-Gomis, M. L. Forcada, A. Kamran, F. Kirefu, P. Koehn, S. Ortiz Rojas, L. Pla Sempere,

G. Ramírez-Sánchez, E. Sarrías, M. Strelec, B. Thompson, W. Waites, D. Wiggins, J. Zaragoza, presented at the Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ed. by D. Jurafsky, J. Chai, N. Schluter, J. Tetreault, pp. 4555–4567, (2026; https://aclanthology.org/2020.acl-main.417/).

73. D. Hendrycks, K. Gimpel, *Gaussian Error Linear Units (GELUs)*, arXiv:1606.08415 [cs], June 2023, (2026; http://arxiv.org/abs/1606.08415).

74. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *LoRA: Low-Rank Adaptation of Large Language Models*, arXiv:2106.09685 [cs], Oct. 2021, (2026; http://arxiv.org/abs/2106.09685).

75. K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, presented at the Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 311–318, (https://doi.org/10.3115/1073083.1073135).

76. A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, *CoRR* **abs/1911.02116**, arXiv:1911.02116, (http://arxiv.org/abs/1911.02116) (2019).

77. M. Freitag, R. Rei, N. Mathur, C.-k. Lo, C. Stewart, E. Avramidis, T. Kocmi, G. Foster, A. Lavie, A. F. T. Martins, presented at the Proceedings of the Seventh Conference on Machine Translation (WMT), ed. by P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, M. Zampieri, pp. 46–68, (https://aclanthology.org/2022.wmt-1.2/).

78. S. H. Ho, *Evaluation of Finetuned LLMs in AMR Parsing*, arXiv:2508.05028 [cs], Aug. 2025, (2026; http://arxiv.org/abs/2508.05028).

79. J. V. Gysel, M. Vigus, J. Chun, K. Lai, S. Moeller, J. Yao, T. J. O'Gorman, A. Cowell, W. B. Croft, C.-R. Huang, J. Hajic, J. H. Martin, S. Oepen, M. Palmer, J. Pustejovsky, R. Vallejos, N. Xue, *KI - Künstliche Intelligenz* **35**, 343–360, (https://api.semanticscholar.org/CorpusID:235563506) (2021).

80. O. Abend, A. Rappoport, presented at the Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ed. by H. Schuetze, P. Fung, M. Poesio, pp. 228–238, (https://aclanthology.org/P13-1023/).

81. D. Deutsch, E. Briakou, I. R. Caswell, M. Finkelstein, R. Galor, J. Juraska, G. Kovacs, A. Lui, R. Rei, J. Riesa, S. Rijhwani, P. Riley, E. Salesky, F. Trabelsi, S. Winkler, B. Zhang, M. Freitag, presented at the Findings of the Association for Computational Linguistics: ACL 2025, ed. by W. Che, J. Nabende, E. Shutova, M. T. Pilehvar, pp. 12257–12284, ISBN: 979-8-89176-256-5, (https://aclanthology.org/2025.findings-acl.634/).

82. M. Esplà-Gomis, presented at the Beyond Translation Memories: New Tools for Translators Workshop, (https://aclanthology.org/2009.mtsummit-btm.6/).

83. Knight, Kevin, Badarau, Bianca, Baranescu, Laura, Bonial, Claire, Griffitt, Kira, Hermjakob, Ulf, Marcu, Daniel, O'Gorman, Tim, Palmer, Martha, Schneider, Nathan, Bardocz, Madalina, *Abstract Meaning Representation (AMR) Annotation Release 3.0*, Artwork Size: 153936 KB Pages: 153936 KB, Jan. 2020, (2026; https://catalog.ldc.upenn.edu/LDC2020T02).