Leuphana University Lüneburg

Master Thesis

# Text-to-SPARQL Generation with Reinforcement Learning: A GRPO-based Approach on DBLP

*Author:*
Jann Pfeifer
*Course of Study:*
Management & Data Science

*Supervisor:*
Prof. Dr. Ricardo Usbeck
*Secondary Supervisor:*
Dr. Debayan Banerjee

*A thesis submitted in fulfilment of the requirements for the degree of Master of Science*

*in the*

Artifical Intelligence and Explanability Research Group
Institute of Information Systems

December 10, 2025

# Contents

# List of Figures

# List of Tables

# Nomenclature

BERT Bidirectional Encoder Representations from Transformers

BGP Basic Graph Pattern

BPE Byte Pair Encoding

C4 Colossal Clean Crawled Corpus

CoT Chain-of-Thought

DoRA Decomposed Low Rank Adaptation

DPO Direct Preference Optimization

FNN Feed-Forward Neural Network

GAE Generalized Advantage Estimation

GPT Generative Pre-trained Transformer

GRPO Group Relative Policy Optimization

GRU Gated Recurrent Unit

ICL In-Context Learning

IR Information Retrieval

IT Instruction Tuning

KG Knowledge Graph

KGQA Knowledge Graph Question Answering

KL Kullback–Leibler

LLM Large Language Model

LM Language Model

LoRA Low Rank Adaptation

LSTN Long-Short-Term-Memory

MDP Markov Decision Process

MHA Multi-Head Attention

ML Machine Learning

MoE Mixture-of-Experts

NLG Natural Language Generation

NLP Natural Language Processing

OOV Out-of-Vocabulary

OWA Open World Assumption

PEFT Parameter Efficient Fine Tuning

PMF Probability Mass Function

PPO Proximal Policy Optimization

RDF Resource Description Framework

RDFS RDF Schema

RL Reinforcement Learning

RLHF Reinforcement Learning from Human Feedback

RNN Recurrent Neural Network

RoPE Rotary Position Embedding

SFT Supervised Fine Tuning

SPARQL SPARQL Protocol and RDF Query Language)

TD Temporal Difference

TRPO Trust Region Policy Optimization

URI Uniform Resource Identifier

VL Vision Language

W3C World Wide Web Consortium

# 1 Introduction

Knowledge graphs (KGs) provide a structured representation of entities and their relationships, and are widely used to organize and query complex domains such as healthcare, engineering, finance, and the scholarly literature [3]. For instance, the DBLP knowledge graph encodes publications, authors, venues, and temporal information about research output, and is used in a range of bibliometric and exploratory search tools. While SPARQL offers a flexible query language for such graphs, it requires users to understand the underlying schema and to adhere to its formal syntax. This makes direct interaction with KGs difficult for non-expert users and is the motivation for methods that translate natural language questions into executable queries.

Knowledge Graph Question Answering (KGQA) addresses this problem by mapping questions to logical forms or SPARQL queries that can be executed against a KG. Early systems approached this as a semantic parsing task and trained sequence models to translate questions into queries on datasets such as LC-QuAD [146] and QALD [114]. More recent work leverages Large Language Models (LLMs) either as supervised semantic parsers or as zero-shot and few-shot generators, sometimes in combination with retrieval and agentic tool use. These approaches have demonstrated that modern LLMs can achieve strong text-to-SPARQL performance, especially when large models and substantial amounts of supervision are available. Section 1.2 reviews these developments in more detail.

Despite this progress, several practical limitations remain. Many state-of-the-art Text-to-SPARQL systems rely on medium to large models and on sizable collections of gold queries, which may be expensive to annotate for new domains. Prompting large models with in-context examples can reduce the need for task-specific training, but often comes with high inference cost and requires careful prompt design. Reinforcement learning (RL) with verifiable rewards offers a complementary route. For tasks where model outputs can be executed, reward signals based on execution correctness or answer overlap can be used to fine-tune a model without relying on fully supervised query sequences. Recent work has shown that such methods can substantially improve small and medium-sized models on mathematical reasoning and text-to-SQL. In the KGQA setting, however, RL-based approaches are still relatively new and are often formulated as multi-step, agentic interaction with the KG rather than as zero-shot Text-to-SPARQL.

The scholarly domain poses additional challenges. Datasets such as DBLP-QuAD

[22] define text-to-SPARQL benchmarks over bibliographic KGs that involve diverse question types, temporal conditions, and compositional query patterns. Existing methods on DBLP-QuAD either fine-tune encoder–decoder or Transformer models with full SPARQL supervision, or adapt large LLMs via prompting and modular pipelines. There is little systematic evidence on whether a small open-weight model can be trained, using RL and reward functions, to act as an effective zero-shot Text-to-SPARQL system in this setting, and how its behavior compares to a strong supervised baseline at the same model scale.

This thesis investigates Group-Relative Policy Optimization (GRPO) [131] as a training method for comparatively small LLMs on zero-shot Text-to-SPARQL over the DBLP knowledge graph. Concretely, the experiments are based on the Qwen3-1.7B [163] instruction-tuned model and apply a GRPO-style training on DBLP-QuAD. The model is trained on a conversational prompt that includes the natural language question together with symbolic hints about relevant entities and relations, and is required to follow a chain-of-thought style output protocol with explicit `<think>` tags and a final SPARQL query. The experiments consider two GRPO variants, the first relying only on answer-level and structural rewards, whereas the second augments these with gold-query-based shaping terms. Both variants are compared to the unmodified zero-shot base model and to a supervised finetuning baseline trained with DoRA adapters [95] on gold SPARQL queries.

The evaluation focuses on multiple aspects of performance. Answer-level metrics such as exact match accuracy, macro-averaged F1 over result sets, and execution accuracy quantify how often the generated queries are both executable and produce a correct answer set. Category-wise F1 scores capture how models handle different SPARQL patterns, including simple single-fact lookups, disambiguation, compositional queries with unions or multiple intents, and negation-related constructs. Additional measures assess temporal questions and generalization to held-out template tuples, which probe the robustness of learned behavior beyond the templates seen during training. A qualitative error analysis complements these aggregate metrics by identifying recurring failure modes of GRPO and supervised finetuning. Taken together, these design choices give rise to a set of concrete research questions that guide the remainder of the thesis.

## 1.1 Research Questions

The work in this thesis is organized around the following research questions, which shape the experimental design and the interpretation of the results and also motivate

the selection and discussion of related work in Section 1.2.

- **RQ1:** How does Group-Relative Policy Optimization influence the performance of small instruction-tuned models on zero-shot knowledge graph question answering over DBLP-QuAD, relative to their zero-shot baseline and to a supervised finetuning baseline?
- **RQ2:** Is GRPO an effective training method for zero-shot Text-to-SPARQL with small language models when only natural language questions, answer sets, and symbolic hints are available, but gold SPARQL queries are not?
- **RQ3:** Which reward functions and reward configurations are most promising for improving Text-to-SPARQL performance of GRPO-trained small language models, and how do they trade off answer accuracy, execution reliability, and training stability?

Based on these questions, the thesis makes three main contributions. First, it presents a pipeline for applying small instruction-tuned LLMs to DBLP-QuAD, including preprocessing steps that normalize gold queries and answer sets, enrich entities and relations with human-readable schema information, and construct prompt templates tailored to chain-of-thought style Text-to-SPARQL generation [1]. Second, it provides a systematic empirical comparison between zero-shot, GRPO trained, and supervised DoRA-finetuned Qwen3-1.7B models on DBLP-QuAD, covering overall accuracy, category-wise behavior, temporal questions, and held-out template generalization. Third, it offers an initial analysis of multi-component reward design for Text-to-SPARQL with GRPO. In particular, the experiments show that GRPO can substantially improve over a weak zero-shot baseline and yields competitive template-level generalization, but that supervised finetuning on gold queries remains stronger overall under the studied conditions. The results also indicate that naive gold-query-based shaping does not necessarily improve answer accuracy and may interact with execution-based rewards in non-trivial ways.

The remainder of this thesis is structured as follows. Chapter 2 reviews background on large language models, knowledge graphs and SPARQL, KGQA, and policy optimization in reinforcement learning. Chapter 3 describes the methodology, including the DBLP-QuAD dataset and preprocessing pipeline, model selection, GRPO training logic, and evaluation metrics. Chapter 4 presents the experimental setup and reports quantitative results, error analyses, and ablation studies. Chapter 5 discusses

[1] The code for preprocessing, training, inference and evaluation is publicly available on Github: https://github.com/jannpf/dblp-grpo

the findings in light of the research questions, highlights limitations, and concludes with directions for future work.

## 1.2 Related Work

Early work on Knowledge Graph Question Answering (KGQA) framed the task as semantic parsing from natural language questions into executable logical forms or SPARQL queries. Since about 2018, many systems have used neural machine translation (NMT) architectures to map questions to SPARQL, typically assuming gold entity and relation linking and focusing on query structure prediction [165, 139]. Banerjee et al. [21] investigate "modern baselines" for SPARQL semantic parsing and compare pointer-generator networks, BART, and T5 on LC-QuAD 1.0 [146] and 2.0 [48] as well as Wikidata-based benchmarks, showing that T5 achieves state-of-the-art performance under gold linking. Complementary work explores structure-aware pretraining for Text-to-SPARQL. Qi et al. [118] propose TSET, a Triplet-Structure Enhanced T5 model that inserts an intermediate pretraining stage with triplet-order objectives between generic T5 pretraining and fine-tuning, outperforming other models on QALD-9 [114], QALD-10 [147] and LC-QuAD [146]. To better handle entities and relations that must be copied from the input, Hirigoyen et al. [65] introduce a copy mechanism for SPARQL NMT, demonstrating that ConvS2S-copy improves accuracy on several benchmarks, including LC-QuAD, by copying KB elements directly into the generated query. Other recent NMT-based approaches enhance ConvS2S encoders with multi-head attention and apply correction mechanisms, resulting in further gains in BLEU and macro-F1 scores compared to previous KGQA systems on QALD-9 and LC-QuAD [34].

With the advent of LLMs, more recent work uses pre-trained generative models for Text-to-SPARQL. While Banerjee et al. [21] already show that pre-trained T5 and BART act as strong supervised baselines, D'Abramo et al. [42] systematically study LLMs for Text-to-SPARQL, evaluating Mixtral, LLaMA-3, and CodeLlama on multiple KGQA benchmarks (e.g., QALD-9/10, LC-QuAD 1.0/2.0, Spider4SPARQL) and find that few-shot in-context learning (ICL) with retrieved demonstrations and beam search can match or surpass many fine-tuned baselines without task-specific training. Beyond single-shot prompting, Xu et al. [160] demonstrate that fine-tuning on a few-shot dataset can increase performance on Wikidata datasets. Further, a recent trend sees agentic approaches on Text-to-SPARQL, where an LLM iteratively plans and interacts with the KG to formulate a final query [137, 12]. Overall, these works show that well-designed prompting and agentic pipelines can yield strong Text-to-SPARQL performance, but they typically assume access to powerful large models

and do not directly address small-model training via RL.

Within the scholarly KGQA setting, several benchmarks and systems have emerged that are directly relevant to this thesis. DBLP-QuAD is a 10,000-question dataset over the DBLP knowledge graph, created by Banerjee et al. [22]. It is currently the largest scholarly KGQA dataset over bibliographic data and serves as the main benchmark for this thesis. The SciQA benchmark extends scholarly QA to the Open Research Knowledge Graph (ORKG). Auer et al. [11] release 2,565 question–SPARQL pairs over ORKG, with both manually crafted and template-generated questions covering many research fields and question types, and show that SciQA is challenging for next-generation QA systems. ORKG-QA, introduced by Jaradeh et al. [73], focuses on table-centric scholarly QA over ORKG comparison tables and was one of the first scholarly KGQA datasets. Several systems have been developed for these benchmarks, for instance, PSYCHIC, a neuro-symbolic framework proposed by Abi Akl [2], combines a DistilBERT-based question analyser with symbolic SPARQL construction and achieves competitive results in the scholarly QALD challenge. For SciQA, several studies evaluate LLMs via prompting and fine-tuning. Jiang et al. [79] show state-of-the-art results for SciQA by employing hybrid-prompt learning and including ontological information in the prompt. Meloni et al. [100] show that careful few-shot prompting and targeted fine-tuning can yield strong results on both SciQA and DBLP-QuAD, but also document systematic error patterns and robustness issues. Together, these works demonstrate that scholarly KGQA is feasible with supervised semantic parsing and LLM prompting, but they either rely on gold SPARQL supervision, assume strong linking components, or use large models.

Reinforcement learning for structured query generation has recently attracted attention, primarily in the Text-to-SQL literature. ConstrainedSQL [33] trains Text-to-SQL LLMs via constrained RL with multiple natural reward signals (execution accuracy and precision/recall-oriented constraints) and reports improvements over previous RL-trained and supervised baselines on the Spider and BIRD benchmarks. Reasoning-SQL [116] employs reward signals based on schema-linking, AI feedback, n-gram similarity, and syntax checks, achieving strong results with a 14B parameter model compared to larger baseline models. Stoisser et al. [136] further employs process-supervision based on synthesized reasoning chains, to enhance tabular reasoning accuracy in Text-to-SQL tasks. Across these studies, policy optimisation methods such as REINFORCE, Proximal Policy Optimization (PPO) and Group Relative Policy Optimisation (GRPO) are used as core RL algorithms. GRPO was introduced in DeepSeekMath, where Shao et al. [131] show that GRPO can substantially improve mathematical reasoning for a 7B model while requiring less memory

than PPO by removing the explicit value function and estimating advantages from groups of sampled trajectories. In the context of knowledge graphs and SPARQL, RL applications are more recent and often agentic. Zhang and Zhao [167] employ GRPO and process-supervision to fine tune LLMs for complex question answering over KGs, showing promising performance in selected datasets. However, process supervision has the risk of providing weak incentives for exploration, motivating further work on purely outcome-driven reward supervision. Vossebeld and Wang [149] fine-tune a compact LLM agent with GRPO to iteratively construct and refine SPARQL queries over Wikidata using outcome-only execution rewards, achieving around 50% answer accuracy on a curated LC-QuAD subset and a large improvement over zero-shot baselines. Further, very recent work like KnowCoder-A1 [35] and KG-R1 [134] (both only available as preprint) similarly adopt GRPO with outcome-based curriculum rewards to incentivize exploration in multi-step reasoning over KGs. These works underline that GRPO and related RL techniques are promising for structured query generation, but existing approaches mostly target SQL or iterative agent pipelines rather than zero-shot Text-to-SPARQL with small models.

Finally, there is growing interest in small or compact models for Structured Query Generation and KGQA. Karki et al. [83] use Parameter-Efficient Fine Tuning (PEFT) on small LLMs to perform Text-to-SQL, achieving competetive performance compared to larger baseline models. Nguyen et al. [105] explicitly explore RL fine-tuning for Text-to-SQL with models "of more conservative size" that fit on commodity hardware, arguing that RL can close part of the gap to very large models while remaining deployable. In KGQA, Huang et al. [72] demonstrate that sub-1B parameter models can be used effectively for KGQA by treating it as subgraph retrieval task. Jiang et al. [78] demonstrates an agentic framework for complex question answering over a KG, demonstrating that a 7B parameter model can outperform large baseline models in this setting.

Overall, the literature suggests that with appropriate supervision and architecture choices, small and mid-sized models can be competitive, which directly motivates the focus of this thesis on GRPO-trained 1.7B-parameter models over DBLP-QuAD.

# 2 Background

## 2.1 Large Language Models

The following section introduces Large Language Models (LLMs) with emphasis on the technical mechanisms that determine their behavior at training and inference time. A Language Model (LM) is a Machine Learning (ML) Model, that assigns a probability to words [80]. More formally, it defines a conditional probability distribution

$$P_\theta(x_T) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{<t})$$

that assigns a conditional probability to a word $x_T$ based on the previous ones $x_{<T}$. LMs are used for a variety of tasks, including Natural Language Generation (NLG), Token or Sequence Classification, Part of Speech (POS) Tagging and Machine Translation (MT) [32, 44, 121].

Early work on language models was dominated by n-gram models, which estimate the probability of a word based on a fixed number of preceding words ([129]; [74], [20], [75]). These models rely on the Markov assumption, limiting the context to a fixed window (e.g., bigram, trigram) to approximate the distribution above. The context length hence is effectively the number of words that the model can condition on. N-gram models evolved and improved over time ([57]) and remained the most commonly used language model throughout the 1990s.

The fundamental limitation of considering only a few words as context was addressed in the early 2000s with the introduction of neural network language models. The seminal work by Bengio et al. [26] proposed learning distributed word representations (or embeddings, see 2.1.1) jointly with a Feed-Forward Neural Network (FNN) that models conditional probabilities in terms of these representations, allowing the model to surpass state-of-the-art performance using larger context windows while keeping the computation demands manageable.

To address the fixed-context limitations of feedforward networks, Recurrent Neural Networks (RNNs) were introduced for language modeling [102]. RNNs do not suffer from the fixed-context length limitation like in the previous approaches, as they iterate over sequences while maintaining hidden states. However, RNNs suffer from vanishing and exploding gradients, which hinder their ability to model very long dependencies. As in other domains, Long-Short-Term-Memory (LSTM) models were also used for language modeling to mitigate this limitation, and later gener-

alized to Gated Recurrent Units (GRU) [66, 36]. These models incorporate gating mechanisms that allow relevant dependencies to be kept over longer sequences. These architectures became the state of the art in language modeling throughout the 2010s, enabling strong performance in speech recognition, translation, and text generation, just to name a few.

The landscape of language modeling changed again with the introduction of the transformer architecture [148], which replaces recurrence with self-attention mechanisms, which are described in more detail in the next chapter. Transformers enabled the development of pre-trained large language models (LLM) like BERT and GPT-3, which marked a transition from task-specific models to general-purpose language models.

In the following section, the transformer-based language model architecture is explained in greater detail.

### 2.1.1 Input Representation

Language models operate on numerical representations of text, which requires a mapping from natural language to discrete symbols that can be processed computationally. This process, referred to as tokenization, defines the vocabulary and segmentation strategy used to represent words, subwords, or characters as tokens.

Early approaches used entire words as tokens, but this caused vocabulary explosion and out-of-vocabulary (OOV) issues. Modern models therefore apply subword tokenization, which decomposes words into frequent character sequences that balance vocabulary size and coverage. Byte-Pair Encoding (BPE), originally introduced as a compression algorithm ([52]) and later adapted for neural translation [128], incrementally merges frequent symbol pairs to form a compact vocabulary. Related algorithms such as WordPiece [127] and SentencePiece [88] follow similar principles and remain standard in contemporary LLMs (see Section 2.1.7).

After tokenization, the input symbols can be represented as sparse matrix, each input symbol being denoted by an index over the vocabulary. Each token is then mapped to a continuous vector space via an embedding model, which maps discrete token identifiers to dense vector representations of fixed dimensionality. These embedding vectors capture syntactic and semantic regularities between tokens learned during training. Early distributed representation approaches include Word2Vec [103] and GloVe [113]. While effective, these static embeddings assign a single vector per word type and cannot reflect context-dependent meaning.

This shortcoming motivated the development of contextual embeddings, where the representation of a word depends on its surrounding words. Contextualization requires models capable of processing entire sequences and dynamically weighting relevant dependencies when calculating the embedding vector. Early solutions employed RNNs and their gated variants such as LSTMs to capture sequential context. However, these architectures required a sequential processing of the inputs, with each new hidden state depending on the states from previous computation steps. All the information is sequentially updated in a single vector. This is why these architectures struggled with long-range dependencies and parallelization of the process.

The subsequent introduction of the Transformer architecture [148] provided a more scalable and effective way to generate contextualized representations. The next section will therefore introduce the transformer architecture and the concept of self-attention mechanisms in more detail, and explain how it allows the processing of token representations to capture linguistic and semantic structure across sequences.

### 2.1.2 Transformer Architecture

#### 2.1.2.1 Self-Attention

At the core of the Transformer architecture is the idea of attention, first applied to neural machine translation by Bahdanau et al. [16]. Attention allows the decoder to focus on specific parts of the inputs when decoding, by scaling the hidden states of specific preceding tokens by attention values $\alpha$, which are learned during training.

In Transformers, self-attention generalizes this idea by allowing each token to attend to all other tokens within the same sequence, rather than relying on sequential dependencies as in recurrent models. While recurrent models process tokens one at a time and maintain a hidden state that evolves across time steps, self-attention computes contextualized representations for tokens in parallel. This parallelization not only removes the inherent sequential bottleneck of recurrent models but also enables direct modeling of long-range dependencies, since any token can attend to any other token regardless of the distance in the sequence [148].

Self-attention operates on three sets of vectors: queries ($Q$), keys ($K$), and values ($V$). These are linearly projected from the input embedding with three learned weight matrices: $Q = X\mathrm{W}_Q, K = X\mathrm{W}_K, V = X\mathrm{W}_V$ with $\mathrm{W}_Q, \mathrm{W}_K, \mathrm{W}_V \in \mathbb{R}^{\mathrm{d}\times\mathrm{d_k}}$. The attention weights are then obtained by comparing queries and keys through a scaled dot product, followed by a softmax normalization:

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}})V$$

where $d_k$ denotes the dimensionality of the key vectors, and scaling by $\sqrt{d_k}$ stabilizes gradients during training by preventing large dot-product magnitudes. The softmax function normalizes the scores into a probability distribution over all tokens. The resulting weighted sum of value vectors produces a context-aware representation for each token, integrating information from all other tokens in the sequence, with higher weight assigned to token representations that are more relevant to the given token context.

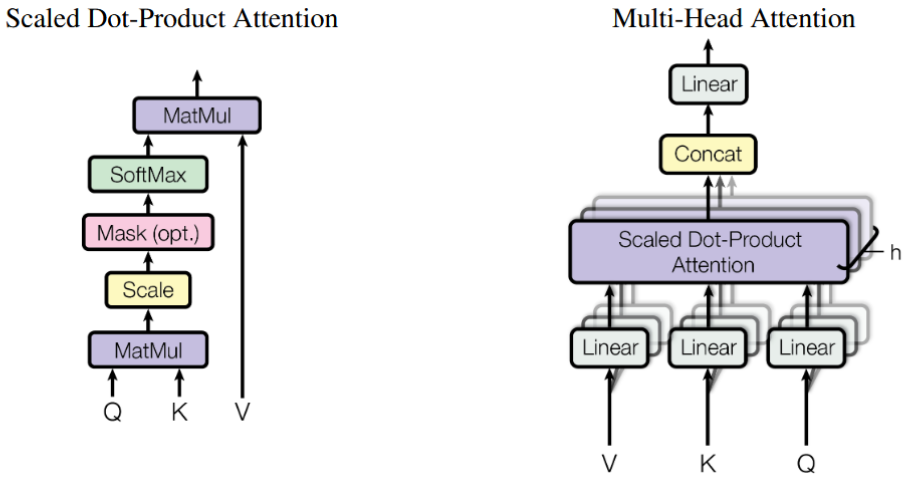


Figure 1: Scaled Dot-Product Attention (left) Multi-Head Attention (right) consists of several attention layers running in parallel. (from Vaswani et al. [148])

Instead of computing a single attention distribution, Transformers employ multi-head attention. Here, the input is projected into multiple subspaces, or *heads*, each with its own parameter matrices $W_Q^i$, $W_K^i$ and $W_V^i$. Each head independently performs scaled dot-product attention, allowing to focus on different aspects of the token relationships. The outputs of all $h$ heads are then concatenated and linearly transformed by learned $W^O$ to match input dimensionality:

$$MultiHead(Q, K, V) = concat(head_1, \ldots, head_h)W^O$$

with

$$head_i = Attention(QW_Q^i, KW_K^i, VW_V^i)$$

This mechanism allows the model to capture different types of relationships simultaneously within a single layer, with each head learning to focus on different aspects of the input.

#### 2.1.2.2 Positional Embeddings

Since self-attention allows the parallel computation of relationships between all tokens, it is inherently permutation-invariant, i.e. it lacks any inherent notion of sequence order. To enable the model to represent sequential dependencies, information about token positions has to be explicitly injected. This is achieved through positional encodings, which provide each token with a representation of its position within the sequence.

The original Transformer implementation by Vaswani et al. [148] introduced sinusoidal positional encodings. Each dimension of the position embedding corresponds to a sinusoid of different wavelengths, resulting in a linear relationship between the distance between tokens in the input sequence and the distance of vectors in the embedding space. The dimensionality of the positional embedding is chosen to match the input embedding dimensionality, so the two can be summed up before further processing.

In later models, various learned or relative positional encoding schemes were introduced to improve generalization and efficiency. Learned positional embeddings, as adopted in models like BERT [44], assign trainable vectors to positions rather than relying on a fixed sinusoidal pattern. More recent approaches such as Rotary Positional Embeddings (RoPE) [138] encode relative positional information directly within the attention mechanism. RoPE represents positions through a complex rotation applied to the query and key vectors before computing their dot products:

$$Q' = QR_\theta, K' = KR_\theta$$

where $R\theta$ is a rotation matrix whose angle depends on token position. This formulation preserves the geometric relationships between tokens and allows the model to compute relative distances implicitly, improving extrapolation to longer sequences and enhancing stability in large-scale autoregressive models (see Section 2.1.7).

#### 2.1.2.3 The Transformer Model

Figure 2 illustrates the transformer model, as initially proposed by Vaswani et al. [148].

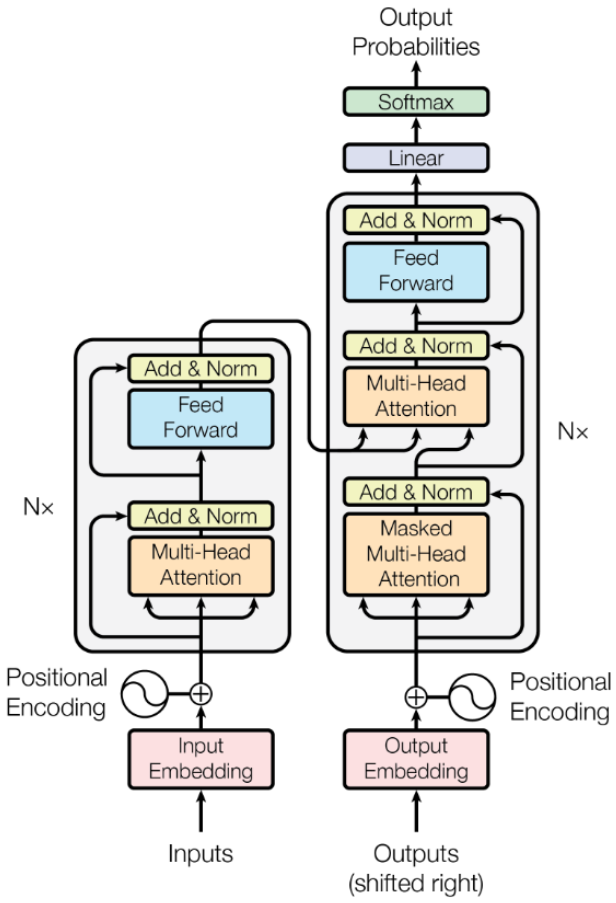


Figure 2: The Transformer Model (from Vaswani et al. [148])

It comprises an encoder-decoder structure where the encoder processes input sequences and the decoder generates output sequences (left and right side of Figure 2 respectively).

The inputs are transformed into embeddings using learned transformation matrices and summed with added positional encodings as described in section 2.1.2.2. Each of the encoder layers then sequentially applies multi-head self-attention over the input

tokens. In parallel, residual connections allow a stable optimization and gradient flow [63]. The outputs from the MHA are added to the residual stream and layer normalization is applied to prevent exploding or vanishing gradients from stacking many layers [13]. Following each self-attention operation, the transformer applies a position-wise FNN to each token representation independently. Originally with two layers and ReLu activation, more recent work has seen prominent usage of GeLu [64] as well as gated activations like SwiGLU [132] in the FNN.

In large-scale variants, this dense FNN is replaced by a Mixture of Experts (MoE) layer [133], where "experts" are identical feed-forward networks (typically 2–4 layer FNNs), and a learned router dynamically selects a sparse subset (e.g., 2 out of 128 experts) per token during pre-training. This router is a linear layer trained end-to-end with the model. The MoE architecture allows scaling efficiently to very large models while preserving per-token parallelism (see also Section 2.1.7).

This pattern repeats identically across all encoder layers, enabling parallel processing of input tokens while capturing long-range dependencies, resulting in dense contextualized representations of the input tokens. Newer variants include normalization before sublayers [159] and RMSNorm instead of standard layer normalization [166].

The decoder employs masked multi-head attention on the output representations and works in an autoregressive setting. Compared to the other MHA blocks, a causal mask zeroes attention here to future positions so that position $t$ can only attend to $[1, ..., t-1]$. Similar to the encoder, the output from the multi-head attention block is added to the residual stream and normalized. Next, the resulting stream is fed into another multi-head attention layer, connecting the queries derived from the outputs of the previous step to the keys and values from the encoder. This allows the output representations to directly attend to all the token representations from the input sequence. This is why this layer is referred to as the cross-attention layer.

After cross-attention, the FNN is applied as in the encoder, again with residual connections and normalization. This process is repeated for $N$ stacked heads, before the output layer then linearly projects the final token representations to vocabulary probabilities to predict the next token.

With this encoder-decoder setup, the output is iteratively generated by the decoder token-by-token, with cross-attention dynamically aligning to the encoder output during each decoding step.

This architecture demonstrated its effectiveness for machine translation tasks, and subsequently led to other models like T5 (Text-to-Text Transfer Transformer) [121].

This is a series of transformer based encoder-decoder models that are pretrained on large corpuses, and can be finetuned to various other NLP tasks such as abstractive summarization, text classification, natural language inference and regression, demonstrating the effectiveness of the architecture for text-to-text tasks in general.

In the following section, alternatives to the encoder-decoder structure are introduced.

### 2.1.3 Encoder-only and decoder-only models

As illustrated in the previous section, the Transformer architecture proved to be effective on various NLP tasks. Beside the original encoder-decoder approach, other variants have emerged that specialize on certain tasks respectively.

Encoder-only models consist solely of the encoder portion of the Transformer (i.e., stacks of self-attention and feed-forward blocks) without the auto-regressive decoder mechanism or cross-attention layers. A prominent example is BERT (Bidirectional Encoder Representations from Transformers) [44], which processes full input sequences bidirectionally (i.e., each token attends to all tokens in the sequence) and is trained with masked-token and next-sentence prediction objectives. Encoder-only models are very effective where the task is one of understanding or representation extraction (e.g., contextual representation generation, classification, named-entity recognition), rather than generation.

In contrast, decoder-only models employ only the decoder stacks of the Transformer, and are trained autoregressively to predict the next token given all prior tokens (including a prompt) These models omit the encoder and cross-attention mechanism, they treat the input prompt and generated tokens uniformly in the decoder stack, and are ideally suited for generative tasks (e.g., dialogue, story-generation, next-token prediction). An early and well-known instance is the GPT family (Generative Pre-trained Transformer), which demonstrated remarkable capabilities at free-form text generation, completion, in-context learning and more [119, 32].

In the following sections, the focus will be on decoder-only models, because this family of models is particularly relevant for this thesis.

### 2.1.4 Inference

At inference time, decoder-only models generate token by token under a causal mask (compare Section 2.1.2.3). When generating tokens from the resulting output probability distribution, the simplest way is to select the token with the highest probability, i.e.:

$$\hat{w}_t = \arg\max_{w \in V} P(w \mid \mathbf{w}_{<t})$$

This strategy, referred to as greedy decoding, has its limitations, as the most likely token at time $t$ might not lead to the overall most likely sequence after $t+1$. Beam search is an extension to greedy decoding that tries to mitigate this issue by maintaining several partial hypotheses across timesteps [98]. With a beam width of $k$, for each timestep $k$ different solutions are kept, and at the end, the sequence with the overall highest probability is chosen as the output.

However, this approach, like greedy decoding, is still deterministic over the inputs and leads to predictable and repetitive outputs, and comparatively high search error rates [69, 99]. Despite, beam search and its variants [51] remain widely used in Machine Translation, where models are typically trained and evaluated to produce a single high-likelihood translation for a given input [135].

Stochastic sampling strategies pose a widely used alternative to beam search, where the next token is drawn randomly from the model's predicted distribution over the vocabulary. However, if drawing directly from the distribution, the probability mass of the unlikely tokens is still significant due to the vocabulary size, which can lead to incoherent text.

Temperature sampling is a simple variant, where before the softmax is applied in the decoder (refer to Section 2.1.2.3), the logits $z_w$ are rescaled by a temperature parameter $\tau > 0$ [5]. With this parameter it is therefore possible to control the sharpness of the distribution and the original softmax simply becomes

$$\text{softmax}_\tau(\mathbf{z})_i = \frac{\exp\left(\frac{z_i}{\tau}\right)}{\sum_{j=1}^{K} \exp\left(\frac{z_j}{\tau}\right)}$$

Sampling directly from the full vocabulary can still select very low probability tokens. Truncation-based sampling therefore restricts sampling to a subset of likely candidates. In top-k sampling, only the $k$ most probable tokens at each timestep are retained and renormalized, and the next token is sampled from this restricted set [49, 68].

Nucleus sampling or top-p sampling generalizes this idea by selecting the smallest set of tokens, whose cumulative probability mass exceeds a threshold $p$. The distribution is renormalized and a sample drawn from this dynamically sized "nucleus" [69]. Empirical studies show that these sampling-based decoding methods can substantially

reduce repetition and improve the trade off between quality and diversity compared to greedy or beam search, particularly in open ended generation tasks [69, 50, 43, 10].

In the next chapter, the training process of Transformer based Large Language Models is illustrated.

### 2.1.5 Training

This section explains the training process of a Large Language models in its typical three phases:

- Pre-Training
- Supervised Fine Tuning / Instruction Tuning
- Alignment / Preference Optimization

#### 2.1.5.1 Pre-Training

In the pretraining stage the base capabilities of LLMs, to estimate word probabilities and generally generate text, are established. To achieve this, it is optimized to maximize the likelihood of each next token given its left context in a self-supervised setting on large textual datasets. More formal, the training objective is the minimization of the cross-entropy loss between the model's output distribution and the one-hot target vector for the correct token. Therefore, with $x_t = y_t$ the loss becomes:

$$\mathcal{L}_{\mathrm{CE}}(\theta) = -\log p_\theta \left( x_t \mid x_{<t} \right)$$

The loss is computed under a causal mask and is typically minimized with optimizers like Adam [85] or AdamW [97] and with teacher forcing. In teacher forcing, the model is always conditioned on the gold prefix during training rather than its own sampled outputs, so the input at step $t$ is the true token sequence $x_{<t}$ and not the earlier model predictions. This well-known approach from recurrent training [157] stabilizes optimization by avoiding exposure to previous mistakes but induces a train–test exposure mismatch that later work addresses with alternatives such as scheduled sampling [25] or professor forcing [89].

When evaluating the model performance over long sequences, using the log-likelihood is impractical, as the result would be depending on the length of the sequence. Perplexity provides a length normalizing metric of this objective on held-out text

and is defined as the exponentiated average cross-entropy [76] over a sequence of length $n$:

$$\text{Perplexity}_\theta(w_{1:n}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log P_\theta(w_i \mid w_{<i})\right)$$

Since perplexity has an inverse relationship with probability, lower perplexity indicates that the model assigns higher probability to the test tokens and therefore is considered a better model for the given data.

Modern LLMs are trained on very large, heterogeneous text corpora that combine broad web crawls with more curated sources. A common backbone is Common Crawl [39], a petabyte-scale open crawl of the public web. Because raw Common Crawl suffers from duplication, representativeness gaps, and problematic content [14], it is typically heavily filtered and deduplicated before use.

For English, the Colossal Clean Crawled Corpus (C4) provides a cleaned and deduplicated Common-Crawl-derived corpus and has been extensively used in large text-to-text pretraining. It applies language identification, heuristic quality filters, boilerplate removal, and large-scale deduplication [121]. The multilingual variant mC4 uses the same approach across more than one hundred languages for models like mT5 [161]. Independent audits of C4 document domain skew, retained noise, and filtering side effects, which motivates more transparent documentation and data sheets [45].

Alternative pipelines built directly on Common Crawl, notably CCNet and the resulting CC-100, extract higher-quality monolingual data in many languages and underpinned multilingual encoders such as XLM-R [155, 40]. OSCAR offers a complementary multilingual web corpus with document-oriented cleaning that targets generative pretraining needs [1]. A newer line of work aims to retain web scale while improving cleanliness: RefinedWeb for Falcon and FineWeb mine many Common Crawl snapshots with stricter heuristics and stronger deduplication, showing that carefully filtered web-only data can match or exceed mixed-source baselines at scale [110, 111].

To balance web style with more formal prose and encyclopedic facts, many mixtures add Wikipedia contents, for example with the Wiki-40B corpus [61] and long-form book contents [169]. This combination was popularized early with BERT [44].

To increase domain coverage, composite corpora aggregate different content sources. The Pile ($\approx$ 825 GiB text) combines 22 sources including academic papers, code,

Wikipedia and more, and was explicitly designed to boost cross-domain generalization relative to raw Common Crawl derivatives [54]. Across these families, data curation and governance significantly affect outcomes. Deduplication and filtering reduce memorization, lower train–test overlap resulting in better evaluation validity, and can improve sample efficiency [92].

#### 2.1.5.2 Supervised Fine Tuning

While the pretraining stage establishes the capability of the LLM to generate text, instruction tuning is used to make LLMs better at following user instructions. The training process is the same as in pretraining, with a next-token-prediction objective and cross-entropy loss with teacher forcing to update the model's parameters (see previous section 2.1.5.1).

But unlike continued pre-training [62], where the pretraining process is continued on a domain specific corpus to provide it with more information on a specific domain, instruction tuning is regarded as Supervised Fine-Tuning (SFT), as it follows the objective to learn the helpful answer to an instruction based on specialized instruction datasets [80].

These instruction datasets compile input-output pairs for many different tasks, that are reformulated into instruction formats, often with multiple prompt templates per task and sometimes with examples to simulate few-shot settings. The approach was introduced with FLAN [153], which showed that scaling the number of instruction-formatted datasets and model size substantially improves zero-shot and generalization performance. Broader task collections such as T0 and Super-NaturalInstructions demonstrated that training on prompted multi-task data and on declarative instructions further boosts generalization performance, also for tasks unseen during training [123, 150, 104, 38].

Notably, LIMA [168] showed that a small, carefully curated set of high-quality training examples can be enough to teach formatting and response style, with human preference parity against much larger models in a significant fraction of cases, showing that targeted instruction tuning is complementary to large-scale pre-training.

#### 2.1.5.3 Parameter Efficient Fine Tuning

Finetuning an LLM to a specific task or domain is very expensive in terms of computational power and memory. Contemporary LLMs consist of Billions of parameters, and to update them with full SFT, the weight updates require optimization and

backpropagation through many layers, which leads to the same infrastructure requirements as during pretraining.

To address this, Parameter Efficient Fine Tuning (PEFT) methods try to reduce the required footprints by limiting the number of parameters that are updated during finetuning. Early approaches suggested adapters, where small Neural Networks are placed and trained in between the layers of the otherwise untouched LLM [122, 70, 24].

To address several limitations in adapters, for example the added inference latency and lack of parallelism, Low-Rank Adaptation (LoRa) replaces full weight updates with a low-rank reparameterization [71].

In LoRa, the finetuned weight matrix $W'$ is represented as

$$W' = W_0 + \Delta W = W_0 + BA$$

with $W_0$ being the frozen initial weight matrix, and the update being constrained to $\Delta W = BA$ with $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, rank $r \ll \min(d, k)$. Only the lower dimensional $A$ and $B$ are trained, which significantly reduces trainable parameters.

LoRA can be placed in attention projections $W_q, W_k, W_v, W_o$ and in FNNs, and the adapters can be merged into $W_0$ for inference, so latency is unchanged. In the original study, LoRA matched or exceeded full fine-tuning on several NLP tasks while cutting memory and checkpoint size by large factors, for example adapting only query and value projections with small rank [71].

Weight-Decomposed LoRA (DoRA) [95] addresses the gap that sometimes remains between LoRA and full fine-tuning, that was previously ascribed to the difference in rank. DoRA decomposes the weight matrix $W$ into:

$$W = m\frac{V}{\|V\|_c} = \|W\|_c\frac{W}{\|W\|_c}$$

with $m$ being the magnitude vector and $V$ being the direction matrix, which is normalized column wise, so that $m$ scales each unit vector in $V$, effectively isolating the direction and the magnitude component.

For the weight updates, LoRa is applied to the direction matrix $V$, scaled by the learned magnitude vector $m$:

$$W' = \underline{m}\frac{V + \Delta V}{\|V + \Delta V\|_c} = \underline{m}\frac{W_0 + \underline{BA}}{\|W_0 + \underline{BA}\|_c}$$

This decoupling allows the two components to be treated separately, which resembles the learning pattern of SFT better than LoRa. The number of updated parameters remains comparatively small and no inference overhead is added, since the components are merged after training. DoRA consistently outperformed LoRA across a variety of tasks [95].

#### 2.1.5.4 Preference Optimization for LLM Alignment

Pretraining and instruction tuning produce capable models, but outputs can remain unhelpful, untruthful, or unsafe. Empirically, pretrained LLMs can be prompted into toxic or biased generations and may mimic common misconceptions, motivating an additional stage that aligns behavior with human intentions and safety constraints [56].

Alignment is typically formulated as the third stage after pretraining and SFT. Earlier work in Reinforcement Learning (RL) showed that systems can learn to interact with an environment using human preferences [37]. This approach was transferred to LLMs in form of Reinforcement Learning from Human Feedback (RLHF) with the development of InstructGPT [109].

Instead of learning from next-token prediction, the objective in RLHF is shaped by preferences over model responses. Human Annotators compare and rate alternatives, providing signals about helpfulness, harmlessness, and general quality. These signals can come from direct human preference labeling, or from AI-generated judgments as a scalable proxy [91].

The first step in RLHF, following the process described in the previous section, is to apply SFT on a curated set of prompt-answer pairs, where human annotators demonstrate the desired model behavior. In the second step, for any given sample in the prompt dataset, multiple answers from the LLM are sampled. A human annotator then ranks the different responses from worst to best. This ranking is used to fit a dedicated reward model, that learns to predict human preferences over a set of responses. Finally, the model is optimized against this reward model, as it tries to maximize the reward it receives for its outputs.

For this optimization, Reinforcement Learning (RL) techniques like Proximal Policy Optimization (PPO) are employed, that aim at reward maximization under a constraint that prevents the model from deviating too far from the pretrained base model. InstructGPT shows that PPO-style RLHF over a reward model substantially improves human-preference win rates [109]. PPO and other variants are explained

in greater detail in the dedicated Chapter 2.3.

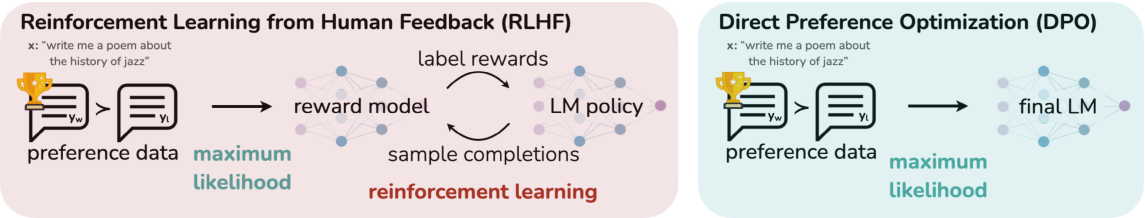


Figure 3: RLHF vs. DPO (from Rafailov et al. [120])

However, the additional reward model and the need for sampling during training lead to high computational cost. Direct Preference Optimization (DPO) [120] addresses this limitation by formulating preference alignment as a maximum likelihood problem without the need for an explicit reward model or RL in the training loop. Given pairwise preferences, DPO optimizes a closed-form objective, enabling supervised-style gradient updates with a binary-cross entropy loss. Figure 3 illustrates the comparison between RLHF and DPO. DPO was shown to often match or exceed PPO-based RLHF in control of sentiments, single-turn dialogue and summarization, while being easier to implement and train [120].

### 2.1.6 In context learning

The training process described in the previous sections use parameter updates to change the model's behavior. In addition to parameter updates, LLMs exhibit a form of adaptation at inference time that is referred to as In-Context Learning (ICL). Here, the model's input prompt is viewed as a learning signal [80], that contains task instructions and possibly a few input–output examples and then predicts outputs for new inputs, without any change to its parameters. The GPT-3 study popularized this phenomenon by showing that sufficiently large autoregressive models can perform a wide range of tasks in a zero-, one-, or few-shot prompting setup, sometimes approaching the performance of explicitly fine-tuned models [32].

At inference time, the model receives a sequence that includes several labeled examples $x_i, y_i$ of a task with a new query $x_{\text{test}}$. The model then generates a continuation that should correspond to $y_{\text{test}}$. All computation is a single forward pass of the transformer under the same causal masking that is used for standard language modeling. The distinction between zero-shot, one-shot and few-shot prompting lies only in how

many demonstrations are included in the prompt. In all ccases,the parameters remain fixed, and adaptation happens purely through attention to the in-context examples. Brown et al. [32] observe that performance across several NLP benchmarks improves both with model size and with the number of in-context examples, even though the model was never explicitly trained on the respective tasks.

Subsequent work formalizes and extends these findings. Garg et al. [55] analyze what function classes transformers can learn in context and show that, under suitable training, they can learn linear functions and certain more complex hypotheses from examples in the prompt. Further, it was shown that LLMs can select from and apply a broad class of ML algorithms via ICL [19].

Several theoretical accounts of in-context learning have been proposed. Xie et al. [158] views ICL as a form of implicit Bayesian inference. With LLMs modeling the pre-training distribution, the task of generating a prompt completion can be viewed as computing the posterior distribution over possible completions by updating the prior distribution with the likelihood from the in-context examples, effectively applying the appropriate concept learned during pre-trained to the task at hand.

Mechanistic interpretability work connects these abstract views to specific circuit structures in transformers. Olsson et al. [108] identify induction heads, attention heads that implement a simple copy-pattern algorithm of the form $[A][B] \dots [A] \rightarrow [B]$, and show that these heads arise at the same point in training where ICL performance improves. They argue that induction heads are a core mechanism for many forms of ICL [108].

ICL also underlies recent prompting methods that target reasoning tasks. Chain-of-thought (CoT) prompting provides demonstrations with intermediate reasoning steps, which substantially improves performance on arithmetic and symbolic reasoning benchmarks in sufficiently large models [154]. Kojima et al. [86] show that even a simple zero-shot instruction such as "Let us think step by step" can elicit similar behavior, indicating that models can infer a reasoning style from textual cues alone, again without parameter updates.

From the perspective of the preceding training section, in-context learning can be viewed as a consequence of the pretraining objective. By optimizing next-token prediction on heterogeneous, long-range coherent data, the model internalizes a rich set of tasks and learning strategies. At inference time, attention over the prompt then acts as a fast, temporary adaptation mechanism that complements slower parameter updates achieved through supervised fine-tuning or preference optimization. Compared to these training-time stages, in-context learning has the advantage of

requiring no additional optimization or storage of task-specific parameters, but it is often sensitive to prompt design, example ordering and context length, and remains an active area of theoretical and empirical research [46].

### 2.1.7 Examples

In this section, selected contemporary LLMs are introduced as examples and their architecture is briefly outlined.

#### 2.1.7.1 LLaMA (Meta)

Meta AI develops and releases the Llama family of large language models as openly available foundation and chat models under a community license, beginning with LLaMA (2023) [144], followed by Llama 2 (2023) [145], Llama 3/3.1 (2024) [47], and most recently Llama 4 (2025) [101]. The series uses decoder-only Transformers trained with next-token prediction and instruction tuning.

Key architectural choices in LLaMA and Llama 2 include a SentencePiece BPE tokenizer $\approx 32k$ vocabulary), RMSNorm pre-normalization, SwiGLU feed-forward blocks, and rotary positional embeddings (RoPE). Llama 2 increases context length to 4k tokens and introduces grouped-query attention (GQA) in larger variants to reduce KV-cache size and improve inference throughput. Pre-training uses only publicly available data, with LLaMA reporting a mixture covering Common Crawl, C4, GitHub, Wikipedia, Books, arXiv and StackExchange. Llama 2 scales the corpus to $\approx 2T$ tokens and describes a post-training pipeline of supervised fine-tuning followed by RLHF (rejection sampling and PPO) for dialogue alignment [144, 145].

The newest dense generation, Llama 3/3.1, retains this backbone while making targeted changes. It adopts a 128k-token vocabulary tokenizer (constructed by combining 100k tokens from tiktoken with 28k non-English tokens) to improve compression and multilingual coverage. The models use RoPE with a larger base frequency for long-context stability, RMSNorm, dense SwiGLU FFNs, and GQA with eight KV heads. The 8B/70B/405B parameter models use 32/64/128 attention heads respectively, and are pre-trained first at 8k context then continued-pre-trained up to 128k context. Llama 3 scales data substantially, reporting $\approx 15T$ tokens overall, with curriculum to extend context and specific up-/down-sampling of sources. Alignment switches to multiple rounds of SFT and DPO, with tool-use integration and safety-focused data curation [47].

Llama 4 introduces a Mixture-of-Experts (MoE) design with native multimodality

and extreme context lengths. Public materials describe two released variants, Scout ($\approx 109B$ total with 16 experts) and Maverick ($\approx 400B$ total with 128 experts), each activating $\approx 17B$ parameters per token. They are reported to support very long contexts via an interleaved scheme that mixes "No-position-encoding" (NoPE) layers with RoPE layers, sometimes referred to as an iRoPE architecture, together with chunked attention in the RoPE layers and attention-temperature scaling to mitigate probability dilution at long sequence lengths. Official model cards and integration notes cite pre-training on $\approx 22T - 40T$ tokens, multilingual and multimodal inputs, and instruct models tuned for 1M (Maverick) and up to 10M tokens (Scout) context windows [101].

Across generations, Llama models tracked the contemporaneous state of the art: LLaMA and Llama 2 established competitive open-weight baselines in general knowledge and reasoning while trailing the strongest closed models of 2023 [144, 145]. Llama 3 substantially narrowed this gap, showing strong performance in multilingual understanding, coding, and math with instruction-tuned variants competitive with leading systems available in 2024 [47]. Early public results and community evaluations for Llama 4 indicate parity with frontier-class models on broad instruction following, tool use, and long-context reasoning, particularly in multimodal and extended-context settings [101].

#### 2.1.7.2 Qwen (Alibaba)

Qwen is an open-weight family of foundation models developed and released by Alibaba Group's Qwen Team, spanning base and instruction-tuned models, specialized math and code variants, and multimodal Vision-Language (VL) models. The series begins with Qwen (2023) [17], proceeds through Qwen2 (2024) [162] and Qwen2.5 (2024–2025, including 1M-token long-context and VL lines) [164, 18], and most recently Qwen3 (2025) [163] which extends the family to dense and MoE models up to 235B parameters under Apache-2.0 licensing. All generations are trained with next-token prediction and instruction tuning.

Architecturally, modern Qwen models use a BPE tokenizer shared across sizes, with Qwen2 reporting a common vocabulary of 151,643 regular tokens plus 3 control tokens. Positional information is provided by RoPE with a larger base frequency for long-context stability, combined with YARN length scaling [112]. Qwen2 further introduces Dual-Chunk Attention to segment very long inputs while preserving cross-chunk relations. Normalization is RMSNorm in a pre-norm stack, and the FNNs use SwiGLU for activations. Attention moves from standard multi-head to grouped-

query attention across sizes, reducing KV-cache cost, with explicit query and KV head counts given per configuration. The series also includes a Mixture-of-Experts (MoE) line for the large flagship models. Typical context lengths grew from 4k to 32k during Qwen2 pre-training, with length-extrapolation to 131k via YARN. Qwen2.5 expanded long-context training in both pre-training and post-training, and Qwen2.5-1M reports 1M-token context together with sparse attention and chunked-prefill optimizations. Qwen3 retains the RMSNorm+SwiGLU+RoPE+GQA recipe while scaling to a 235B-parameter MoE with $\approx 22B$ activated per token and adds a unified "thinking" and "non-thinking" inference mode [162, 163, 164].

Training data scale and recipe evolve across generations. Qwen2 trains on a high-quality multilingual mixture emphasizing code and mathematics, scaled to $\approx 7T$ tokens, with an explicit long-context continued-pre-training phase from 4,096 to 32,768 tokens and length-extrapolation techniques for up to 131,072 tokens. Qwen2.5 increases the pre-training corpus to $\approx 18T$ tokens with staged mixtures and expands post-training to $\approx 1M$ supervised examples plus multi-stage RL that combines DPO and online GRPO (see later Section 2.3.3.3). Qwen2.5-1M details long-context data synthesis, progressive pre-training, long-response SFT, and deployment-oriented inference changes for 1M tokens. Qwen3 further scales pre-training to $\approx 36T$ tokens, broadens multilingual coverage to 119 languages and dialects, and combines reasoning-mode and standard instruction-following in a single model family. Post-training across generations uses SFT plus preference-based alignment, moving from RLHF in early Qwen to SFT+DPO in Qwen2, and to larger, staged SFT and RL pipelines in Qwen2.5 and Qwen3 [162, 163, 164].

Qwen and Qwen2 provided competitive open-weight baselines across general language understanding, multilingual tasks, coding and math while trailing the strongest closed models of their time. Qwen2.5 narrowed this gap with improved long-context processing and stronger alignment, and Qwen2.5-1M demonstrated practical million-token handling. Results for Qwen3 show clear improvements over previous generations, with the model family outperforming other open-weight systems of comparable size, including the corresponding Llama models, across reasoning, coding, math and multilingual benchmarks. The smaller Qwen3 variants already surpass prior open-source baselines in their size class, and the larger configurations achieve competitive performance relative to leading proprietary models, which the report attributes to the expanded training corpus and the mixture-of-experts architecture [17, 162, 163, 164].

## 2.2 Knowledge Graphs and SPARQL

### 2.2.1 The Semantic Web and Knowledge Graphs

This research is situated within the architectural framework of the Semantic Web, originally envisioned by Berners-Lee et al. [29] as an extension of the existing World Wide Web. While the traditional Web consists of unstructured documents designed for human consumption, the Semantic Web aims to create a web of data that can be processed and interpreted automatically by machines. To achieve this, the architecture relies on a stack of standardized technologies, that provide common formats for data interchange, schema definition, and querying [28]. This architecture is often referred to as "semantic web layer cake" and is depicted on a high level in Figure 4.

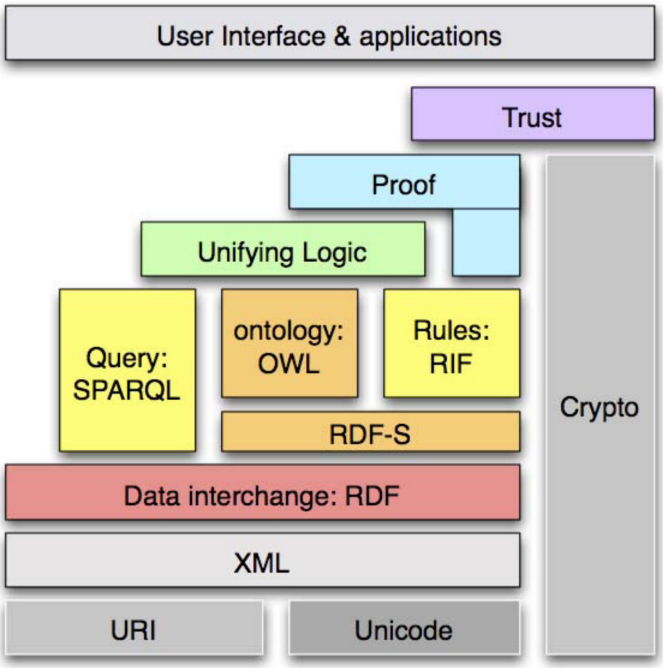


Figure 4: The semantic web layer cake architecture diagram. (from Berners-Lee and Swick [28])

Closely related to this architecture are the Linked Data principles, which establish best practices for publishing structured data. These principles dictate that data should be published using standard Web protocols such as HTTP and uniform resource identifiers (URIs) to create a global, interconnected data space where entities

link to one another across different datasets [30].

Within this technical ecosystem, Knowledge Graphs (KGs) have emerged as a central paradigm for data integration and knowledge representation [107]. Although definitions vary, the community has converged on the understanding that a knowledge graph is a graph-structured representation of entities and their relations, grounded in an ontology that defines the semantics of the data [67].

Formally, a knowledge graph is defined as a directed labeled graph $G = (V, E, L)$. In this definition, $V$ represents a set of vertices (entities), $L$ represents a set of edge labels (relations), and $E \subseteq V \times L \times V$ represents the set of edges connecting these vertices.

A critical characteristic of knowledge graphs on the Semantic Web is that they operate under the Open World Assumption (OWA). In traditional relational databases, systems typically operate under a Closed World Assumption, where any information not present in the database is presumed to be false. In contrast, the OWA dictates that the absence of a statement does not imply falsity, but implies that the information is currently unknown [15]. This assumption is essential for the decentralized nature of the Web, as it allows graphs to be extended incrementally with new facts from heterogeneous sources without creating logical contradictions with existing information.

### 2.2.2 The RDF Data Model

The Resource Description Framework (RDF) is the standard abstract data model for the Semantic Web as defined by the World Wide Web Consortium (W3C). RDF breaks knowledge down into discrete assertions known as triples. An RDF triple is an ordered 3-tuple $(s, p, o)$, which corresponds to a simple sentence structure comprising a subject $s$, a predicate $p$, and an object $o$ [41].

The components of an RDF triple are drawn from three disjoint sets: Internationalized Resource Identifiers (IRIs) denoted by $\mathcal{I}$, Literals denoted by $\mathcal{L}$, and Blank Nodes denoted by $\mathcal{B}$. The subject of a triple must be an element of the union $\mathcal{I} \cup \mathcal{B}$. The predicate must be an element of the set $\mathcal{I}$. The object can be an element of the union $\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ .

Internationalized Resource Identifiers (IRIs) are the core mechanism for identification in RDF. They serve as global, unique names that allow entities to be referenced unambiguously across different datasets. For example, an IRI might uniquely identify a specific person, distinguishing them from all other individuals with the same name.

Blank Nodes act as existential variables representing resources that exist but do not require a global identifier. They are locally scoped to a specific graph and are frequently used to model complex n-ary relationships or container structures where naming the intermediate node is unnecessary [106].

Literals represent concrete data values and may only appear in the object position of a triple. RDF distinguishes between plain language-tagged strings (used for natural language text) and typed literals. A typed literal consists of a lexical form (a string) and a datatype IRI. RDF relies on the XML Schema (XSD) type system to define the value space for these literals, supporting standard types such as integers, booleans, and dates. This formal typing is critical for semantic processing because it allows a query engine to interpret the character string "1990" and the integer value 1990 as distinct entities, thereby enabling valid arithmetic and comparison operations during query execution [41].

Figure 6 shows a simple example of a RDF graph based on the four triples given in Figure 5.

```
<#js> rdf:type foaf:Person .
<#js> foaf:name "John Smith" .
<#js> foaf:workplaceHomepage <http://univ.com/> .
<http://univ.com/> rdfs:label "University" .
```

Figure 5: Example RDF triples (from Tomaszuk and Hyland-Wood [143]).

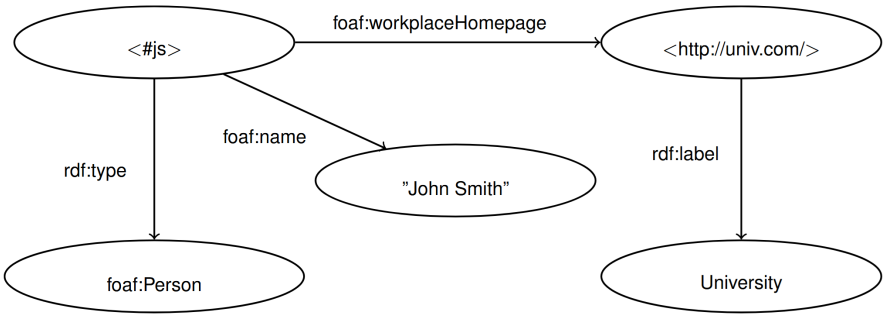


Figure 6: A Resource Description Framework (RDF) graph with four triples. (from Tomaszuk and Hyland-Wood [143])

### 2.2.3 RDF Schema (RDFS)

While RDF provides the structure for expressing data, it does not inherently provide the vocabulary to describe the meaning of that data. RDF Schema (RDFS) provides this semantic extension. It introduces a standardized vocabulary for describing groups of related resources (classes) and the relationships between them (properties) [31]. RDFS is characterized as a lightweight ontology language, balancing expressivity with computational efficiency.

A central feature of RDFS is its ability to model hierarchies. `rdfs:subClassOf` is a property that allows the definition of a subset relationship between two classes. In set-theoretic terms, if a class $C_1$ is defined as a subclass of $C_2$ (written as the triple $(C_1, \texttt{rdfs:subClassOf}, C_2)$), it implies that the set of all instances of $C_1$ is entirely contained within the set of instances of $C_2$. Consequently, if an individual $x$ is asserted to be an instance of $C_1$, it can be inferred that $x$ is also an instance of $C_2$. For example, if "Journal Article" is a subclass of "Publication", any resource typed as a "Journal Article" is implicitly also a "Publication" [31].

Similarly, the property `rdfs:subPropertyOf` defines a specialization hierarchy for relations. If a property $P_1$ is a subproperty of $P_2$, then any two resources connected by $P_1$ are necessarily connected by $P_2$.

RDFS also supports the definition of semantic constraints via domain and range declarations. A triple $(P, \texttt{rdfs:domain}, C)$ states that the property $P$ is associated with the class $C$. Specifically, it implies that any resource appearing as the subject of a triple with predicate $P$ must belong to class $C$. The property `rdfs:range` functions similarly for the object of the triple. In RDFS, domain and range declarations act as inference rules. If a dataset contains a triple that seemingly violates a domain declaration, the system does not reject the data as invalid. Instead, it accepts the data and infers the necessary class membership for the subject to satisfy the constraint [8]. While the computational complexity of full RDFS entailment is NP-complete [142], most practical triple stores implement rule-based fragments that allow for tractable and scalable reasoning [6].

### 2.2.4 Querying with SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is the W3C standard recommendation for querying and manipulating RDF data. Unlike SQL, which is designed to query rectangular tables in relational databases, SPARQL is designed to query the directed graph structure of RDF through a process known as graph pattern

matching [115, 117].

The fundamental unit of a SPARQL query is the Basic Graph Pattern (BGP). A BGP acts as a template consisting of a set of triple patterns. These triple patterns resemble RDF triples, but any component (subject, predicate, or object) can be replaced by a variable, typically denoted with a question mark (e.g., `?author`). Evaluating a BGP involves finding a set of solution mappings. A solution mapping $\mu$ is a partial function that maps variables in the query to actual RDF terms in the dataset. A mapping is considered a solution if, when the variables in the BGP are replaced by their mapped values, the resulting set of triples exists within the target graph [7, 117].

SPARQL 1.1 extends this basic matching mechanism with an algebra of operators to handle more complex retrieval requirements. The `FILTER` operator restricts the set of solutions based on boolean expressions. This allows users to apply constraints such as numeric comparisons (e.g., filtering for years greater than 2020) or regular expression matching on string literals. The `OPTIONAL` operator is designed to handle the semi-structured and often incomplete nature of knowledge graphs. It attempts to extend the current solution mapping with additional patterns. Crucially, if the optional pattern cannot be matched, the solution is not discarded; instead, the variables in the optional part are simply left unbound. This functions similarly to a "Left Outer Join" in SQL. The `UNION` operator enables disjunction, allowing the query to merge results from alternative graph patterns [9].

The language supports four distinct result forms. `SELECT` returns a table of variable bindings and is the standard format used for Question Answering tasks. `CONSTRUCT` returns a valid RDF graph by populating a user-defined template with the query solutions, effectively transforming the graph structure. `ASK` returns a simple boolean value indicating whether a pattern match exists, while `DESCRIBE` returns an implementation-dependent graph describing a resource. Additionally, SPARQL 1.1 includes aggregation functions such as `COUNT`, `SUM`, and `AVG`, as well as grouping modifiers like `GROUP BY`. These features allow queries to compute statistical summaries directly over the graph structure rather than simply retrieving raw facts [9].

A simple example SPARQL query, based on the graph from Figure 6, is given in Listing 1. The query retrieves the name associated with the entity `<#js>` and the label of the organization specified as its workplace.

```
SELECT ?name ?orgLabel
WHERE {
  <#js> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
```

```
        <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/name> ?name ;
        <http://xmlns.com/foaf/0.1/workplaceHomepage> ?org .

  ?org <http://www.w3.org/2000/01/rdf-schema#label> ?orgLabel .
}
```

Listing 1: Example SPARQL query.

### 2.2.5 The DBLP Knowledge Graph

This thesis utilizes the April 2024 version of the DBLP Computer Science Bibliography as the primary data source for experimentation. While DBLP has historically been maintained as a semi-structured XML collection [93], the DBLP Knowledge Graph (DBLP KG) exposes this dataset as high-quality Linked Data, grounded in a domain-specific ontology [4].

The ontology of the DBLP KG is designed to reflect the specific editorial processes and data structures of the bibliography. It is centered around three primary classes. The class `dblp:Publication` represents scholarly works and includes a hierarchy of subclasses such as `dblp:JournalArticle` and `dblp:ConferenceandWorkshopPaper`. The class `dblp:Creator` represents agents and distinguishes between individual persons and groups. A unique feature of this ontology is `dblp:AmbiguousCreator`. This class explicitly models author names that have not yet been disambiguated into unique identities (e.g., a "J. Smith" who might refer to multiple distinct people). By modeling ambiguity explicitly rather than forcing a potentially incorrect resolution, the graph preserves the provenance and quality of the underlying data [84].

To model the concept of authorship, DBLP employs a reification pattern using the class `dblp:Signature`. In a simple graph, an edge is binary, connecting two nodes (e.g., Publication $\rightarrow$ Person). However, authorship is often an n-ary relationship that includes additional attributes, such as the order of authors in the byline or their specific affiliations for that paper. To capture this, the DBLP KG introduces the Signature as an intermediate node: a Publication is linked to a Signature, and the Signature is linked to a Creator. This structure allows for the attachment of edge-specific metadata to the authorship relation itself.

Finally, the DBLP KG functions as a hub in the scholarly web by linking entities to external identifiers via the `datacite:Identifier` class. This provides dense connections to external systems such as ORCID, Wikidata, and OpenAlex, enabling cross-platform data integration.

### 2.2.6 Knowledge Graph Question Answering (KGQA)

Knowledge Graph Question Answering (KGQA) systems abstract the complexity of formal query languages, enabling users to retrieve information from a graph using natural language. Methodologies in this field are generally categorized into two paradigms: information retrieval (IR) approaches and semantic parsing approaches [90]. IR-based methods typically extract a subgraph relevant to the entities mentioned in the question and rank candidate answers based on graph features or semantic similarity.

This work adopts the Semantic Parsing paradigm. Semantic parsing models KGQA as a translation task, where the objective is to map a natural language input $N$ to a formal logical form $L$. In this context, a SPARQL query, which, when executed on the graph, returns the correct answer [77]. This process involves several complex sub-tasks, including Entity Linking (mapping textual phrases to unique IRIs), Relation Extraction (identifying the correct predicates), and Composition (structuring the logic with correct operators). Semantic parsing is particularly relevant for questions that require aggregation (e.g., counting papers) or comparative reasoning, as these logic-heavy operations are difficult to resolve through simple graph traversal techniques.

### 2.2.7 Neural Semantic Parsing and Large Language Models

The state-of-the-art in semantic parsing has shifted from rule-based systems to Neural Semantic Parsing, specifically leveraging Large Language Models (LLMs). In this formulation, the problem is treated as a sequence-to-sequence generation task. The model predicts a sequence of SPARQL tokens conditioned on the input question and a textual representation of the graph schema [94].

However, applying LLMs to the Text-to-SPARQL task presents specific challenges, most notably regarding Schema Faithfulness. Knowledge graph schemas, such as DBLP, often contain thousands of predicates and classes. It is computationally infeasible to include the entire ontology definition in the context window of the model. Consequently, effective systems must employ "schema linking" techniques to identify and prune the ontology to a relevant subset before generation [59]. Furthermore, LLMs are prone to hallucination, where they generate plausible-looking but non-existent IRIs. Addressing this requires fine-tuning strategies, such as the Group Relative Policy Optimization (GRPO) utilized in this work, to align the output probability distribution of the model with the strict topological constraints of the underlying Knowledge Graph.

### 2.2.8 Evaluation Metrics

To assess the performance of Text-to-SPARQL systems, the literature employs a variety of metrics that evaluate generated queries at different levels of abstraction. The strictest metric, Exact Match (EM) accuracy, compares the generated SPARQL string literal directly to a gold-standard query. While rigorous, EM is frequently criticized for being overly penalizing, as it fails to account for semantically equivalent queries that differ only in syntactic details, such as variable naming conventions or the ordering of filter clauses [82].

To address the limitations of syntactic comparison, Execution Accuracy (ExAcc) serves as a robust functional metric. It compares the result sets returned by executing both the generated query and the gold query against the target endpoint. If the results are identical—accounting for set ordering and datatypes—the generation is considered correct. ExAcc verifies that the model has captured the correct user intent and logic, regardless of the specific syntax used to express it.

However, binary metrics such as Execution Accuracy do not capture partial correctness. To provide a continuous measure of performance, set similarity metrics such as the $F_\beta$ score are utilized to evaluate the overlap between the retrieved answer set and the expected gold standard set [27]. This approach allows for a granular assessment of answer completeness (recall) and correctness (precision), particularly in cases where the system retrieves a valid subset or a superset of the correct entities. Reflecting these considerations, this study reports both $F_\beta$ scores and execution accuracy to provide a comprehensive evaluation of model performance, as detailed in Chapter 3.5.

## 2.3 Policy Optimization in Reinforcement Learning

In this chapter, the theoretical foundation of Reinforcement Learning (RL) and Policy Optimization is established. The first section conceptually introduces the RL framework and terminology. The second section discusses Policy Gradient Methods, before the third section introduces Trust Region Policy Optimization (TRPO) and its variants, most notably GRPO, which is relevant for later chapters.

### 2.3.1 The Reinforcement Learning Framework

Reinforcement Learning (RL) is a paradigm for sequential decision-making in which an agent interacts with an environment over a sequence of time steps. At each time step, the agent observes the current state of the environment, chooses an action, and

in return receives a reward and a new state from the environment. This forms the agent-environment interaction loop, as illustrated in Figure 7

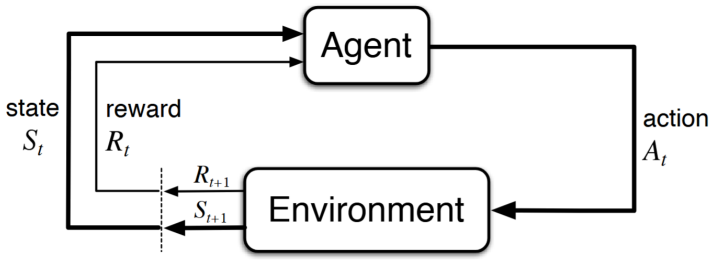


Figure 7: The agent–environment interaction (from Sutton and Barto [140])

The agent's policy $\pi$ can be seen as a function, mapping states to actions, and thus determining which actions the agent takes in a given state. The agent's objective is to learn a policy that maximizes its long-term cumulative reward. The cumulative reward over time is formally defined as return $G$, which can be any function over the rewards. In the simplest summation case, with an episodic task (i.e. finite number of total timesteps $T$), the return can be defined as

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T. \tag{1}$$

The environment's dynamics are typically modeled as a Markov Decision Process (MDP), i.e. the task follows the Markov assumption with the probability of a new state $s'$ and reward $r$ only depending on the previous state $s$ and the chosen action $a$:

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|St = s, At = a\}. \tag{2}$$

To maximize the return from this environment, not only the immediate reward needs to be considered, but the long-term expected return for being in a state or for taking a specific action. This prediction of future rewards is used to assess how good a given policy is in the long-run. This notion is expressed in terms of a value function $v$, representing the expected cumulative reward from state $s$ onward when following

policy $\pi$. Similarly, an action-value function $q$ expresses the expected cumulative reward starting from state $s$, taking action $a$ under the policy $\pi$:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s], \tag{3}$$
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{4}$$

While the value function $v_\pi(s)$ expresses how good it is to be in a state under policy $\pi$, and the action-value function $q_\pi(s, a)$ expresses how good it is to take a specific action in that state, the advantage function measures the relative benefit of choosing action $a$ compared to the policy's average behavior at state $s$. Formally, it is defined as

$$A_\pi(s, a) = q_\pi(s, a) - v_\pi(s). \tag{5}$$

The advantage is therefore a centered quantity that highlights whether an action yields higher or lower expected return than the typical action chosen by the current policy. Positive values indicate actions that outperform the state baseline, while negative values indicate actions that are worse than what the policy would usually do. This formulation is particularly useful for policy gradient methods, since it reduces variance by removing the common baseline term $v_\pi(s)$ while preserving the direction of the gradient

A foundational assumption in RL is the reward hypothesis, which posits that any goal can be expressed as the maximization of some scalar reward signal. In formal terms, "all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (called reward)" [140]. This implies that if we design an appropriate reward function for a task, an RL agent that maximizes that reward should, in principle, achieve the intended goal. This hypothesis underpins the RL approach, in that the design of the reward signal indirectly specifies the desired behavior for the agent.

While RL originated in domains like robotics and games, its framework is general and can be applied to domains like natural language generation (see also Section 2.1.5.4). Considering a LLM in a question-answer setting, the LLM can be viewed as an RL agent interacting with a textual environment. The state may be the current dialogue context or prompt (including the question asked and any conversation history), and

an action is the generation of the next token or word in the answer. The policy $\pi_\theta$ of the LLM (with parameters $\theta$) defines a probability distribution over possible next-token actions given the state (the prompt and generated prefix). An episode might consist of the model generating a complete answer, after which it receives a scalar reward measuring the quality of that answer. For example, the reward could be $+1$ for a perfectly correct and coherent answer, 0 for an incorrect answer, or a more fine-grained score given by a human evaluator or an automatic metric.

Using RL, LLMs can be fine-tuned such that the probability of high-reward outputs corresponding to higher quality answers are increased and the probability of low-reward outputs is decreased. In essence, the language model learns through trial-and-error to refine its text generation policy in order to maximize the expected reward, much like an agent learning to achieve goals in a traditional RL environment.

### 2.3.2 Policy Gradient Methods

The previous section introduced the agent–environment interaction, return $G_t$ and the value functions $v_\pi(s)$ and $q_\pi(s, a)$ that describe how good it is to follow a given policy $\pi$ from a state or state–action pair. In many classical algorithms, such as dynamic programming and temporal-difference control, the policy is derived indirectly from these value estimates. Policy gradient methods, in contrast, directly parametrize the policy as a stochastic mapping $\pi_\theta(a \mid s)$ with parameters $\theta$ and adjust these parameters to maximize the expected return under the induced trajectory distribution.

Formally, one can describe the performance of a parametrized policy by the objective

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_0], \tag{6}$$

where $G_0$ is the return defined above for an episode starting at time $t = 0$. The policy gradient theorem provides a general expression for the gradient of this objective. For a differentiable stochastic policy and an MDP as described in the previous section, it states that

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t \mid S_t)\, q_{\pi_\theta}(S_t, A_t)\right], \tag{7}$$

where $q_{\pi_\theta}$ is the action-value function under $\pi_\theta$ [141]. This result shows that the gradient can be written in terms of the score function $\nabla_\theta \log \pi_\theta(A_t \mid S_t)$ and does not require differentiating through the environment dynamics.

The REINFORCE algorithm, introduced by Williams [156], is the earliest and most fundamental Monte Carlo policy gradient method. In the original connectionist formulation, the update for a weight $w_{ij}$ in a network of stochastic units is

$$\Delta w_{ij} = \alpha \, (r - b_{ij}) \, e_{ij}, \tag{8}$$

where $\alpha$ is the step-size, $r$ is a scalar reinforcement signal, $b_{ij}$ is a baseline associated with parameter $w_{ij}$, and $e_{ij}$ is an eligibility term corresponding to the gradient of the log-probability of the unit's output with respect to $w_{ij}$ [156]. This expression is an instance of the likelihood-ratio (or score-function) estimator applied to the expected reinforcement.

In modern reinforcement learning notation, REINFORCE is usually expressed directly in terms of trajectories generated by $\pi_\theta$. For an episodic task, let $G_t$ denote the return from time step $t$ as defined above. The policy gradient theorem suggests the Monte Carlo gradient estimator

$$\hat{g}(\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t \mid S_t) \, G_t, \tag{9}$$

which leads to the update

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t \mid S_t) \, G_t. \tag{10}$$

This estimator is unbiased but typically has high variance because it relies on full-episode returns. A standard variance reduction technique is to subtract a baseline that depends only on the state and does not change the expected value of the gradient estimator [141, 58]. Using a state-dependent baseline $b(S_t)$ yields the generalized REINFORCE update

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t \mid S_t) \, \big(G_t - b(S_t)\big). \tag{11}$$

If the baseline $b(S_t)$ approximates the value function $v_{\pi_\theta}(S_t)$ introduced in (3), then the term $\big(G_t - b(S_t)\big)$ acts as an empirical estimate of the advantage

$$A_{\pi_\theta}(S_t, A_t) = q_{\pi_\theta}(S_t, A_t) - v_{\pi_\theta}(S_t), \tag{12}$$

as defined in (5). In this view, the update increases the log-probability of actions with positive estimated advantage and decreases it for actions whose expected return

is worse than the state baseline. Baselines of this kind can substantially reduce the variance of policy gradient estimates while leaving their expectation unchanged [58].

In practice, policy gradient methods such as REINFORCE are implemented by repeatedly sampling episodes under the current policy, computing returns $G_t$ and (optionally) baseline or advantage estimates for each time step, and then updating $\theta$ according to the gradient estimator above. Although this approach is conceptually simple and directly optimizes the expected return, the reliance on Monte Carlo returns often leads to slow and unstable learning due to high variance in the gradient estimates. These limitations motivate actor–critic methods, where a learned value function is used to provide lower-variance estimates of $q_{\pi_\theta}$ or $A_{\pi_\theta}$ and thereby yield more stable and sample-efficient policy updates [87].

### 2.3.3 Trust Region Policy Optimization

While policy gradient methods directly optimize the expected return by following stochastic gradient estimates, they can suffer from instability in practice. Large parameter updates may push the policy into regions of parameter space where it performs poorly, and using trajectories collected under an old policy to update a significantly changed new policy introduces bias, since the policy gradient estimator is inherently on-policy. At the same time, high variance in the gradient estimates can cause slow and erratic learning, even when baselines and advantage estimators are used [58].

Trust Region Policy Optimization (TRPO), introduced by Schulman et al. [124], addresses these issues by constraining each policy update to remain within a small trust region of the previous policy. The method builds on the conservative policy iteration framework of Kakade and Langford [81], which provides a bound relating the performance difference between two policies to a surrogate objective and a divergence term. In particular, under certain assumptions one can bound the performance of a new policy $\pi_\theta$ by

$$J(\pi_\theta) \;\geq\; L_{\pi_{\theta_{\text{old}}}}(\pi_\theta) - C \max_s \mathrm{KL}(\pi_{\theta_{\text{old}}}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s)) \,, \tag{13}$$

where $L_{\pi_{\theta_{\text{old}}}}$ is a surrogate objective estimated using data from the old policy and $C$ is a constant depending on the discount factor and reward range [81, 124]. Maximizing this surrogate while keeping the divergence small yields a sequence of policies that are guaranteed to improve monotonically in the idealized setting.

In practice, TRPO replaces the theoretical max over states by an empirical average

and optimizes the following constrained problem:

$$\max_{\theta} \quad \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \tag{14}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[ \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)] \right] \leq \delta, \tag{15}$$

where $\hat{A}_t$ is an empirical estimate of the advantage $A_{\pi_{\theta_{\text{old}}}}(s_t, a_t)$ defined in (5), and the expectation is taken over states and actions sampled from the old policy $\pi_{\theta_{\text{old}}}$ [124]. The ratio between new and old policy probabilities plays the role of an importance sampling factor, turning on-policy expectations under the old policy into a surrogate for the new policy performance.

Directly solving the constrained problem in (14)–(15) is expensive, so TRPO uses a sequence of approximations. The objective is linearized around $\theta_{\text{old}}$, and the KL constraint is approximated quadratically using the Fisher information matrix of the policy, which is the Hessian of the KL divergence at $\theta_{\text{old}}$ [124]. The resulting quadratic program is solved approximately with conjugate gradient to obtain a search direction, followed by a backtracking line search that selects a step size satisfying the KL constraint in (15). This procedure yields a natural policy gradient style update that respects a trust region in policy space, and in practice leads to robust performance improvements for large nonlinear policies such as neural networks.

#### 2.3.3.1 Generalized Advantage Estimation

The effectiveness of methods such as TRPO depends on the quality of the advantage estimator $\hat{A}_t$. Simple Monte Carlo estimates based on returns $G_t$ have low bias but high variance, whereas temporal-difference (TD) estimates have higher bias but lower variance. Generalized Advantage Estimation (GAE), introduced by Schulman et al. [125], provides a principled way to interpolate between these extremes. It defines a family of advantage estimators

$$\delta_t = R_t + \gamma V_\phi(S_{t+1}) - V_\phi(S_t), \tag{16}$$

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l}, \tag{17}$$

where $V_\phi$ is a learned approximation of the value function and $\lambda \in [0, 1]$ is a trace parameter analogous to that in TD($\lambda$). Setting $\lambda = 1$ recovers a Monte Carlo style estimator, while smaller values of $\lambda$ place more weight on short-horizon TD errors, reducing variance at the cost of additional bias. Empirically, GAE has been shown

to substantially improve sample efficiency and stability for policy gradient methods in high-dimensional continuous control tasks, and it is commonly used to compute $\hat{A}_t$ in both TRPO and PPO [124, 125].

#### 2.3.3.2 Proximal Policy Optimization

Despite its favorable theoretical properties, TRPO has a relatively complex implementation and requires second-order information via Fisher-vector products and line search at every update. Proximal Policy Optimization (PPO) was proposed by Schulman et al. [126] as a simpler first-order alternative that retains many of the practical benefits of TRPO. PPO is also based on the surrogate objective with an importance sampling ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \tag{18}$$

but instead of enforcing an explicit KL constraint, it incorporates a soft trust region directly into the objective. In its clipped variant, the surrogate objective is defined as

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t\big)\big], \tag{19}$$

where $\hat{A}_t$ is typically computed using GAE and $\varepsilon$ is a small hyperparameter [126]. For positive advantages, clipping caps the benefit of increasing $r_t(\theta)$ above $1+\varepsilon$, and for negative advantages it caps the benefit of decreasing $r_t(\theta)$ below $1-\varepsilon$. Taking the minimum between the unclipped and clipped terms yields a pessimistic surrogate, which acts as a lower bound on the improvement predicted by the unconstrained surrogate. This construction removes the incentive to push $r_t(\theta)$ far outside the interval $[1-\varepsilon, 1+\varepsilon]$, thereby discouraging overly large policy updates.

In practical implementations, PPO optimizes a composite loss that combines the clipped policy surrogate, a squared-error loss for the value function, and an entropy bonus encouraging exploration [126]. Unlike TRPO, PPO performs multiple epochs of minibatch stochastic gradient updates on the same batch of collected trajectories, which improves sample efficiency while the clipping mechanism keeps the policy updates stable. The original paper also considers an alternative formulation with an explicit KL penalty term in the objective, where a coefficient is tuned to keep the KL divergence near a target value, but empirically the clipped surrogate was found to be more robust and less sensitive to hyperparameter choices [126].

Because of its conceptual simplicity, ease of implementation and robust empirical performance across a wide range of continuous control and discrete-action tasks,

PPO has become a standard choice for large-scale deep RL. In particular, PPO-style objectives with additional KL regularization are used to fine-tune LLMs in RLHF pipelines (see Section 2.1.5.4).

#### 2.3.3.3 Group Relative Policy Optimization

While Proximal Policy Optimization (PPO) has become a standard approach for fine-tuning LLMs with RL, its implementation typically maintains a separate value function network alongside the policy model. This dual-network architecture increases memory usage and computational complexity, especially for large language models, since the value network is often a full copy of the policy backbone. Group Relative Policy Optimization (GRPO), introduced by Shao et al. [131] and subsequently adopted in models such as DeepSeek-R1 [60], addresses this limitation by retaining a PPO-style clipped surrogate and KL regularization while eliminating the need for an explicit value function.

Formally, following Shao et al. [131], for each question $q$ a group of $G$ outputs

$$\{o_1, o_2, \ldots, o_G\}$$

is sampled from the current (old) policy $\pi_{\theta_{\text{old}}}$. A reward model $r_\phi$ then scores each output, yielding rewards $r = \{r_1, r_2, \ldots, r_G\}$. Instead of estimating advantages via a learned value function and GAE, GRPO constructs a group-relative advantage by normalizing rewards within the group. At the sequence level, the advantage for output $o_i$ is defined as

$$A_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G)}. \tag{20}$$

For outcome supervision, this scalar advantage is then broadcast to all tokens in the completion,

$$\hat{A}_{i,t} = A_i \quad \text{for all } t \in \{1, \ldots, |o_i|\}, \tag{21}$$

so that every token in $o_i$ shares the same learning signal. For process supervision, where a process reward model assigns scores to intermediate reasoning steps, the step-wise rewards are first normalized within the group and then accumulated over future steps; the advantage for token position $t$ is defined as the sum of normalized rewards from that step onward [131].

Adapting the PPO surrogate to this setting and writing the token-level form commonly used for language models, the GRPO objective can be expressed as

$$L^{\mathrm{GRPO}}(\theta) = \hat{\mathbb{E}}_{q,\{o_i\}_{i=1}^{G}} \left[ \frac{1}{G} \sum_{i=1}^{G} \left( \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \left( r_{i,t}(\theta)\, \hat{A}_{i,t},\ \mathrm{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right) \hat{A}_{i,t} \right) \right. \right. \\ \left. \left. -\beta\, D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid s_i) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid s_i)\right) \right) \right], \tag{22}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid s_i)}{\pi_{\theta_{\mathrm{old}}}(o_{i,t} \mid s_i)} \tag{23}$$

is the per-token importance sampling ratio as in PPO, $\varepsilon$ is the clipping parameter, $\beta$ controls the strength of KL regularization, and $\pi_{\mathrm{ref}}$ is a fixed reference policy, typically the supervised fine-tuned checkpoint before the RL process. This objective is directly analogous to the PPO clipped surrogate, but it uses the group-relative advantage (20) instead of a GAE-based estimate from a learned value function, and it includes the KL penalty as an explicit term in the loss rather than folding it into the reward [131]. Figure 8 illustrates the conceptual difference between GRPO and PPO.

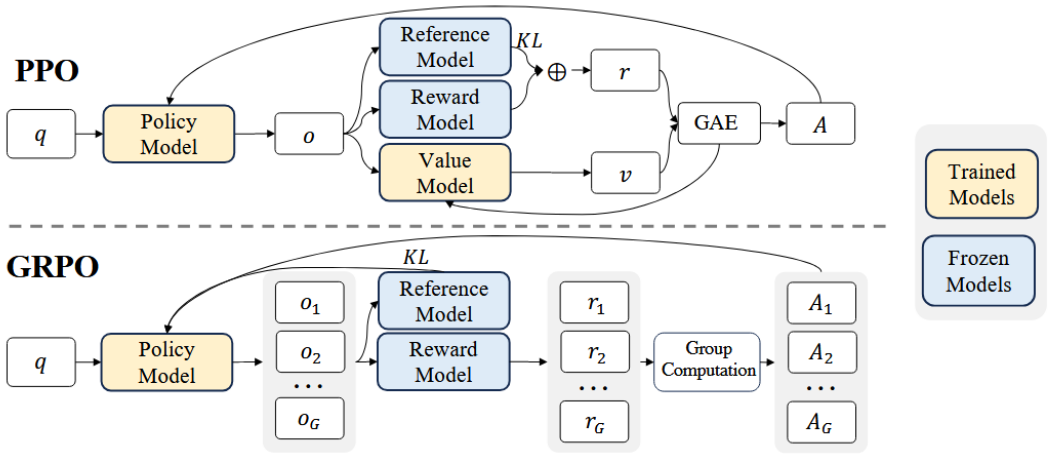


Figure 8: GRPO vs. PPO (from Shao et al. [131]). GRPO estimates the expected advantage based on a sample (group) of sequences for each training example, therefore eliminating the need for a separate value model.

This formulation offers several practical advantages over PPO-style RLHF for large language models. First, by eliminating the value function network, GRPO avoids storing and training an additional model of comparable size to the policy, which substantially reduces memory and compute overhead relative to PPO-based pipelines [131]. Second, the group-relative normalization focuses the update on the *relative* ranking of responses within each prompt and makes the learning signal less sensitive to absolute reward scale differences across prompts, which aligns well with preference-based and verifiable reward models [131, 60]. At the same time, the standard-deviation normalization introduces a known bias towards prompts with low reward variance, motivating variants such as Dr. GRPO and difficulty-aware GRPO that modify or remove this term [96].

GRPO naturally accommodates both outcome supervision, where only final answers receive rewards, and process supervision, where intermediate reasoning steps are rewarded. In DeepSeekMath, process-supervised GRPO yields stronger performance than outcome-supervised GRPO on challenging mathematical reasoning benchmarks such as GSM8K and MATH [131]. Combined with domain-specific pre-training, this RL setup enables DeepSeekMath-7B to achieve then state-of-the-art performance among open-weight models on MATH, approaching proprietary systems like GPT-4 and Gemini-Ultra [131]. These results, together with subsequent GRPO-based alignment of models such as DeepSeek-R1 [60], make GRPO a widely adopted choice for task-specific RL fine-tuning of large language models.

# 3 Methodology

## 3.1 Dataset and Preprocessing

### 3.1.1 DBLP–QuAD Dataset

All experiments in this thesis are conducted on DBLP-QuAD, a knowledge-graph question answering (KGQA) dataset built on top of the DBLP scholarly knowledge graph (see Section 2.2.5). DBLP-QuAD contains 10 000 question–SPARQL pairs that can be executed against the DBLP KG to retrieve the gold answers, and is, to date, the largest publicly available scholarly KGQA dataset of this kind [22].

DBLP-QuAD was originally released as part of the Scholarly QALD challenge at ISWC 2023 and is distributed via Hugging Face [23].

#### 3.1.1.1 Dataset Generation

DBLP–QuAD is generated using a semi-automatic, template based framework inspired by the OVERNIGHT approach to semantic parsing dataset construction [151, 22]. The authors first design a set of 98 template tuples by inspecting the DBLP ontology. Each tuple $t = (s_t, Q_t, E_t, P_t)$ consists of a SPARQL query template $s_t$, a set of semantically equivalent natural language question templates $Q_t$, a set of entity placeholders $E_t$, and the set of predicates $P_t$ used in the query template [22].

The template inventory covers the two main entity types of the DBLP knowledge graph (creator and publication), the bibliographic type and eleven predicates of the schema. Templates are grouped along two axes: they are either creator focused or publication focused, and they are assigned to one of ten query types (Single Fact, Multiple Facts, Boolean, Negation, Double Negation, Double Intent, Union, Count, Superlative or Comparative, and Disambiguation). For each template tuple, between four and seven paraphrased question templates are provided, including both standard wh questions and instruction style paraphrases, which yields variation in surface form.

To instantiate these templates, the framework samples local two hop subgraphs from the DBLP RDF snapshot. Starting from a randomly drawn publication node, the procedure collects adjacent creator nodes and literal metadata, then expands the creator nodes to obtain their associated literals, which yields a small neighborhood subgraph of the DBLP knowledge graph. For each desired data instance, a template

tuple is sampled in a stratified manner so that entity type and query type are approximately balanced. The placeholders in the SPARQL query template and in the natural language question templates are then filled with URIs and surface forms from the sampled subgraph to obtain concrete SPARQL queries and corresponding natural language questions. During this step, simple rule-based data augmentation is applied to literals in order to increase linguistic diversity and make entity linking more challenging. Names are permuted or shortened to initials, venues are alternated between short forms and full names, temporal durations are expressed either numerically or in words, affiliations are shortened from full postal addresses to institution names, and titles used in disambiguation questions are reduced to noun phrase keywords extracted with SpaCy [22]. For each resulting data instance, the instantiated SPARQL query is executed against a Virtuoso endpoint hosting the DBLP RDF dump, and only those question query pairs that return at least one answer are retained. Each example in DBLP–QuAD thus consists of an executable SPARQL query, a main natural language question and a paraphrase, together with the instantiated entities and predicates.

#### 3.1.1.2 Dataset Structure and Statistics

Each DBLP-QuAD instance consists of a structured record that combines natural language and symbolic fields. In addition to the main natural language question and its paraphrase, each instance stores the corresponding SPARQL query, the query type, an identifier of the underlying template, a list of entities participating in the query, a list of relations, a Boolean flag that marks temporal questions, and a Boolean flag indicating whether the instance stems from templates that are held out from the training set (Banerjee et al., 2023; Abi Akl et al., 2023). Listing 2 shows one sample from the dataset in a JSON format.


```
{
    "id": "Q0509",
    "query_type": "SINGLE_FACT",
    "question": {
        "string": "Name the authors of the paper 'Cultivating Interaction
            Ubiquity at Work'."
    },
    "paraphrased_question": {
        "string": "'Cultivating Interaction Ubiquity at Work' was authored
            by which authors?"
    },
    "query": {
```

```
        "sparql": "SELECT DISTINCT ?answer WHERE { <https://dblp.org/rec/
            journals/tis/Sorensen10> <https://dblp.org/rdf/schema#authoredBy>
             ?answer }"
    },
    "template_id": "TP01",
    "entities": [
        "<https://dblp.org/rec/journals/tis/Sorensen10>"
    ],
    "relations": [
        "<https://dblp.org/rdf/schema#authoredBy>"
    ],
    "temporal": false,
    "held_out": false
}
```

Listing 2: Example JSON sample

In total, DBLP-QuAD comprises 10 000 unique question–query pairs that are split into training, validation, and test sets with a fixed ratio of 7:1:2. For each of the ten query types, the dataset contains exactly 1 000 instances, equally divided between creator- and publication-focused questions. The instances cover 13 348 distinct creator and publication entities and 11 predicates from the DBLP KG. Overall, 2,350 questions are marked as temporal, which require reasoning over publication years or other time related information.

The authors report that natural language questions in DBLP–QuAD have an average length of 17.32 tokens and 114.1 characters, while the corresponding SPARQL queries have an average vocabulary size of 12.65 tokens and 249.48 characters [22]. The paraphrased question templates exhibit moderate lexical diversity, with average unigram and bigram Jaccard similarities of 0.62 and 0.47 respectively, and an average Levenshtein edit distance of approximately 33 characters between paraphrases [22], which indicates that paraphrases differ in more than minor edits.

To assess generalization, the dataset includes a subset of question–query pairs generated from template tuples that are withheld from the training set. Roughly 19 percent of the instances in the validation and test sets are created from these withheld templates, which leads to non i.i.d. distributions and enables the evaluation of compositional and zero shot generalization beyond the patterns seen during training [22].

#### 3.1.1.3 Dataset Limitations

While DBLP-QuAD is a comparatively large and carefully constructed scholarly KGQA dataset, Banerjee et al. [22] discuss several limitations that are relevant for interpreting experimental results. First, although all templates are human-written, the questions themselves are generated programmatically from the KG and only two authors were involved in authoring and validating the templates. As a result, the distribution of questions may not fully reflect real user information needs. Second, since around 80 % of the syntactic template structures in validation and test also occur in the training set, there is some degree of test leakage at the level of surface query patterns. Third, the dataset inherits the shortcomings of the DBLP KG, for example missing literals for some predicates and paper titles that do not correspond to how papers are referred to in practice. This leads to challenging entity linking cases, such as well known papers that are typically referred to by acronyms like "GPT-3" or "BERT" instead of their full titles. Since entity and relation linking is assumed to be perfect, this limitation does not affect the scope of this work.

### 3.1.2 Dataset Preprocessing

For the experiments reported in this thesis, the publicly released DBLP-QuAD dataset is retrieved from the Huggingface repository `awalesushil/DBLP-QuAD` and is used without modifying its train-validation-test split. The dataset then preprocessed by means of a multi-stage pipeline that standardizes queries and answer sets, enriches entities and relations with human readable metadata, and formats the final input in a conversational style suitable for LLMs.

Since the original DBLP-QuAD dataset does not contain explicit gold answer sets, all gold queries are executed against a Virtuoso SPARQL endpoint populated with the DBLP knowledge graph (for more details on the SPARQL endpoint see Section 3.4.2). For each query, the complete result set is retrieved using the JSON serialization of SPARQL results and stored alongside the original sample. This step ensures that every question–query pair is associated with a materialized gold answer set as returned by the underlying DBLP knowledge graph.

The raw JSON responses returned by the SPARQL endpoint are not directly suitable for downstream training and evaluation. Therefore, the bindings section of each SPARQL result is transformed into a compact and uniform internal representation. Concretely, each result is converted into a list of tuples, where each tuple corresponds to a single answer row and contains the values of the projected variables in a fixed order. This abstraction removes SPARQL specific details of the JSON format and

provides a simple, model agnostic structure that can be used both for automatic evaluation and for manual inspection of answer sets.

To reduce superficial variability between logically equivalent queries, a query normalization procedure is applied to all gold SPARQL queries. This procedure removes unnecessary aliases and redundant projections as far as possible without changing query semantics, and variable names are canonicalized by replacing all original variable identifiers with a standardized sequence v1, v2, and so on according to their order of appearance. As a result, queries that differ only in the choice of variable names or trivial aliasing are mapped to the same normalized form. This canonicalization simplifies later analysis of query patterns and reduces spurious differences that would otherwise affect tokenization and query comparison. Additionally, the query is stripped from redundant whitespaces and converted to lower case. A simple example of a normalized query can be found in the field ‘query_normalized‘ in Listing 3.

In addition to normalizing queries and answer sets, the dataset is enriched with human-readable information about all knowledge graph entities that appear in the questions, queries, or answers. For this purpose, a dump of the DBLP knowledge graph is downloaded and used as an offline index to resolve entity identifiers. For each entity, a human-readable label is extracted, for example the name of an author or the title of a venue, and attached to the corresponding URI. As an illustration, the URI `<https://dblp.org/pid/95/2265>` is augmented with the label “Robert Schober”, which makes it immediately clear that this resource refers to the respective researcher. The same procedure is applied to all other types of entities in the dataset, such as publications, venues, and conferences. The enriched information is stored in an additional field ‘entities_detailed‘ for each sample, which contains, for every referenced entity, both the original URI and the resolved label. This improves interpretability and allows models to access human-readable entity names without dereferencing URIs during training or inference.

A similar enrichment step is performed for relations. The RDF schema defining the DBLP vocabulary is downloaded and used to extract descriptive metadata for every predicate occurring in the dataset. In particular, for each relation, the schema provides a label as well as domain and range information. For example, the relation `https://dblp.org/rdf/schema#wikidata` is described as linking DBLP entities to corresponding Wikidata identifiers, with an entity as domain and any URI as range. By attaching these labels and type constraints, relations are no longer represented solely as opaque URIs but are accompanied by short textual descriptions and their expected argument types. The resulting information is stored in a ‘relations_detailed‘ field that is added to each sample and lists, for every relation used in the

query, its URI, label, domain, and range.

Finally, a prompt field is constructed for each sample that encodes the full model input in a conversational format aligned with contemporary chat based LLMs. Each prompt consists of a system message that specifies the general behavior and task of the model, and a user message that is derived from the original question in the dataset. As described in Section 2.1.6, the input prompt design can have a significant effect on the output quality of an LLM, hence the prompts for this task were carefully and iteratively designed, to provide guidance both on syntactical and semantical style of the expected output, including the CoT pattern. The prompts can be found in Appendix A.1.1 and A.1.2 respectively. By storing this structured prompt alongside the original fields and the enriched metadata, the preprocessed dataset becomes directly usable for training and evaluation without additional on the fly formatting.

Listing 3 shows the sample from Listing 2 after the preprocessing:


```
{
    "id": ["Q0509"],
    "question": ["Name the authors of the paper "Cultivating Interaction
       Ubiquity at Work"."],
    "entities": [("<https://dblp.org/rec/journals/tis/Sorensen10>",)],
    "relations": [("<https://dblp.org/rdf/schema#authoredBy>",)],
    "query": ["SELECT DISTINCT ?answer WHERE { <https://dblp.org/rec/
       journals/tis/Sorensen10> <https://dblp.org/rdf/schema#authoredBy> ?
       answer }"],
    "answer": [[("https://dblp.org/pid/79/611",)]],
    "query_type": ["SINGLE_FACT"],
    "entities_detailed": [("<https://dblp.org/rec/journals/tis/Sorensen10>
       (['Carsten S\\u00F8rensen: Cultivating Interaction Ubiquity at Work.
        (2010)'])",)],
    "relations_detailed": [("#Publication <https://dblp.org/rdf/schema#
       authoredBy> #Creator (The publication is authored by the creator.)
       ",)],
    "prompt": [{"role": ["system"], "content": ["You are a SPARQL expert.\n
       Your task is to generate a syntactically and semantically correct
       ..."]}],
    "query_normalized": ["select distinct ?v1 where { <https://dblp.org/rec/
       journals/tis/sorensen10> <https://dblp.org/rdf/schema#authoredby> ?
       v1 }"]
}
```


Listing 3: Preprocessed Sample

Taken together, these preprocessing steps yield a version of DBLP-QuAD in which gold queries and answers are materialized and normalized, entities and relations are enriched with human-readable labels and schema information, and each sample is equipped with a ready to use conversational prompt. This improves the interpretability, consistency, and practical usability of the dataset as a benchmark for applying LLMs to KGQA over DBLP.

## 3.2 Model Selection

For the experiments in this work, the Qwen3-1.7B model is adopted as the base model because the Qwen family shows strong reasoning performance across a variety of tasks (see Section 2.1.7.2) and, more importantly for this thesis, systematically outperforms other open-weight models such as Llama when fine-tuned with PPO-style reinforcement learning on mathematical reasoning tasks.

Gandhi et al. [53] systematically compare Qwen2.5-3B and Llama-3.2-3B under an identical TinyZero PPO setup on the Countdown game and find that Qwen achieves large gains from RL while Llama improves only modestly. They attribute this discrepancy to richer pre-existing "cognitive behaviors" such as verification and backtracking in Qwen that RL can amplify, and show that injecting such behaviors into Llama restores its ability to benefit from RL [53].

Shao et al. [130] study reinforcement learning with verifiable rewards (RLVR) using GRPO on Qwen2.5-Math models and report absolute improvements on MATH-500 of roughly 20 to 30 points, even when using spurious or random rewards, whereas comparable spurious reward signals fail to produce consistent gains and can even hurt performance for Llama3.1-8B-Instruct and OLMo2-7B [130].

Complementary evidence from OctoThinker shows that under an R1-Zero style RL configuration, Qwen2.5-3B exhibits stable length growth and substantial accuracy gains on benchmarks such as GSM8K and MATH-500, while Llama-3.2-3B under the same setup often shows minimal gains and unstable response lengths. The authors explicitly note that current open RL results are concentrated on Qwen and design OctoThinker to close this gap for Llama [152].

Although these studies focus on math reasoning, they indicate that small Qwen models provide a RL-friendly starting point that can realise comparatively large gains from PPO style algorithms such as GRPO. Within this landscape, Qwen3-1.7B offers a favourable trade-off between capacity and training cost, is part of a multi-stage post-training pipeline that includes a dedicated GRPO-based "Reasoning RL"

stage and subsequent distillation into smaller models, and is released as an open-weight model with mature tooling, which together make it a suitable base model for the experiments in this work.

From a methodological perspective, this thesis restricts itself to a single base model. This choice reflects practical constraints, as GRPO training on DBLP-QuAD is computationally and memory intensive, and a broader comparison across multiple model sizes or architectures would have required larger GPU resources and training time. The available budget is therefore concentrated on a more thorough exploration of reward configurations and training runs for one representative small model. A more systematic cross-model analysis is left to future work and is discussed in Section 5.

## 3.3 Training Logic

### 3.3.1 Overview and Training Objective

The SPARQL generation for DBLP-QuAD is formulated as a single-step sequential decision problem within the RL framework, as covered in Section 2.3. The policy $\pi_\theta$ is instantiated by the Qwen3-1.7B decoder-only LLM, as described in the previous section. This policy defines a probability distribution over token sequences that encode candidate SPARQL queries.

Each training instance involves a single episode, initiated by a natural language question together with perfectly linked entities and relations from the DBLP KG. The model's task is to generate a syntactically and semantically correct SPARQL query using only the provided KG elements. Since entity and relation linking is assumed to be perfect, the model operates solely on pre-disambiguated KG elements and is not required to resolve mentions to URIs.

The overall training objective is to maximize the expected reward of the generated queries under the policy $\pi_\theta$. This maximization is achieved using the Group Relative Policy Optimization (GRPO) algorithm, as detailed in Chapter 2.3.3.3.

In the GRPO setup, the policy generates $G = 4$ candidate completions, or rollouts, for each input question within a batch. Each completion receives a scalar terminal reward based on the generated SPARQL query. The rewards within each group are subsequently normalized by subtracting the group mean and dividing by the group standard deviation. This standardization process yields a centered and scaled signal that serves as an implicit advantage estimate for each rollout.

As described in Chapter 2.3.3.3, the policy update employs a PPO-style clipped

surrogate objective. To maintain training stability and prevent divergence from the sampling policy, the log-probability ratio between the new and old policy is clipped within a symmetric interval defined by the clipping parameter. Throughout the experiments in this thesis, the clipping is set to $\epsilon = 0.2$ . Furthermore, a KL divergence penalty is applied to regularize the updated policy against a fixed reference policy, which is the base Qwen3-1.7B model prior to GRPO fine-tuning. The KL penalty coefficient is set to $\beta = 0.04$. Exploration is primarily controlled through the decoding parameters and the KL constraint, and no explicit entropy bonus is utilized.

### 3.3.2 Policy Input and Output Generation

The policy input is structured as a two-part prompt, consisting of a system message and a user message. The full prompt template is provided in Appendix A.1.

The system message provides comprehensive instructions to the model:

1. Generation of a syntactically and semantically correct SPARQL query.
2. Usage of full URIs without prefixes.
3. Default usage of 'SELECT DISTINCT' for information retrieval queries and 'ASK' for yes/no questions.
4. Required step-by-step internal reasoning enclosed within '<think> ... </think>' tags.
5. Output format constrained to only the final SPARQL query, without additional explanation or formatting outside of the thought block.

The user message contains the natural language question and a complete list of relevant entities and relations. Each entity is presented with its URI and human-readable label. Each relation includes its URI, label, and schema-level context such as comment, domain, and range, providing the model with lightweight ontological grounding, as described in Section 3.1.2.

During both training rollouts and final inference, completions are sampled from the policy using the specific decoding settings as recommended the Qwen3 technical report [163] (temperature 0.6, top-p 0.95, top-k 20, min-p 0). The same configuration is maintained across both stages to ensure the learned policy is evaluated consistently. Details on the decoding parameters, including the repetition penalty (set to 1.0), are relegated to Appendix A.2.1.

### 3.3.3 Reward Function Design

The final scalar reward is a weighted linear combination of six individual components, designed to evaluate answer correctness, structural adherence, format, and similarity to the gold query. These components are scalar-valued, typically ranging in $[-0.5, 1]$ or $[0, 1]$. The resulting aggregated reward is subsequently normalized per group for use in the GRPO loss function.

#### 3.3.3.1 Execution-Feedback Reward ($R_{\text{exec}}$)

This component directly assesses answer-level correctness by executing the generated query ($q_{\text{gen}}$) against the DBLP SPARQL endpoint and comparing the resulting answer set ($A_{\text{gen}}$) to the gold answer set ($A_{\text{gold}}$).

Before execution, a 'LIMIT 3000' clause is appended to 'SELECT' queries to mitigate overly large result sets and potential timeouts. While it is acknowledged, that this could affect the resulting F1 score calculation (also see Section 3.5), the number of cases where the cardinality of the reference result set is in the same order of magnitude as the limit, is very low.

If the limited query fails, the original query is executed as a fallback. Execution failure, due to syntax errors or unsupported constructs, results in a fixed penalty of $-0.5$. If execution succeeds, the reward is calculated as the $F_1$ score between the normalized answer sets:

$$R_{\text{exec}} = \mathrm{F}_1(A_{\text{gen}}, A_{\text{gold}}),$$

where $F_1$ is computed symmetrically ($\beta = 1$) over the returned tuples. If both sets are empty, the reward is 0. This component ensures that the primary optimization target is the functional correctness of the generated query, with a moderate penalty for non-executable outputs.

#### 3.3.3.2 Structural Coverage Reward ($R_{\text{struct}}$)

This reward encourages the grounding of the generated query in the provided context by checking for the inclusion of the required entities and relations present in the input prompt. It is defined as:

$$R_{\text{struct}} = 0.5 \cdot I_{\text{relations}} + 0.5 \cdot I_{\text{entities}},$$

where $I_{\text{relations}}$ is an indicator function (value 1 or 0) for the presence of all required relation URIs in $q_{\text{gen}}$, and $I_{\text{entities}}$ is the analogous indicator for all required entity

URIs. This component promotes queries that utilize the relevant KG context, rather than ignoring parts of the input or inventing new URIs.

#### 3.3.3.3 Format and Well-formed-ness Reward ($R_{\text{format}}$)

The format reward ensures that the completion contains a usable query after the internal thought process. If the completion includes the required closing ‘</think>‘ tag, the suffix is extracted and checked for non-emptiness, yielding $R_{\text{format}} = 1$ if successful and 0 otherwise. If no thought tags are present, any non-empty completion is accepted with a reward of 1. This mechanism acts as a soft constraint on the output format before syntax checking by $R_{\text{exec}}$.

#### 3.3.3.4 Length Reward ($R_{\text{len}}$)

This component mildly penalizes completions approaching the maximum allowed length (1024 tokens) to discourage truncation risk. The reward decreases linearly from 1 as the token length $x$ exceeds a target length $n = 768$:

$$R_{\text{len}} = \text{clip}\left(1 - \frac{x - n}{a - n}, 0, 1\right),$$

where $a = 1024$ is the maximum length.

#### 3.3.3.5 Query Similarity Reward ($R_{\text{sim}}$)

This reward provides token-level supervision by comparing the generated query to the gold SPARQL query ($q_{\text{gold}}$). After normalization and tokenization (as described in Section 3.1.2), sentence-level BLEU is computed:

$$R_{\text{sim}} = \text{BLEU}\big(\text{tokens}(q_{\text{gen}}), \text{tokens}(q_{\text{gold}})\big).$$

This component, available only when gold queries are used during training, encourages the policy to mimic the structure and style of the ground truth.

#### 3.3.3.6 Query Length Ratio Reward ($R_{\text{len-ratio}}$)

To regularize the overall complexity of the generated query, this reward compares the token lengths of $q_{\text{gen}}$ and $q_{\text{gold}}$ using a symmetric log-distance:

$$r = \frac{|q_{\text{gen}}|}{|q_{\text{gold}}|}, \qquad R_{\text{len-ratio}} = \exp\big(-\alpha|\log r|\big),$$

with $\alpha = 2$. The reward peaks at 1 when the lengths are equal and decays exponentially as the ratio $r$ deviates from 1, thus favoring queries whose overall complexity is comparable to that of the gold query, without enforcing exact equality.

Like the similarity reward, the length ratio reward depends on the gold query and is only available in settings where gold SPARQL is provided during training.

#### 3.3.3.7 Reward Aggregation and Weights

For each generated completion, the six components are linearly aggregated using weights that reflect their relative importance:

$$R = 3 \cdot R_{\text{exec}} + 2 \cdot R_{\text{sim}} + 1 \cdot R_{\text{struct}} + 0.5 \cdot R_{\text{format}} + 1 \cdot R_{\text{len}} + 1 \cdot R_{\text{len-ratio}}.$$

The highest weight is assigned to execution-based correctness ($R_{\text{exec}}$), as this is the primary goal. The similarity reward ($R_{\text{sim}}$) is also strongly weighted when gold queries are available, while the remaining components function as lower-weighted shaping terms designed to stabilize learning and enforce output constraints. It was observed that these lower-weighted functions tend to saturate quickly.

The aggregated scalar rewards are normalized per group (subtract mean, divide by standard deviation) before incorporation into the GRPO loss function, to operate on centered and scale-invariant signals while preserving the relative ranking within each group.

Ablation studies concerning the gold-query dependent rewards are presented in Section 4.4.

### 3.3.4 Post-Processing and Error Handling

A robust post-processing mechanism is implemented to reliably extract the final SPARQL query from the raw completion, accommodating minor deviations from the output format specified in the system prompt.

The post-processing procedure locates the last occurrence of the closing '</think>' tag (case-insensitive) and discards all preceding content, isolating the query candidate from the internal Chain-of-Thought. The remaining suffix is then searched for fenced code blocks, potentially annotated with 'sparql'. If code blocks are present, the content of the last block is extracted. If no fenced blocks are found, the entire suffix after the '</think>' tag is used as the candidate query. This logic ensures the recovery of correct queries even when the model inappropriately wraps the output in code fences.

Error handling, including the treatment of invalid or non-executable queries, is primarily managed within the $R_{\text{exec}}$ reward function, which assigns a negative fixed penalty for hard failures, mapping such cases to stable numerical signals for optimization.

### 3.3.5 Optimization and Training Schedule

GRPO training is conducted in a full-parameter fine-tuning setting, adapter-based methods such as LoRA or DoRA are not employed during this RL phase. The optimization is performed using the AdamW optimizer with a learning rate of $1 \times 10^{-6}$, with linear learning rate decay. Standard decay rates ($\beta_1 = 0.9$, $\beta_2 = 0.999$) are used, and weight decay is disabled.

Concrete batch sizes, gradient accumulation steps and the GPU configuration for the main runs are described in the experimental setup (Section 4.1)

### 3.3.6 Baselines and Training Variants

The performance of the main GRPO configuration is benchmarked against two baselines. The first is the unmodified Qwen3-1.7B base model, evaluated directly without any task-specific fine-tuning. The second is a Supervised Fine-Tuning (SFT) baseline, where Qwen3-1.7B is trained using DoRA adapters on the same DBLP-QuAD data using cross-entropy loss and teacher forcing. For a full list of the hyperparameters used for the DoRA training, refer to Appendix A.2.3. Due to resource constraints, DoRA was chosen as PEFT method instead of full supervised fine tuning, as it was shown to provide a strong supervised reference point, while being less compute and memory intensive during training (see Section 2.1.5.3).

Additionally, ablation experiments are performed to analyze the contribution of specific reward functions. One emphasis is on omitting the gold-query dependent rewards ($R_{\text{sim}}$ and $R_{\text{len-ratio}}$) to simulate a more realistic scenario where only gold answers are available. These variants and their results are discussed in more detail in Chapter 4.4.

## 3.4 Implementation Details

This section describes the software and runtime environment used for all GRPO experiments in this thesis. The goal is to enable reproducibility and to clarify the practical constraints under which the models were trained.

### 3.4.1 Software stack and runtime

Training relies on the HuggingFace TRL implementation of Group-Relative Policy Optimization. Rollout generation, loss computation, gradient updates, and parameter synchronization are handled by the HF GRPOTrainer. The language model is loaded locally using the Transformers library, and no external model serving framework is used during training.

Training is performed using the default data-parallel strategy provided by the HuggingFace Trainer, and gradients are synchronized across the devices through the 'torch.distributed' backend, without model parallelism or sharded parameter strategies. For the specific hardware configuration and computing environment in the experiments, see the experimental setup in Section 4.1.

All evaluation and rollout generation operations use the same locally loaded model, and TensorBoard is used as the logging backend to monitor training progress, recording the total reward, individual reward components, execution failure rates, and the empirical KL divergence to the reference policy across the training run.

Model checkpoints are written every 20 optimization steps and stored together with the corresponding optimizer states. No early stopping mechanism is used and all evaluations reported in the Experiments chapter use the final checkpoint after one full epoch over the test set.

Random seed handling is not explicitly controlled, and given the stochasticity of GRPO sampling, decoding, and distributed training, small variations between runs are expected, although the overall behavior is stable under the chosen hyperparameters.

For the exact versions of PyTorch, Transformers, TRL, CUDA, and associated dependencies, refer to the requirements listed in the Github repository [2].

### 3.4.2 SPARQL execution environment

Generated queries are executed on a Virtuoso endpoint that hosts the April 2024 DBLP knowledge graph.

The endpoint is accessible at 'https://dblp-april24.skynet.coypu.org/sparql' and runs Virtuoso version 07.20.3239. The execution pipeline interacts with this endpoint over HTTP using the default timeout behavior of the SPARQL client library.

[2] https://github.com/jannpf/dblp-grpo

Prior to every execution, a local query cache is checked, if it already contains an answer for the normalized version of the current query. If yes, the cached version is retrieved to limit the number of requests made to the SPARQL endpoint. If not, the query is executed against the endpoint, and the results stored in the cache.

The raw endpoint results are normalized into a canonical representation before comparison with the gold answers and storage in the cache. This includes mapping bindings to tuples, removing ordering differences, and ensuring consistent handling of empty results.

## 3.5 Evaluation Metrics

This section describes the metrics used to assess the performance of the GRPO-trained models. The goal is to evaluate both answer correctness and query formulation quality under a range of conditions. Since the DBLP-QuAD dataset provides gold answers and gold SPARQL queries, metrics are defined at both the answer level and the query level. Additional metrics capture generalization across question categories and template splits.

### 3.5.1 Exact-Match Accuracy (EMAcc)

Exact match accuracy measures how often the answer produced by executing the generated query matches the gold answer set exactly. Let $A_{\text{gen}}$ denote the answer set returned by executing the generated query and $A_{\text{gold}}$ the gold answer set. An instance is counted as correct if

$$A_{\text{gen}} = A_{\text{gold}}.$$

Since answer sets are normalized into sets of tuples, ordering differences are ignored and duplicates are removed. Exact match is the strictest answer-level metric and reflects whether a generated query produces the correct result without extraneous or missing entries.

### 3.5.2 F1 score on answer sets

For cases where the generated answer partially overlaps the gold answer, an $F_1$ score is computed between the two sets. Let $A_{\text{gen}}$ and $A_{\text{gold}}$ be sets of tuples. True positives, false positives, and false negatives are defined in the usual set-theoretic manner and the $F_1$ score is given by

$$F_1 = \frac{2\,|A_{\text{gen}} \cap A_{\text{gold}}|}{2\,|A_{\text{gen}} \cap A_{\text{gold}}| + |A_{\text{gen}} \setminus A_{\text{gold}}| + |A_{\text{gold}} \setminus A_{\text{gen}}|}.$$

This metric captures partial correctness for multi-answer queries and is less brittle than exact match, while still reflecting answer-level semantic accuracy.

### 3.5.3 Category-wise F1

DBLP-QuAD groups questions into semantic categories such as Single Fact, Multiple Facts, Boolean, Negation, Double Negation, Intent, Union, Count, Comparative, and Disambiguation. Following prior work, model performance is analyzed per category using the $F_1$ measure defined above.

Category-wise $F_1$ scores provide insight into which query structures and reasoning patterns a model learns effectively and which categories remain challenging. This is particularly relevant for KGQA, where categories correspond to distinct SPARQL templates with different logical forms.

### 3.5.4 Execution Accuracy (ExAcc)

This metric records the fraction of completions for which the resulting SPARQL query is executable. Executions fail due to syntax errors, extraction failures, or unsupported constructs. Formally, the Execution Accuracy is defined as the percentage of evaluation instances in which both the limited and the original query do not raise an exception during execution.

This measure reflects how reliably the model adheres to SPARQL syntax and to the output format required by the evaluation pipeline.

### 3.5.5 Temporal accuracy (TempAcc)

DBLP contains time-dependent information such as publication years, conference editions, and date-stamped relations. For questions with a temporal component, the generated query must select correctly constrained results.

Temporal accuracy is defined analogously to exact match accuracy, but computed on the subset of evaluation instances that explicitly involve temporal filtering, such as comparisons (‘>=‘, ‘<‘, ‘BETWEEN‘) or temporal relations. This metric isolates the model’s ability to handle time-related constraints in SPARQL.

### 3.5.6 Held-out template accuracy (GenAcc)

To evaluate generalization beyond seen query patterns, the model is tested on a set of queries generated from templates that do not appear in the training set. This held-out split is part of the original DBLP-QuAD design.

Held-out accuracy is computed using the same exact match criterion as in the EMAcc, but restricted to this unseen template partition. This metric assesses whether the model can generalize to new logical forms and not only to surface patterns encountered during finetuning.

# 4 Experiments

This chapter presents the empirical evaluation of the GRPO-trained models on the DBLP-QuAD dataset. The main objectives are to compare GRPO to the base model performance as well as the supervised finetuning baseline with DoRA, and to analyze performance with respect to answer level accuracy, generalization, temporal questions, and question categories. Finally, different reward configurations are compared to assess the effect of each reward function. All experiments use the evaluation metrics defined in Section 3.5 and are conducted on the held-out test split of DBLP-QuAD consisting of 2 000 examples.

## 4.1 Experimental setup and configurations

All experiments share a common compute environment and, for the GRPO models, a fixed training scheme that is used across the main configuration and all reward ablations.

### 4.1.1 Compute environment

All runs are conducted on a single machine equipped with two NVIDIA RTX A6000 GPUs, each with 48 GB of VRAM. For the GRPO experiments, the model is finetuned in a full-parameter setting without parameter-efficient adapters, whereas the supervised baseline uses DoRA adapters as described below. Mixed-precision training with the bfloat16 format is enabled throughout all runs.

Maximum input lengths are constrained by available GPU memory. The combined system and user prompt is padded to a maximum of 668 tokens as that constitutes the maximum length of tokenized inputs encountered in the training set. Generated completions are limited to 1 024 tokens. These limits are applied uniformly across all GRPO iterations and across all ablation variants. The same prompt length and completion limits are also used for the zero-shot and DoRA baselines to ensure a comparable evaluation setting.

### 4.1.2 GRPO training scheme

Training the GRPO models follows a fixed update scheme. To manage GPU memory constraints, 16 gradient accumulation steps are employed. The per-device batch size is set to 4 questions, and for each question the model generates a group of $G = 4$ rollouts. A single forward pass on one device therefore produces 16 sampled

completions. With two devices and 16 gradient accumulation steps, each optimizer update aggregates gradients over 16 micro-batches, corresponding to 512 sampled completions per update step across both GPUs.

Training is performed for a single epoch over the GRPO training set of 7 000 questions, which results in 437 optimization steps under this accumulation strategy. The full list of hyperparameters for GRPO training is provided in Appendix A.2.2.

Training the main GRPO configuration for one epoch requires approximately 35 hours on this two-GPU setup. This runtime includes the cost of repeated SPARQL endpoint calls during rollout evaluation, which constitute the majority of the computational overhead despite the use of query caching. The reported runtime provides a practical estimate of the computational resources required to train small LLMs with GRPO on KGQA tasks of similar scale.

### 4.1.3 Model configurations

As discussed in the methodology chapter, all evaluated models are based on the Qwen3-1.7B architecture and share the same prompt format, tokenization, and data preprocessing pipeline. The following configurations are compared.

#### 4.1.3.1 Zero-shot base model

The unmodified Qwen3-1.7B model is evaluated in a zero-shot setting on DBLP-QuAD using the same prompt template as in the other settings, as given in Appendix A.1. This baseline quantifies the performance of a pretrained model without any DBLP- or SPARQL-specific finetuning.

Two variants are considered: one that includes chain-of-thought (CoT) reasoning in the prompt and expects the model to produce intermediate reasoning before the final SPARQL query, and one that omits CoT and directly prompts for the query. For the CoT variant, many completions, especially for more complex questions, exceed the maximum generation length, which leads to truncated SPARQL queries and a large fraction of execution errors (see Section 4.2). The non-CoT variant avoids this truncation effect and therefore serves as a more conservative estimate of the base model's capability to produce valid SPARQL queries.

#### 4.1.3.2 Supervised finetuning baseline (DoRA)

As a strong non-RL baseline, a supervised finetuning (SFT) model is trained using DoRA adapters on the gold SPARQL queries. The model is optimized with a

standard cross-entropy loss on the target query tokens, conditioned on the DBLP-QuAD prompts. Training is performed for a single epoch over the training split in order to match the GRPO experiments in terms of the number of data passes. The hyperparameters utilized during DoRA training are listed in Appendix A.2.3. This configuration represents a conventional semantic parsing setup in which full gold queries are available during training.

Chain-of-thought prompting is disabled in this setting. The dataset provides only final SPARQL queries, and there is no supervised signal for intermediate reasoning steps. Introducing CoT text in the targets would therefore add a large sequence of tokens that the model cannot meaningfully learn from, since the reasoning content is not part of the supervision objective. As a result, the SFT process would primarily update the model with respect to the final query tokens, while the preceding reasoning would remain essentially unchanged from the base model. For a clean and well-defined baseline, the model is therefore trained to predict only the final SPARQL query without any intermediate reasoning steps.

#### 4.1.3.3 GRPO without gold queries

The first GRPO configuration assumes that only natural language questions and answer sets are available during training, but not the gold SPARQL queries. The reward signal is constructed from a subset of the reward functions described in Section 3.3.3, namely

- Execution Feedback Reward: rewards high similarity between the generated and the reference set of answers resulting from query execution, penalizes non-executable queries
- Structural Coverage Reward: Rewards correct usage of entities and relations given in the prompt
- Format Reward: rewards adherence to the instructed format to include `<think>` tags and to avoid special formatting or explanations
- Length Reward: penalizes excessively long generations, whose length approach the maximum completion length

This setting corresponds to a weakly supervised scenario and tests whether GRPO can learn to generate effective SPARQL queries purely from answer-level feedback.

| Model | EMAcc | ExAcc | F1 |
|---|---|---|---|
| Qwen3 1.7B Base | 0.07 | 0.50 | 0.14 |
| Qwen3 1.7B Base (with CoT) | 0.11 | 0.23 | 0.11 |
| Qwen3 1.7B DoRA (1 Epoch) | **0.69** | **0.91** | **0.78** |
| Qwen3 1.7B GRPO (without gold queries) | 0.47 | 0.83 | 0.48 |
| Qwen3 1.7B GRPO (with gold queries) | 0.44 | 0.85 | 0.45 |

Table 1: Overall answer-level performance on DBLP-QuAD. EMAcc denotes exact match accuracy on answer sets, ExAcc the fraction of queries that execute without errors, and F1 the macro-averaged F1 score over result sets.

#### 4.1.3.4 GRPO with gold queries

The second GRPO configuration adds gold-query-dependent components to the reward signal. In addition to the execution, structural, format, and length rewards described above, it incorporates

- Query Similarity Reward: sentence level BLEU score with the gold SPARQL query
- Query Length Ratio Reward: Acts as regularizer for differences in query complexity, for instance varying number of BGPs and aggregation clauses, filter clauses, etc.

This configuration assumes that gold SPARQL queries are available during training and can therefore combine answer-level feedback with direct supervision on the query form. Comparing this model to the GRPO variant without gold queries allows assessing the effect of gold-query-based rewards within the same RL framework. Reward ablation experiments in Section 4.4 further show the contribution of individual components.

## 4.2 Main results

### 4.2.1 Answer-level accuracy

Table 1 reports the overall answer-level performance on the DBLP-QuAD test set. The metrics comprise exact match accuracy (EMAcc) on the result sets, execution accuracy (ExAcc), which measures the fraction of queries that execute without errors, and the macro-averaged F1 score over result sets.

The zero-shot base model attains low answer-level performance, with EMAcc of 0.07

and F1 of 0.14 when CoT is disabled. Enabling CoT slightly increases EMAcc to 0.11, but is associated with a decrease in F1 from 0.14 to 0.11 and with a marked drop in execution accuracy from 0.50 to 0.23. This behavior is consistent with a large fraction of CoT completions being truncated before a valid query is produced, which reduces the proportion of executable outputs and does not lead to higher answer quality overall.

The supervised DoRA model reaches the highest scores across all three metrics, with EMAcc of 0.69, ExAcc of 0.91 and F1 of 0.78. These values indicate that, when trained with full query supervision, the 1.7B-parameter model can answer a large fraction of DBLP-QuAD questions correctly and produce executable SPARQL queries for most test instances.

Both GRPO models substantially improve over the zero-shot baselines. The GRPO variant trained without gold queries achieves EMAcc of 0.47, ExAcc of 0.83 and F1 of 0.48, which makes it the second-best model overall in terms of EMAcc and F1. The GRPO model with gold-query-based rewards attains slightly lower EMAcc (0.44) and F1 (0.45), but a marginally higher ExAcc of 0.85. In other words, the gold-based variant produces executable queries slightly more often, while the no-gold variant yields a somewhat higher fraction of fully correct answers. In summary, both GRPO configurations close a considerable part of the gap between the zero-shot base models and the fully supervised DoRA baseline, while the DoRA model remains the strongest configuration on this dataset.

It is also instructive to consider the gap between execution accuracy and exact match accuracy. For the base model without CoT and for both GRPO variants, ExAcc exceeds EMAcc by 0.36 to 0.43 points, indicating that many queries execute successfully but still return incorrect answer sets. For the DoRA model this gap is smaller (0.22), which means that a larger proportion of executable queries are also answer-correct. For the CoT base model, both ExAcc and EMAcc are low and the gap between them is relatively small (0.12).

### 4.2.2 Category-wise performance

To analyze performance heterogeneity across question types, Table 2 reports F1 scores by DBLP-QuAD category. The categories comprise Boolean, Count, Disambiguation, Double-Intent, Double-Negation, Multi-Fact, Negation, Single-Fact, Superlative+Comparative, and Union queries.

Across the finetuned models (DoRA and both GRPO variants), categories that corre-

| Model | Bool | Cnt | Disamb | D-Int | D-Neg | Multi | Neg | SF | Sup+Comp | Un |
|---|---|---|---|---|---|---|---|---|---|---|
| Base | 0.00 | 0.10 | 0.07 | 0.04 | 0.00 | 0.09 | 0.00 | 0.25 | 0.14 | 0.04 |
| Base+CoT | 0.16 | 0.05 | 0.08 | 0.08 | 0.06 | 0.10 | 0.12 | 0.44 | 0.00 | 0.01 |
| DoRA | **0.72** | **0.59** | **0.85** | **0.58** | **0.86** | **0.71** | **0.75** | **0.96** | **0.46** | **0.56** |
| GRPO-no | 0.47 | 0.40 | 0.70 | 0.37 | 0.63 | 0.40 | 0.47 | 0.93 | 0.19 | 0.27 |
| GRPO-gold | 0.44 | 0.39 | 0.61 | 0.40 | 0.60 | 0.45 | 0.42 | 0.90 | 0.18 | 0.15 |

Table 2: F1 scores by DBLP-QuAD question category. *Bool* = Boolean, *Cnt* = Count, *SF* = Single-Fact, *Un* = Union, etc. *Base* = Qwen3-1.7B base model; *DoRA* = Qwen3-1.7B with DoRA adapters; *GRPO-no* / *GRPO-gold* = GRPO without/with gold-query rewards.

spond to relatively simple graph patterns, such as Single-Fact and Disambiguation, achieve the highest F1 scores. The DoRA model reaches 0.96 F1 on Single-Fact queries and 0.85 on Disambiguation, while the GRPO models also attain high scores in these categories, with 0.93 and 0.90 on Single-Fact and between 0.61 and 0.70 on Disambiguation. On Single-Fact queries, the gap between GRPO and DoRA is therefore relatively small compared to the differences observed in other categories.

For more complex categories that involve compositional reasoning, such as Union and Superlative+Comparative, performance is noticeably lower for all finetuned models. The DoRA model obtains F1 scores of 0.56 (Union) and 0.46 (Sup+Comp), while the GRPO models reach between 0.15 and 0.27 in these categories.

Intermediate categories such as Boolean, Negation, Double-Negation, Multi-Fact, and Double-Intent fall between these extremes, with the DoRA model consistently yielding the highest F1 and the GRPO variants forming a second tier.

The relative ranking of categories is broadly similar across DoRA and GRPO. Categories where DoRA performs particularly well (for example Single-Fact and Double-Negation) are also among the stronger categories for GRPO, whereas categories where DoRA is weaker (for example Union and Sup+Comp) are likewise challenging for GRPO.

A closer comparison of the two GRPO configurations shows that the variant without gold queries is slightly stronger in most categories. It attains higher F1 scores on Boolean, Count, Disambiguation, Double-Negation, Negation, Single-Fact, Superlative+Comparative, and Union questions, while the gold-based variant has small advantages on Double-Intent and Multi-Fact queries. These differences are moderate in magnitude, but they indicate that gold-query-based rewards do not lead to uniformly higher F1 scores across all question types in this setup.

The base models exhibit a different pattern. They achieve modest F1 scores on simpler categories such as Single-Fact (0.25 without CoT and 0.44 with CoT) and Count or Superlative+Comparative, but remain close to zero on most of the more complex categories. The effect of CoT is mixed: it improves F1 on Boolean, Negation, and Single-Fact questions, but decreases F1 on Count, Superlative+Comparative, and Union queries. For several complex categories, the CoT-augmented base model remains near zero despite these changes. For the CoT variant, the low category-wise F1 scores are consistent with the very low execution accuracy observed in the aggregate results.

### 4.2.3 Generalization and temporal accuracy

As described in Section 3.1.1.2, DBLP-QuAD includes two annotation dimensions that allow a more fine-grained analysis beyond global averages. First, a subset of questions is marked as originating from SPARQL templates that are held out during training, which enables an evaluation of template-level generalization. Second, questions are annotated as temporal if the underlying query involves time- or date-related conditions. Table 3 reports answer-level performance on these two subsets. Here, TempAcc and GenAcc denote exact match scores as defined in Section 3.5, computed on the temporal and held-out subsets, respectively.

| Model | TempAcc | GenAcc |
|---|---|---|
| Qwen3 1.7B Base | 0.09 | 0.06 |
| Qwen3 1.7B Base (with CoT) | 0.10 | 0.15 |
| Qwen3 1.7B DoRA (1 Epoch) | **0.69** | 0.40 |
| Qwen3 1.7B GRPO (without gold queries) | <u>0.52</u> | **0.53** |
| Qwen3 1.7B GRPO (with gold queries) | 0.47 | <u>0.48</u> |

Table 3: Answer-level performance on temporal questions (TempAcc) and on questions derived from held-out templates (GenAcc), reported as F1 scores on the respective subsets.

For temporal questions, the ranking of models mirrors the overall results. The DoRA model attains the highest temporal performance with TempAcc of 0.69, followed by the GRPO models at 0.52 and 0.47, respectively. Both GRPO variants again substantially outperform the base models, whose temporal scores remain around 0.09–0.10. For the supervised DoRA model, temporal performance is somewhat lower than its overall F1 (0.78), whereas for the GRPO variants the temporal scores closely track their aggregate F1 values.

On the held-out template subset, the GRPO models achieve the strongest results. The GRPO configuration without gold queries reaches GenAcc of 0.53, and the GRPO model with gold-query-based rewards reaches 0.48. The supervised DoRA model attains GenAcc of 0.40, while the base models remain well below 0.20. For both GRPO variants, the generalization scores are close to, and slightly higher than, their overall answer-level F1 scores (0.48 and 0.45, respectively), indicating that their behavior on unseen templates is broadly aligned with their aggregate performance on the full test set. In contrast, the DoRA model exhibits a larger drop on held-out templates, from 0.78 overall F1 to 0.40 GenAcc.

The relationship between temporal and generalization performance also differs across models. For DoRA, temporal performance (0.69) clearly exceeds performance on held-out templates (0.40). For the GRPO variants, TempAcc and GenAcc are very similar (0.52 vs 0.53 for GRPO without gold queries and 0.47 vs 0.48 for GRPO with gold queries). The base models are weak in both dimensions, with the CoT variant achieving a slightly higher GenAcc (0.15) than TempAcc (0.10).

## 4.3 Error Analysis

The error analysis is based on two stratified samples of incorrect generations obtained from the test set, where the queries either failed to execute or produce a result set that does not match the gold answers. For the GRPO model trained without gold queries, 1 058 generations were classified as incorrect, from which 100 examples were sampled uniformly at random. For the DoRA baseline, 613 generations were incorrect, and again 100 failures were sampled. These 200 queries were analyzed qualitatively and assigned to a small set of recurring error categories that were induced bottom-up from the observed outputs.

### 4.3.1 Error Categories

Across the sampled failures, six error categories emerged repeatedly. First, queries that require disjunction are often mishandled, for example when a logical union over answer sets is replaced by an intersection or by returning tuples instead of a merged answer set (UNION errors). Second, negation and double negation are frequently misrepresented, especially in the use of ‘FILTER NOT EXISTS‘ and related constructs (negation errors). Third, many failures involve incorrect use of aggregation and superlatives, such as computing the wrong aggregate, aggregating the wrong quantity, or returning the aggregate value instead of the argument that attains it (aggregation errors). Fourth, several queries with multiple intents, multiple outputs,

or multi-step reasoning over the graph exhibit incorrect join structure or missing constraints (multi-intent and multi-fact errors). Fifth, questions involving temporal constraints or comparative reasoning over publication years suffer from misaligned windows, misplaced inequalities, or wrong answer types (temporal and comparative errors). Finally, some generations are syntactically invalid SPARQL, for example due to template artifacts, malformed ‘UNION‘ branches, ill-formed aggregation syntax, or hallucinated schema elements (syntax errors). Queries can exhibit more than one of these patterns simultaneously.

### 4.3.2 GRPO Error Patterns

For the GRPO model, the predominant failure modes are structural. In the 100-sample, 20% of the errors stem from incorrect handling of ‘UNION‘: instead of representing disjunction over publications, venues, or years, the model often forces a shared variable to satisfy both branches simultaneously, which turns an intended union into an intersection and frequently yields empty results. In other cases, the model retrieves information for both branches but represents them as a pair of output variables such as ?year1 ?year2, whereas the reference query uses a single answer variable and a proper union. The resulting answer shape no longer matches the gold result.

Negation and double negation account for another 24% of failures, where reference queries use ‘FILTER NOT EXISTS‘, but the generated queries either drop the filter, attach it to an incomplete pattern, or add spurious negation, which inverts the truth value in 19 of the 100 cases. Generally, for DOUBLE_NEGATION and other Boolean-style questions, the model occasionally switches from ASK in the reference to SELECT in the generated query and returns resources where a Boolean value is expected. In the evaluation, such cases appear as mismatches between True/False reference answers and non- empty or empty result sets returned by the query.

Aggregation and superlatives also account for a share of errors, mainly due to aggregation errors in 21 cases. The model tends to apply ‘MIN‘ and ‘MAX‘ over years without returning the corresponding authors, or replaces a direct lookup of ‘numberOfCreators‘ by a ‘COUNT‘ over existing triples, thereby the semantics of the query do not align with the question.

Multi-intent questions, in particular double-intent queries with two answer slots and disambiguation scenarios, often exhibit wrong traversal direction for predicates, missing “other than” filters, or collapsed projections that only return one of the required outputs (32 cases overall).

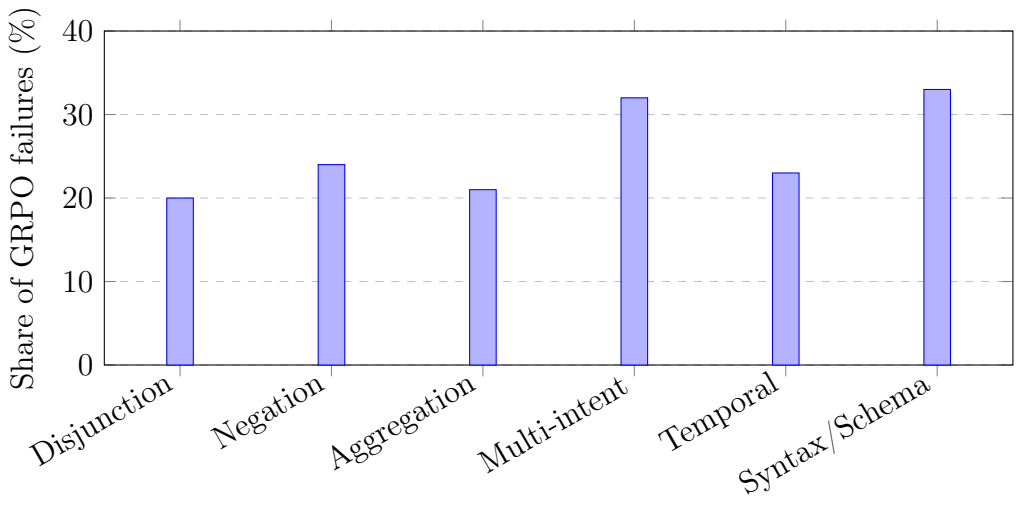


Figure 9: Distribution of error categories in a random sample of 100 incorrect GRPO generations. Bars show the proportion of failures in which each category is present. Categories are not mutually exclusive.

Temporal constraints based on expressions such as ‘YEAR(NOW()) - k‘ are frequently approximated by static ranges, which breaks the intended relative window, and some of these temporal queries also suffer from incorrect comparisons (for instance returning a boolean instead of the earlier of two years). In total, 23 of the 100 cases stem from queries dealing with temporal operations.

Around one third of the sampled GRPO failures result in ‘QueryBadFormed‘ at the endpoint, caused by malformed ‘FILTER‘ clauses, incorrectly grouped ‘UNION‘ branches, incorrect keywords or URIs, or template fragments leaking into the query text. This ’CoT-Leakage’, where overly long thinking content leads to subsequent output truncation, occurred in 8% of cases. The usage of SQL vocabulary such as ‘OR‘, ‘AND‘, ‘FROM‘ and ‘IS NULL‘ was the cause for another 9 syntactical errors. Additionally, hallucinated predicates that do not exist in the schema, despite being given in the prompt template, account for 11% of cases. While these cases are fewer than the purely semantic mismatches, they still represent a noticeable portion of the total failures and prevent any execution.

### 4.3.3 Comparison to DoRA Error Patterns

The distribution over error categories for the DoRA baseline is more concentrated than for GRPO. Syntax and formatting errors are present, but tend to be more local, and many failing queries are structurally close to the reference SPARQL query. Violations of the SPARQL syntax occur less frequently than in the GRPO sample. Instead, DoRA failures are mostly caused by semantic mismatches in aggregation and superlative queries, and by incorrect use of 'UNION' and answer shapes, in line with the higher counts for the aggregation and disjunction categories in the DoRA error sample.

Also, while in the error sample for GRPO both temporal and held-out questions were represented by approximately 20%, the errors in the DoRA sample show a clear prominence in the held-out templates, with 24% temporal questions and 41% of the errors coming from templates that were withheld during training. This is in line with the generalization performance reported in Section 4.2.3.

UNION-related errors remain frequent with 21% and often combine with aggregation, but DoRA more often retains an explicit 'UNION' while misplacing the grouping or adding global filters, instead of dropping the union structure entirely.

Negation and double negation continue to be challenging, although fewer DoRA failures are purely syntactic. Typical problems include inserting 'FILTER NOT EXISTS' into queries that are positive ASK queries which systematically flips the answer, or omitting the filter in cases where it is required. Multi-intent and disambiguation queries sometimes show incorrect join structure or missing constraints.

Finally, both models struggle with mapping Boolean questions to the correct query shape, often generating 'SELECT' queries where the reference uses 'ASK', yielding sets of bindings rather than 'True' or 'False', despite explicitly being prompted to use ASK for yes/no questions.

Overall, the DoRA errors are more concentrated in subtle semantic mismatches within otherwise well-formed queries, while the GRPO-trained model exhibits a higher proportion of syntax errors with malformed SPARQL and more frequent multi-intent, multi-fact and aggregation errors. At the same time, several failure modes are shared between GRPO and DoRA. Both models struggle with disjunction, often confusing unions of answer sets with intersections or with tuple-valued outputs. Both have difficulty expressing negation using 'FILTER NOT EXISTS' and treat double-negation cases inconsistently. Temporal queries and comparative questions over publication years remain challenging for both models, and multi-intent

| Model | EMAcc | ExAcc | F1 | TempAcc | GenAcc |
|---|---|---|---|---|---|
| $R_{\text{exec}}$ | 0.40 | 0.71 | 0.41 | 0.40 | 0.49 |
| $R_{\text{exec,format}}$ | 0.42 | 0.80 | 0.43 | 0.41 | 0.49 |
| $R_{\text{exec,format,struct}}$ | 0.46 | 0.80 | 0.48 | 0.51 | 0.51 |
| $R_{\text{exec,format,struct,len}}$ | **0.47** | 0.83 | 0.48 | **0.52** | **0.53** |
| $R_{\text{exec,format,struct,len,gold(sim,len-r)}}$ | 0.44 | **0.85** | 0.45 | 0.47 | 0.48 |

Table 4: Answer-level, execution, temporal, and generalization performance for GRPO reward ablations. Configurations differ only in the included reward components: execution feedback ($R_{\text{exec}}$), format reward, structural coverage reward, length reward, and gold-query-based similarity and length-ratio rewards.

questions with two required outputs or multiple constraints are a common source of errors in both samples. These similarities suggest that certain aspects of the DBLP schema and question types present intrinsic challenges for small language models, irrespective of the training strategy.

## 4.4 Ablations

To assess the contribution of individual reward components, a series of ablation experiments are conducted for the GRPO models. All configurations share the same architecture, prompt format, data splits, and training schedule described in Section 4.1; only the composition of the reward function is varied. Table 4 reports answer-level accuracy (EMAcc, F1), execution accuracy (ExAcc), and the temporal and generalization scores (TempAcc and GenAcc) for each reward variant.

The execution-only configuration $R_{\text{exec}}$ already yields a substantial improvement over the zero-shot base model (compare Table 1). It attains EMAcc of 0.40 and F1 of 0.41, which accounts for the majority of the performance of the full GRPO configuration without gold queries (EMAcc 0.47, F1 0.48). Execution accuracy reaches 0.71, and the temporal and generalization scores are 0.40 and 0.49, respectively. The gap between ExAcc and EMAcc is 0.31, indicating that a non-trivial fraction of queries execute successfully but still return incorrect answer sets, yet the difference is smaller than for the base model in Table 1.

Adding the format reward ($R_{\text{exec,format}}$) only leads to slight gains across the metrics. EMAcc increases from 0.40 to 0.42 and F1 from 0.41 to 0.43. Temporal accuracy rises slightly to 0.41, while GenAcc remains at 0.49. Execution accuracy shows a clearer improvement, from 0.71 to 0.80. Overall, this configuration preserves most of the

answer-level performance of $R_{\text{exec}}$ but improves the fraction of executable queries.

When the structural coverage reward is added ($R_{\text{exec,format,struct}}$), there is a more pronounced increase in answer-level and subset metrics. EMAcc improves to 0.46 and F1 to 0.48, matching the main GRPO configuration without gold queries in Table 1. Temporal and generalization scores rise to 0.51 on both subsets. Execution accuracy remains at 0.80, so the main effect of the structural reward appears in the correctness of the result sets rather than in execution reliability.

Including the length reward ($R_{\text{exec,format,struct,len}}$) corresponds to the full GRPO configuration without gold queries. Compared to $R_{\text{exec,format,struct}}$, the changes are incremental. EMAcc increases slightly from 0.46 to 0.47, F1 stays at 0.48, and ExAcc rises from 0.80 to 0.83. TempAcc and GenAcc improve from 0.51 to 0.52 and 0.53, respectively. The length reward therefore refines an already strong configuration, with the clearest differences appearing in the execution accuracy.

Finally, the configuration that adds gold-query-based similarity and length-ratio rewards ($R_{\text{exec,format,struct,len,gold(sim,len-r)}}$) corresponds to the GRPO model with gold queries in the main results. This variant achieves the highest execution accuracy among all GRPO models (0.85), but slightly lower answer-level and subset metrics than the best no-gold configuration: EMAcc decreases from 0.47 to 0.44, F1 from 0.48 to 0.45, TempAcc from 0.52 to 0.47, and GenAcc from 0.53 to 0.48. The ExAcc–EMAcc gap grows to 0.41. Across the first four configurations, EMAcc, F1, TempAcc, and GenAcc increase monotonically or stay constant as reward components are added. Including the gold-query-based rewards breaks this trend: execution accuracy continues to improve, while answer-level accuracy and performance on temporal and held-out templates decrease slightly compared to the strongest configuration without gold-query-dependent terms.

# 5 Discussion and Conclusion

## 5.1 Discussion

The experiments in Chapter 4 evaluate how GRPO training affects a small instruction-tuned model on DBLP-QuAD, and how this compares to a supervised DoRA baseline and zero-shot prompting of the base model. In this section, the main findings are interpreted in light of the methodological choices in Chapter 3, before turning to limitations and directions for future work.

### 5.1.1 GRPO versus zero-shot baselines

Relative to the zero-shot base model, both GRPO configurations achieve substantial gains across all answer-level metrics. Exact-match accuracy increases from 0.07 (base, no CoT) and 0.11 (base with CoT) to 0.47 and 0.44 for the GRPO models without and with gold-query-based rewards, respectively (Table 1). Execution accuracy increases from 0.50 and 0.23 to 0.83 and 0.85, and macro-averaged F1 improves from 0.14 and 0.11 to 0.48 and 0.45.

These gains are particularly noteworthy given the challenging prompting setup. The base model with CoT suffers from truncated completions and poor adherence to the required `<think>...</think>` pattern, which results in a large fraction of non-executable queries despite explicit instructions. GRPO training, by combining execution-based rewards with format and length constraints, is able to regularize the same CoT-style prompt into a much more reliable generator of SPARQL queries. In that sense, GRPO can be seen as a mechanism that aligns the model with the desired output protocol under constrained generation budgets, rather than as a purely task-agnostic performance booster.

At the same time, the improvements should be interpreted cautiously. The zero-shot baselines are deliberately strong only in the sense of being instruction-following models; they do not receive any DBLP-specific finetuning. The observed gains therefore show that GRPO can move a small model from a weak baseline to a moderately competent DBLP-QuAD solver, but they do not establish GRPO as the most effective training strategy under all conditions.

### 5.1.2 GRPO versus supervised DoRA

The DoRA baseline achieves the highest overall performance on DBLP-QuAD, with EMAcc of 0.69, ExAcc of 0.91 and F1 of 0.78 after a single epoch of supervised

finetuning on gold queries. In contrast, the GRPO models reach EMAcc of 0.44–0.47 and F1 of 0.45–0.48. The gap is consistent across most question categories, temporal questions, and the full test set.

This suggests that, at least for the present configuration and model size, supervised finetuning remains more effective than GRPO when full SPARQL supervision is available. The DoRA model benefits from direct cross-entropy supervision on the gold query sequence, which provides dense gradient information at every decoding step. GRPO, by contrast, operates on a much sparser and noisier scalar reward signal, even though it incorporates gold-query-based components in one of the variants. Given the relatively small size of the training set (7,000 questions), it is plausible that the supervised objective is simply more sample-efficient under these conditions.

The comparison is also constrained by the training budget. GRPO is run for a single epoch with 437 optimizer steps, and the DoRA baseline is similarly trained for a single epoch to match the number of data passes. It is therefore possible that both methods have not fully converged and that their relative performance could change under a different training schedule. However, given that DoRA already outperforms GRPO by a considerable margin after one epoch, the current results do not support strong claims in favor of GRPO as a replacement for supervised finetuning in settings where gold queries are readily available.

### 5.1.3 Category-wise behavior and task difficulty

The category-wise analysis in Table 2 reveals that all finetuned models, including DoRA and GRPO, perform best on relatively simple query types such as Single-Fact and Disambiguation. On Single-Fact questions, the gap between DoRA (0.96 F1) and GRPO (0.93 and 0.90 F1) is small, and all three methods significantly outperform the base models. This indicates that once the model has learned the basic mapping between entities, relations and answer variables, the remaining difficulty in such categories is relatively low.

In contrast, categories that require more complex logical forms show markedly lower scores. Union and Superlative+Comparative queries are challenging for all methods, with the DoRA model achieving F1 scores of 0.56 and 0.46 and the GRPO variants only reaching 0.15–0.27. Double-intent, multi-fact, and negation-related categories fall in between, with DoRA again consistently ahead and GRPO forming a second tier. These patterns are mirrored in the qualitative error analysis, where both GRPO and DoRA frequently struggle with disjunction, correct use of `FILTER NOT EXISTS`, aggregation semantics, and multi-intent joins.

Taken together, the category-wise results suggest that GRPO is competitive with supervised finetuning on easier, more local patterns. As query structures become more compositional, supervised finetuning maintains a clearer advantage. From a KGQA perspective, this hints at an interesting division of labor: reinforcement learning may be sufficient to align a small model with basic SPARQL patterns and the expected interface, whereas more complex logical phenomena still benefit strongly from dense supervised signals.

### 5.1.4 Generalization and temporal reasoning

One of the most encouraging findings for GRPO is the performance on held-out templates. The GRPO configuration without gold-query-based rewards attains a GenAcc of 0.53 on these unseen template instances, which is higher than both the GRPO variant with gold-query rewards (0.48) and the DoRA baseline (0.40). For GRPO, performance on held-out templates closely matches its overall F1, whereas DoRA exhibits a noticeable drop when moving from the full test set (0.78 F1) to held-out templates (0.40 GenAcc).

This pattern suggests that GRPO may confer a modest advantage in template-level generalization. Because the GRPO objective is driven primarily by answer-level correctness and structural coverage, the model is encouraged to discover alternative query formulations that produce correct answers beyond the specific surface templates seen during training.

In contrast, supervised finetuning is explicitly optimized to reproduce the gold query sequences, which demonstrates a stronger reliance on training templates and thus leads to reduced robustness when confronted with novel question forms, as reflected in the gap of performance in held-out examples for the SFT model.

At the same time, the magnitude of the generalization gains should not be overstated. All models still incur substantial errors on the held-out subset, and GRPO remains clearly behind DoRA on overall performance. In addition, the held-out split is defined at the level of template tuples rather than arbitrary logical forms, so it is unclear how far these gains extend to truly out-of-distribution query structures.

For temporal questions, the picture is more straightforward. The DoRA baseline attains TempAcc of 0.69, while GRPO reaches 0.52 and 0.47, and the base models remain below 0.11. GRPO therefore improves temporal reasoning substantially compared to zero-shot, but does not close the gap to supervised finetuning. The error analysis shows that temporal comparisons and relative windows remain challenging

for both GRPO and DoRA, particularly when combined with other operators such as aggregation or disjunction.

### 5.1.5 Reward design, ablations, and the role of gold queries

The ablation study mainly highlights how much of GRPO's effectiveness already comes from a very simple signal. A model trained only with execution feedback ($R_{\text{exec}}$) recovers the bulk of the improvement over the base model and moves it from single-digit exact match to answer-level scores that are already close to those of the full GRPO configuration without gold queries (see Table 4). In relative terms, the execution-only variant reaches around 85 % of the F1 of the best no-gold GRPO model. This suggests that, for Text-to-SPARQL in the present setup, having access to executed answer sets is by far the most important component. Additional shaping terms mainly fine-tune behavior rather than creating the performance gap to the zero-shot baseline.

The remaining reward components then modulate how this execution signal is used. The format reward primarily stabilizes the interface: it encourages the model to adhere to the requested `<think>...</think>` structure and to return a single SPARQL query, which is reflected in higher execution accuracy without dramatic changes in answer-level metrics. The structural coverage reward sharpens the use of the schema that is provided in the prompt, and its main impact is visible on the more demanding subsets such as temporal and held-out template questions, where it nudges the model towards selecting entities and relations that are more consistent with the underlying graph patterns. The length reward acts as a regularizer on completion length and brings small but consistent improvements, particularly in settings where long CoT traces would otherwise push the query close to the generation limit. In combination, these terms make the policy more disciplined and robust, but they add comparatively little on top of what execution feedback already achieves.

The comparison between GRPO with and without gold-query-based rewards yields a somewhat counterintuitive result. Adding similarity and length-ratio rewards that explicitly reference the gold SPARQL query does not improve answer-level metrics. The variant without gold queries achieves slightly higher EMAcc and F1, although the gold-based variant has marginally higher execution accuracy.

There are several plausible explanations. First, the additional rewards may introduce competing optimization signals. Execution-based correctness already encodes the main supervision, while BLEU-style similarity and length-ratio terms bias the model towards reproducing the exact gold query form. Since multiple semantically

equivalent queries can yield identical answer sets, encouraging the model to stay close to a single reference may penalize valid alternative formulations and increase reward variance. This may, in turn, hinder policy improvement.

Second, the gold-dependent rewards may be more susceptible to reward hacking. The model can increase BLEU or match the gold length without necessarily preserving semantic correctness, for example by copying spurious tokens or template artifacts. The error analysis indicates that some GRPO failures involve malformed queries that superficially resemble the gold structure, which is consistent with this risk.

Third, the shaping rewards tend to saturate early in training. Once the model has learned to produce syntactically valid queries and to respect length constraints, the marginal signal from format, structural and similarity rewards becomes small compared to the execution reward. In this regime, the gold-based rewards may primarily constrain the policy rather than providing additional guidance. The empirical observation that both GRPO variants converge to similar performance supports this interpretation.

Overall, the results do not support strong claims that gold-query-based rewards are beneficial in the present setup. Instead, they suggest that answer-level rewards combined with simple structural and format constraints already suffice to obtain most of the gains over the base model, and that more careful reward design would be required to extract additional benefits from gold queries within an RL framework.

### 5.1.6 Prompting, CoT, and supervision mismatch

A recurring theme in the experiments is the sensitivity of the models to prompt design. The base model with CoT performs poorly, primarily because either the reasoning content is excessively long and the final SPARQL query is subsequently truncated, or due to failure to adhere to the instructed formatting. GRPO mitigates this problem by explicitly rewarding well-formed outputs and penalizing non-executable queries, which induces the model to adapt its CoT behavior to the available generation budget.

For the supervised DoRA baseline, CoT is deliberately disabled during training. Since there are no labels for the reasoning process, only the final query tokens can be used as supervision. Maintaining CoT in this setting would therefore force the model to generate unsupervised reasoning tokens that are not aligned with any training objective and that consume a substantial fraction of the generation budget. Removing CoT for the SFT baseline is thus a pragmatic choice, even if it slightly reduces direct

comparability to the CoT-based GRPO models.

This mismatch illustrates a more general tension between process supervision and outcome supervision in Text-to-SPARQL. GRPO in this work optimizes the outcome (the executed query and its answers), while CoT is treated as an auxiliary channel that is indirectly shaped by format rewards. The experiments show that this is sufficient to make CoT compatible with the task, but they do not address whether explicit process rewards or intermediate supervision could further improve performance.

### 5.1.7 Methodological limitations

Several methodological aspects constrain the scope of the conclusions that can be drawn.

First, the experiments focus on a single base architecture (Qwen3-1.7B) and a single dataset (DBLP-QuAD). While the choice of Qwen is motivated by prior work on RL-friendliness, it remains unclear whether the observed patterns would generalize to other small models such as Llama or Mistral, or to other KGQA benchmarks with different schemas and question distributions. Further, a preprocessed variant of DBLP-QuAD is used with added relation and entity details, which reduces comparability to other benchmarks without this inclusion.

Second, the GRPO configuration itself is relatively narrow, with a fixed group size, clipping parameter and KL coefficient, without systematic exploration of alternative GRPO variants such as Dr. GRPO or adaptive KL schedules. Only a small number of reward weightings are tested, and the analysis of reward saturation remains qualitative. It is possible that different hyperparameter settings or alternative formulations of the same reward components would lead to different trade-offs between stability, efficiency and final performance. The runtime of approximately 35 hours for one GRPO epoch limits the number of runs that can be carried out and made extensive hyperparameter search impractical. In particular, the ablation results are based on single runs per configuration and do not systematically probe training stability or variance across seeds, so conclusions about "promising" reward designs should be read as indicative rather than definitive.

Third, training is performed for a single epoch for both GRPO and DoRA, and there is no systematic study of sample efficiency or convergence behavior. Moreover, random seeds are not controlled explicitly, so some of the results may be subject to stochastic variation that is not fully captured by a single run.

Finally, the experimental setup assumes perfect entity and relation linking. The model receives pre-resolved URIs in the prompt and is not required to perform mention detection or entity disambiguation. This isolates the SPARQL generation problem but also distances the setup from end-to-end KGQA systems, where linking is a major source of error. The conclusions about GRPO therefore apply to the query formulation component in isolation rather than to full KGQA pipelines.

## 5.2 Conclusion

This thesis investigated the use of GRPO for training small language models on zero-shot Text-to-SPARQL over the DBLP knowledge graph. The experimental results provide partial answers to the research questions formulated in Section 1.1.

Regarding the first research question, GRPO has a clear and positive influence on the performance of the small instruction-tuned base model. Starting from a weak zero-shot baseline, GRPO-trained models achieve substantial gains in exact-match accuracy, F1 and execution accuracy, both on the full test set and on temporal questions. They also learn to follow a CoT-style prompting protocol that the base model handles poorly, and they do so without access to gold SPARQL queries in one of the configurations. At the same time, the improvements are bounded. Supervised DoRA finetuning on gold queries remains the strongest overall approach under the studied conditions, with higher answer-level accuracy across most question categories.

With respect to the second research question, the present findings indicate that GRPO is a promising training method for zero-shot Text-to-SPARQL with small models when gold queries are not available, while for a statement on general effectiveness further experimentation is required. In the specific setting of this work, GRPO leverages answer-level rewards and simple structural and format constraints to close a large part of the gap between zero-shot and fully supervised performance. The method seems particularly attractive for its template-level generalization, where the GRPO model without gold-query-based rewards outperforms the DoRA baseline on held-out templates while remaining behind on the overall test set. However, when gold queries are available, supervised finetuning with DoRA achieves better performance at lower computational cost.

Regarding the third research question on reward design, the ablation study points to a clear pattern that a model trained only with execution feedback already recovers most of the final GRPO performance. It substantially improves answer-level and execution metrics over the zero-shot baseline and reaches results close to the best configuration without gold queries. Additional format and structural rewards yield

smaller but consistent gains, mainly by increasing execution reliability and improving behavior on temporal and held-out template questions, and a length reward further fine-tunes this configuration. Within the limits of this work, the combination of execution, format, structural, and length rewards offers the most balanced trade-off between answer accuracy, execution reliability, and generalization, but the bulk of the improvement over the base model is driven by the execution signal alone. By contrast, extending the reward with similarity and length-ratio terms derived from gold queries improves execution reliability further but does not translate into better answer-level or generalization performance, suggesting that naively combining outcome rewards with reference-based shaping terms is not sufficient to match supervised training. Training remained qualitatively stable across all tested configurations, but stability was not evaluated systematically and may depend on hyperparameter choices that were not explored here.

Overall, the results paint a nuanced picture. GRPO can substantially improve the DBLP-QuAD performance of a small model starting from zero-shot, and it offers promising advantages in terms of template-level generalization and alignment with complex prompting schemes. At the same time, the method is computationally demanding and still trails behind supervised finetuning when full supervision is available.

Future work, building on the limitations discussed above, should explore more extensive ablations, alternative GRPO variants, and different base models and datasets. The improvements with respect to template-level generalizations should be investigated on other datasets to clarify if this is a local phenomenon on DBLP or of general nature. For this, GrailQA [59] would be a natural candidate, since it focuses on the generalization ability specifically.

Further, since GRPO natively supports process-level supervision, future work could include defining rewards to shape the reasoning process explicitly, perhaps using query decomposition to construct a learning signal for intermediate steps, to address observed shortcomings when dealing with more complex question categories. Curriculum based strategies offer another avenue of improvement, where the question complexity is increased over training. Lastly, this work was focused on a purely zero-shot setting. Including category-wise query examples in the prompt could be a promising extension, as this approach showed success in the related works.

These extensions could help to further clarify under which conditions GRPO is a competitive or even preferable alternative to supervised Text-to-SPARQL training for small language models.

# References

[1] Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. Towards a cleaner document-oriented multilingual crawled corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4344–4355, 2022.

[2] Hanna Abi Akl. PSYCHIC: A neuro-symbolic framework for knowledge graph question-answering grounding. In *ISWC 2023-International Semantic Web Conference*, 2023.

[3] Bilal Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021.

[4] Marcel R Ackermann, Hannah Bast, Benedikt Maria Beckermann, Johannes Kalmbach, Patrick Neises, and Stefan Ollinger. The dblp knowledge graph and SPARQL endpoint. *Transactions on Graph Data and Knowledge*, 2(2): 3–1, 2024.

[5] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985. ISSN 0364-0213. doi: 10.1016/S0364-0213(85)80012-4.

[6] Waqas Ali, Muhammad Saleem, Bin Yao, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. A survey of RDF stores & SPARQL engines for querying knowledge graphs. *The VLDB Journal*, 31(3):1–26, 2022.

[7] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Computing Surveys (CSUR)*, 50(5):1–40, 2017.

[8] Grigoris Antoniou and Frank Van Harmelen. *A Semantic Web Primer*. MIT press, 2004.

[9] Carlos Aranda, Olivier Corby, Souripriya Das, Lee Feigenbaum, Paula Gearon, Birte Glimm, Steve Harris, Sandro Hawke, Ivan Herman, Nicholas Humfrey, Nico Michaelis, Chimezie Ogbuji, Matthew Perry, Alexandre Passant, Axel Polleres, Eric Prud'hommeaux, Andy Seaborne, and Gregory Todd Williams. SPARQL 1.1 Overview. W3C Recommendation (2013). *URL https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/*, 2013.

[10] Esteban Garces Arias, Meimingwei Li, Christian Heumann, and Matthias

Aßenmacher. Decoding decoded: Understanding hyperparameter effects in open-ended text generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9992–10020, 2025.

[11] Sören Auer, Dante AC Barone, Cassiano Bartz, Eduardo G Cortes, Mohamad Yaser Jaradeh, Oliver Karras, Manolis Koubarakis, Dmitry Mouromtsev, Dmitrii Pliukhin, Daniil Radyush, et al. The sciqa scientific question answering benchmark for scholarly knowledge. *Scientific Reports*, 13(1):7240, 2023.

[12] Caio Viktor S. Avila, Vânia M.P. Vidal, Wellington Franco, and Marco A. Casanova. Experiments with text-to-SPARQL based on ChatGPT. In *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*, pages 277–284, 2024. doi: 10.1109/ICSC59802.2024.00050.

[13] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[14] Stefan Baack. A Critical Analysis of the Largest Source for Generative AI Training Data: Common Crawl. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pages 2199–2208, New York, NY, USA, June 2024. Association for Computing Machinery. ISBN 979-8-4007-0450-5. doi: 10.1145/3630106.3659033.

[15] Franz Baader and Barbara Morawska. Unification in the description logic EL. *Logical Methods in Computer Science*, 6, 2010.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[17] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[18] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report, 2025.

[19] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as

statisticians: Provable in-context learning with in-context algorithm selection. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 57125–57211, 2023.

[20] James K. Baker. Stochastic modeling for automatic speech understanding. In *Readings in Speech Recognition*, pages 297–307. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, May 1990. ISBN 978-1-55860-124-6.

[21] Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. Modern baselines for SPARQL semantic parsing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2260–2265, 2022.

[22] Debayan Banerjee, Sushil Awale, Ricardo Usbeck, and Chris Biemann. DBLP-QuAD: A question answering dataset over the DBLP scholarly knowledge graph. In *BIR@ECIR*, pages 37–51, 2023.

[23] Debayan Banerjee, Sushil Awale, Ricardo Usbeck, and Chris Biemann. Awalesushil/DBLP-QuAD, 2023.

[24] Ankur Bapna and Orhan Firat. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, 2019.

[25] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[26] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003. ISSN ISSN 1533-7928.

[27] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[28] Tim Berners-Lee and Ralph Swick. Semantic web development. Final Techni-

cal Report AFRL-IF-RS-TR-2006-294, AIR FORCE RESEARCH LABORATORY, Rome, New York, September 2006.

[29] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *The Scientific American*, May 2001.

[30] Christian Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, 2009. doi: 10.1109/MIS.2009.102.

[31] Dan Brickley, Ramanathan V Guha, and Brian McBride. RDF vocabulary description language 1.0: RDF schema. W3C recommendation (2004). *URL http://www.w3.org/tr/2004/rec-rdf-schema-20040210*, 2004.

[32] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 1877–1901, Red Hook, NY, USA, December 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

[33] Weiqin Chen, Nhan Pham, Michael Glass, Long Vu, Gaetano Rossiello, Shankar Subramaniam, and Santiago Paternain. ConstrainedSQL: Training llms for Text2SQL via constrained reinforcement learning. In *Annual Conference on Neural Information Processing Systems*, 2025.

[34] Yi-Hui Chen, Eric Jui-Lin Lu, and Kwan-Ho Cheng. Enhancing SPARQL query generation for question answering with a hybrid encoder–decoder and cross-attention model. *Journal of Web Semantics*, 87:100869, 2025. ISSN 1570-8268. doi: 10.1016/j.websem.2025.100869.

[35] Zhuo Chen, Fei Wang, Zixuan Li, Zhao Zhang, Weiwei Ding, Chuanguang Yang, Yongjun Xu, Xiaolong Jin, and Jiafeng Guo. KnowCoder-A1: Incentivizing agentic reasoning capability with outcome supervision for KBQA. *arXiv preprint arXiv:2510.25101*, 2025.

[36] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder ap-

proaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[37] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[38] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[39] Common Crawl Foundation. Common crawl WARC dataset: October 2025 crawl (CC-MAIN-2025-43), 2025.

[40] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747.

[41] Eric Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation (2014). *URL https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/*, 2014.

[42] Jacopo D'Abramo, Andrea Zugarini, and Paolo Torroni. Investigating Large Language Models for Text-to-SPARQL Generation. In Weijia Shi, Wenhao Yu, Akari Asai, Meng Jiang, Greg Durrett, Hannaneh Hajishirzi, and Luke Zettlemoyer, editors, *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 66–80, Albuquerque, New Mexico, USA, May 2025. Association for Computational Linguistics. ISBN 979-8-89176-229-9.

[43] Alexandra DeLucia, Aaron Mueller, Xiang Lisa Li, and João Sedoc. Decoding methods for neural narrative generation. In Antoine Bosselut, Esin Durmus, Varun Prashant Gangal, Sebastian Gehrmann, Yacine Jernite, Laura Perez-Beltrachini, Samira Shaikh, and Wei Xu, editors, *Proceedings of the First Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 166–185, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.gem-1.16.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

[45] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[46] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.64.

[47] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu

Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[48] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*, pages 69–78. Springer, 2019.

[49] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082.

[50] Matthew Finlayson, John Hewitt, Alexander Koller, Swabha Swayamdipta, and Ashish Sabharwal. Closing the Curious Case of Neural Text Degeneration. In *The Twelfth International Conference on Learning Representations*, October 2023.

[51] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, 2017.

[52] Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, February 1994.

[53] Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. In *Second Conference on Language Modeling*, 2025.

[54] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[55] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

[56] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A

Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, 2020.

[57] Joshua T. Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, October 2001. ISSN 0885-2308. doi: 10.1006/csla.2001.0174.

[58] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[59] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond iid: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488, 2021.

[60] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[61] Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40B: Multilingual language model dataset. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4.

[62] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 8342–8360. Association for Computational Linguistics (ACL), 2020.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[64] D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[65] Rose Hirigoyen, Amal Zouaq, and Samuel Reyd. A copy mechanism for handling knowledge base elements in SPARQL neural machine translation. In Yulan He, Heng Ji, Sujian Li, Yang Liu, and Chua-Hui Chang, editors, *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 226–236, Online only, November 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-aacl.22.

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

[67] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

[68] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1152.

[69] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*, September 2019.

[70] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[71] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[72] Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Pan. Less is more: Making smaller language models competent

subgraph retrievers for multi-hop KGQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15787–15803, 2024.

[73] Mohamad Yaser Jaradeh, Markus Stocker, and Sören Auer. Question answering on scholarly knowledge graphs. In Mark Hall, Tanja Merčun, Thomas Risse, and Fabien Duchateau, editors, *Digital Libraries for Open Knowledge*, pages 19–32, Cham, 2020. Springer International Publishing. ISBN 978-3-030-54956-5.

[74] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976. ISSN 1558-2256. doi: 10.1109/PROC.1976.10159.

[75] F. Jelinek. Self-organized language modeling for speech recognition. In *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, May 1990. ISBN 978-1-55860-124-6.

[76] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63, August 2005. ISSN 0001-4966. doi: 10.1121/1.2016299.

[77] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514, 2021.

[78] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9505–9523, 2025.

[79] Longquan Jiang, Xi Yan, and Ricardo Usbeck. A structure and content prompt-based method for knowledge graph question answering over scholarly data. In *QALD/SemREC@ ISWC*, 2023.

[80] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. Online Manuscript Released, 3rd edition, 2025.

[81] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference*

*on Machine Learning*, Icml '02, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

[82] Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, 2023.

[83] Sujan Karki, Pukar Karki, Binay Lal Shrestha, and Tantra Nath Jha. Smaller large language models for text-to-sql: Performance analysis and optimal performance. In *2025 International Conference on Inventive Computation Technologies (ICICT)*, pages 1–7, 2025. doi: 10.1109/ICICT64420.2025.11005216.

[84] Jinseok Kim. Evaluating author name disambiguation for digital libraries: A case of DBLP. *Scientometrics*, 116(3):1867–1886, 2018.

[85] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[86] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, October 2022.

[87] Vijay R Konda and John N Tsitsiklis. Actor-citic agorithms. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 1008–1014, 1999.

[88] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2018.

[89] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[90] Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. A survey on complex knowledge base question answering: Methods, challenges and solutions. In *Proceedings of the Thirtieth International Joint Con-*

*ference on Artificial Intelligence*, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization, 2021.

[91] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning*, pages 26874–26901. PMLR, 2024.

[92] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577.

[93] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *International Symposium on String Processing and Information Retrieval*, pages 1–10. Springer, 2002.

[94] Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. Few-shot in-context learning on knowledge base question answering. In *The 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

[95] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-First International Conference on Machine Learning*, 2024.

[96] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[97] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

[98] M. F. Medress, F. S. Cooper, J. W. Forgie, C. C. Green, D. H. Klatt, M. H. O'Malley, E. P. Neuburg, A. Newell, D. R. Reddy, B. Ritea, J. E. Shoup-Hummel, D. E. Walker, and W. A. Woods. Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, 9(3):307–316, December 1977. ISSN 0004-3702. doi: 10.1016/0004-3702(77)90026-1.

[99] Clara Meister, Ryan Cotterell, and Tim Vieira. If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (Emnlp)*, pages 2173–2185, 2020.

[100] Antonello Meloni, Diego Reforgiato Recupero, Francesco Osborne, angelo salatino, Enrico Motta, Sahar Vahadati, and Jens Lehmann. Exploring large language models for scientific question answering via natural language to SPARQL translation. *ACM Transactions on Intelligent Systems and Technology*, August 2025. ISSN 2157-6904. doi: 10.1145/3757923.

[101] Meta AI. The Llama 4 Herd: The beginning of a new era of natively multimodal intelligence. Blog post, 2025.

[102] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech 2010*, pages 1045–1048, 2010. doi: 10.21437/Interspeech.2010-343.

[103] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*, January 2013.

[104] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 3470–3487. Association for Computational Linguistics (ACL), 2022.

[105] Xuan-Bang Nguyen, Xuan-Hieu Phan, and Massimo Piccardi. Fine-tuning text-to-SQL models with reinforcement-learning training objectives. *Natural Language Processing Journal*, 10:100135, 2025. ISSN 2949-7191. doi: 10.1016/j.nlp.2025.100135.

[106] Natasha Noy and Alan Rector. Defining N-ary Relations on the Semantic Web. W3C Working Group Note (2008). *URL https://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/*, 2006.

[107] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Communications of The Acm*, 62(8):36–43, July 2019. ISSN 0001-0782. doi: 10.1145/3331166.

[108] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen,

Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *CoRR*, abs/2209.11895, 2022.

[109] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[110] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. *CoRR*, abs/2306.01116, 2023.

[111] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

[112] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models. In *The Twelfth International Conference on Learning Representations*, October 2023.

[113] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162.

[114] Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234. IEEE, 2022.

[115] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. In *International Semantic Web Conference*, pages 30–43. Springer, 2006.

[116] Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan O Arik. Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL. In *Second Conference on Language Modeling*, 2025.

[117] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation (2008). *URL https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/*, 2008.

[118] Jiexing Qi, Chang Su, Zhixin Guo, Lyuwen Wu, Zanwei Shen, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. Enhancing SPARQL query generation for knowledge base question answering systems by learning to correct triplets. *Applied Sciences*, 14(1521), 2024. ISSN 2076-3417. doi: 10.3390/app14041521.

[119] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[120] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 53728–53741, 2023.

[121] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[122] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

[123] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[124] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp

Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[125] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[126] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

[127] Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, March 2012. doi: 10.1109/ICASSP.2012.6289079.

[128] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162.

[129] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948. ISSN 1538-7305. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[130] Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

[131] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, April 2024.

[132] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[133] Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, February 2017.

[134] Jinyeop Song, Song Wang, Julian Shun, and Yada Zhu. Efficient and Transferable Agentic Knowledge Graph RAG via Reinforcement Learning, October 2025.

[135] Felix Stahlberg. Neural Machine Translation: A Review. *Journal of Artificial Intelligence Research*, 69:343–418, October 2020. ISSN 1076-9757. doi: 10.1613/jair.1.12007.

[136] Josefa Lia Stoisser, Marc Boubnovski Martell, and Julien Fauqueur. Sparks of tabular reasoning via Text2SQL reinforcement learning. In *The 4th Table Representation Learning Workshop at ACL 2025*, 2025.

[137] Alexis Strappazon, Michael Granitzer, Előd Egyed-Zsigmond, Jelena Mitrovic, and Mehdi Ben Amor. Instruct-to-SPARQL: A text-to-SPARQL dataset for training wikidata agents. In *ACM SIGIR Conference on Human Information Interaction and Retrieval*. ACM, 2025.

[138] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[139] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, 2018.

[140] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0-262-03924-9.

[141] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[142] Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2):79–115, 2005. ISSN 1570-8268. doi: 10.1016/j.websem.2005.06.001.

[143] Dominik Tomaszuk and David Hyland-Wood. RDF 1.1: Knowledge representation and data integration language for the web. *Symmetry*, 12(84), 2020. ISSN 2073-8994. doi: 10.3390/sym12010084.

[144] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023.

[145] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[146] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer, 2017.

[147] Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. Qald-10–the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa. *Semantic Web*, 15(6):2193–2207, 2024.

[148] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[149] Floris Vossebeld and Shenghui Wang. Learning to Refine: An Agentic RL Approach for Iterative SPARQL Query Construction. In Daniil Dobriy, Sanju Tiwari, Jennifer D'Souza, Nandana Mihindukulasooriya, and Francesco Osborne, editors, *Proceedings of the Second International Workshop on Retrieval-*

*Augmented Generation Enabled by Knowledge Graphs (RAGE-KG 2025)*, volume 4079 of *CEUR Workshop Proceedings*, pages 19–32, Nara, Japan, November 2025. CEUR.

[150] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. SUPER-NATURALINSTRUCTIONS: Generalization via declarative instructions on 1600+ NLP tasks. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, 2022.

[151] Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In Chengqing Zong and Michael Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1129.

[152] Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025.

[153] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[154] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837, 2022.

[155] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, 2020.

[156] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696.

[157] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, June 1989. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1989.1.2.270.

[158] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An Explanation of In-context Learning as Implicit Bayesian Inference. In *International Conference on Learning Representations*, October 2021.

[159] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[160] Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. Fine-tuned LLMs know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over Wikidata. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5778–5791, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.353.

[161] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, 2021.

[162] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

[163] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng

Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

[164] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1M technical report, 2025.

[165] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In Chengqing Zong and Michael Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1128.

[166] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 12381–12392, 2019.

[167] Zhiqiang Zhang and Wen Zhao. A collaborative reasoning framework powered by reinforcement learning and large language models for complex questions answering over knowledge graph. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10672–10684, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[168] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. LIMA: Less is more for alignment. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 55006–55021, 2023.

[169] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27, 2015.

# A Appendix

## A.1 Prompt Templates

### A.1.1 System Prompt

```
Your task is to generate a syntactically and semantically correct SPARQL
    query that answers a given natural language question, using only the
    provided entities and relations.

Please follow these instructions:
- carefully analyze the given question, pay special attention to negations
    (not, etc.).
- Think step by step, reasoning through the transformation from question to
     query.
- Enclose your detailed reasoning in <think> ... </think> tags.
- Output the final SPARQL query - without any extra explanation or
    formatting.

Query Generation Rules:
- Only use the provided entities and relations; do not invent or infer
    additional ones.
- Use all provided entities and relations in the query.
- Do not use prefixes; write all URIs in full.
- By default, use SELECT DISTINCT in your queries, unless the context
    clearly requires otherwise.
- For yes/no questions, always use the ASK keyword to obtain a boolean
    result
- Carefully consider whether the answer should be the subject or object in
    each relevant triple pattern.
- Always use single quotes for literals

Example output format:
<think> Step-by-step reasoning here. </think> SPARQL query here
```

### A.1.2 User Prompt

```
Generate a SPARQL query to answer the following question.

Question: {question}
Relevant Entities: {entities}
Relevant Relations: {relations}
```

## A.2 Hyperparameters

### A.2.1 Decoding Parameters

Table 5: Generation hyperparameters used for inference.

| Parameter | Value |
|---|---|
| Base model | Qwen3-1.7B (chat) |
| Max input length | 700 tokens |
| Max new tokens | 1024 tokens |
| Temperature | 0.6 |
| Top-$p$ | 0.95 |
| Min-$p$ | 0.0 |
| Top-$k$ | 20 |
| Beam search | disabled |
| Repetition penalty | 1.0 |
| Padding side | left |

### A.2.2 GRPO Training Parameters

Table 6: Hyperparameters for GRPO training.

| Parameter | Value |
|---|---|
| Base model | Qwen3-1.7B (chat) |
| Training set size | 7000 |
| Batch size (per device) | 4 |
| Gradient Accumulation Steps | 16 |
| Group size | 4 |
| Max new tokens | 1024 |
| Learning rate | 1e-6 |
| Optimizer | AdamW |
| Beta 1 / Beta 2 | 0.9 / 0.999 |
| Weight decay | 0.0 |
| LR schedule | linear |
| Advantage normalization | per-group |
| KL penalty ($\beta$) | 0.04 |
| KL target policy | initial pretrained model |
| Gradient clipping ($\varepsilon$) | 0.2 |
| Mixed precision | bfloat16 |

### A.2.3 DoRA Fine-tuning Parameters

Table 7: Hyperparameters for DoRA supervised finetuning.

| Parameter | Value |
|---|---|
| Base model | Qwen3-1.7B (chat) |
| Training epochs | 1 |
| Batch size (per GPU) | 8 |
| Gradient accumulation | 8 |
| Effective batch size | 64 |
| Learning rate | 2e-5 |
| Optimizer | AdamW |
| Weight decay | 0.0 |
| LR schedule | linear |
| Warmup ratio | 0.0 |
| Max sequence length | 1024 tokens |
| Loss | Cross-entropy on target SPARQL |
| DoRA rank | 8 |
| DoRA alpha | 32 |
| DoRA dropout | 0.05 |
| Target modules | q, k, v, o projections |
| Mixed precision | bfloat16 |

## Note on AI Usage

This thesis was written with the help of the following Artificial Intelligence tools and services:

- LanguageTool from languagetool.org for grammar checking
- ChatGPT 5.1 by OpenAI for discussion and improvement of wording and formulation
- Qwen 3 Coder on ChatAI by GDWG for debugging

## Declaration of Academic Honesty

I, Jann Pfeifer, hereby declare that I have not previously submitted the present work for other examinations. I wrote this work independently. All sources, including sources from the Internet, that I have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables, and images), have been acknowledged by me as such. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

---

Jann Pfeifer
Lüneburg, December 10, 2025