

# National Institute of Informatics Internship Report

Batuhan Karaca

March 20, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background Research</b>	<b>2</b>
2.1	The Book: Knowledge Graphs . . . . .	2
2.2	Reading about LLMs . . . . .	2
2.2.1	Markov Property and Markov Chain . . . . .	2
2.2.2	Recursive Neural Networks . . . . .	3
2.2.3	Transformers . . . . .	3
2.3	The Paper: Unifying Large Language Models and Knowledge Graphs: A Roadmap . . . . .	4
<b>3</b>	<b>Task: Relation Extraction via Manual Prompting</b>	<b>4</b>
<b>4</b>	<b>Task: Prompt Tuning via Light-weight LLM</b>	<b>5</b>
4.1	Reinforcement Learning Concepts . . . . .	6
4.1.1	Bellman Equation . . . . .	6
4.1.2	Value Based Methods . . . . .	6
4.1.3	Policy Gradient Methods . . . . .	7
4.2	LoRA (Low-Rank Adaptation of Large Language Models) . . . . .	8
4.2.1	RS-LoRA (Rank-Stabilized LoRA) . . . . .	8
4.2.2	Q-LoRA(Quantized LoRA) . . . . .	8
4.2.3	LoftQ (LoRA-Fine-Tuning-aware Quantization) . . . . .	9
4.3	Metrics . . . . .	9
4.3.1	Evaluation Metric: F1 score . . . . .	9
4.3.2	Reward Metrics . . . . .	10
4.4	Implementation . . . . .	11
4.4.1	Dataset . . . . .	11
4.4.2	Model and Training . . . . .	12
4.4.3	Evaluation . . . . .	12
4.5	Hyperparameter Tuning . . . . .	13
<b>5</b>	<b>Self-Reflection and Future Directions</b>	<b>13</b>
<b>6</b>	<b>Appendix</b>	<b>13</b>
6.1	Manual Prompting Prompts . . . . .	13
6.1.1	ChatGPT . . . . .	13
6.1.2	Phi-3-mini . . . . .	15
	<b>References</b>	<b>18</b>

## 1 Introduction

I am Batuhan Karaca, a 2nd year MSc student in the University of Freiburg. I was selected for the National Institute of Informatics (NII) internship program for the winter period, 1 October 2024-29 March 2025, to work under Prof. Hideaki Takeda. His main area is *knowledge graphs*.

In the section *Background Research*, the concepts in the related fields, mainly KGs (knowledge graphs) and LLMs (large language models) that are read on are summarized. The tasks are explained in the later sections. Finally, what could have been done better and how the work could be improved in the future is discussed.

## 2 Background Research

The coworkers were PhD students specialized in different areas of KGs. Because I did not have background knowledge in the area, the supervisor first recommended the book "Knowledge Graphs" by Hogan et al. Reading the book took about 1 month. The first part briefly describes the contents of the book. After the book, due to my interest in deep learning and transformer models, the supervisor suggested the paper *Unifying Large Language Models and Knowledge Graphs: A Roadmap* that describes the methods so far in LLM community being used with KGs. I had prior knowledge on transformers due to a deep learning course at the MSc studies. However, I felt the need to polish and expand my knowledge on LLMs in order to understand the concepts in the paper and the future studies. Using the knowledge acquired in BSc, MSc and personal studies, I started to read about the history of LLMs. This process is described in the next part. After feeling ready, I stopped LLM research and started reading the paper, to which the focus is shifted to in the final part. These processes took about 2 weeks.

### 2.1 The Book: Knowledge Graphs

The book starts with a description of different types of a graph, such as a *edge-labeled directed graph*, with a group of nodes or *entities* where some of the node pairs are connected with one or more edges pointing towards either of the nodes. Each node pair that is connected by an edge is called a *binary relation* or a *triple*. The relations need not be binary, yet they are most of the time for convenience. In this regard, they are similar to a database. Nodes and edges should have an identifier known as *label*. Alongside these labels, another type of graph may include a number of properties for each node and edge, called a *property graph*.

KGs show numerous similarities with a DB (*database*). The stored information could be queried using specific tools such as *SparQL*, a query language similar to SQL. Furthermore, they could have a schema. Relational DBMSs (database management systems) could constrain which group of columns identify a row, the values that the columns could take, and so forth. Given an ontology, there could be multiple ways of describing the same path in a graph that is consistent (or *positive*) with the ontology. A KG (or a DB) with CWA (*closed world assumption*) has no *negative* relations that contradict its constraints, and OWA (*open world assumption*) vice-versa. KGs on the other hand, could govern a much more flexible approach, called *ontology*. Ontologies are a number of logical rules deciding the relationships between the nodes and edges. These paths *entail* and are *entailments* of each other.

Ontologies could be inferred from the graphs using traditional machine learning methods such as rule mining, or rather more novel ones such as NNs (neural networks). Rule mining infers new rules based on the frequency (*confidence*) of the edges in an existing graph. NNs could be trained to output specific ontology features such as class and type; or the aforementioned confidence score. We could then use hidden features of the network for denser representations (*knowledge graph embeddings*) of the graph elements to use for other applications. Recent approaches use transformers such as BERT and LLMs more on which is later. People also tried to create NNs that learn the graphs itself (*graph neural networks*) in the past.

There is another subfield that is interested in constructing the graph from different types of sources as unorganized as plain text, or structured ones such as markup text and tables. When it comes to semantic text, LLMs are preferred compared to methods crafted by hand. The book touches upon how to assess the quality of the constructed graph in terms of metrics. In addition, it points out *refinement* the concept of making the graph consistent with its ontology by adding positive edges (*completion*), and removing negative edges (*correction*)[1].

### 2.2 Reading about LLMs

#### 2.2.1 Markov Property and Markov Chain

Markov property is defined for a stochastic process when the future only depends on the present. The mathematical definition is

$$p(x_t | x_{t-1:0}) = p(x_t | x_{t-1}, x_{t-2}, \dots, x_0) \quad (1)$$

$$= p(x_t | x_{t-1}) \quad (2)$$

where  $p$  denotes probabilities. Markov property is a fundamental assumption that facilitates working with temporal data, whether it is the current token of a sentence (natural language processing) or the current state of an environment (reinforcement learning). This assumption seems limiting at the first glance. However, the tokens in the past still indirectly influence the current token. We could also use a running window (e.g. LLM context window) to encapsulate a set of tokens as a compound token [2].

### 2.2.2 Recursive Neural Networks

Assuming we have a sequence  $x_{1:T} = x_1, x_2, \dots, x_T$  a RNN (*Recursive Neural Network*) layer computes

$$h(x_t) = f\left(\underbrace{h(x_{t-1})w_{(t-1)0} + x_{t-1}w_{(t-1)1} + b_{t-1}}_{g(x_{t-1})}\right) \quad (3)$$

$$(4)$$

Where  $f$  is an activation function. It is easy to see that RNNs have the Markov property. Using the chain rule we have

$$\frac{\delta h(x_t)}{\delta h(x_{t-t_0})} = \prod_{\tau=1}^{t_0} \frac{\delta h(x_{t-\tau+1})}{\delta h(x_{t-\tau})} \quad (5)$$

$$= \prod_{\tau=1}^{t_0} \frac{\delta h(x_{t-\tau+1})}{\delta g(x_{t-\tau})} \frac{\delta g(x_{t-\tau})}{\delta h(x_{t-\tau})} \quad (6)$$

$$= \prod_{\tau=1}^{t_0} f'\left(g(x_{t-\tau})\right) w_{(t-\tau)0} \quad (7)$$

For a stable activation flow, we need

$$f'\left(g(x_{t-\tau})\right) w_{(t-\tau)0} = \pm 1 \quad (8)$$

$$f\left(g(x_{t-\tau})\right) = \pm \frac{g(x_{t-\tau})}{w_{(t-\tau)0}} + c \quad (9)$$

where  $c$  is a constant w.r.t (with respect to)  $g$ .  $f$  is a linear function w.r.t  $g$ , unlike tanh that is used with conventional RNNs. Deep network layers need to have nonlinear components as well. Therefore, Hochreiter et al. decided to use nonlinear activations in the input  $g(x_{t-\tau})$  itself. Each nonlinear structure is called a *gate*. This overall layer is called LSTM (long short-term memory) and could be stacked just like a standard RNN layer [3]. Some works stack two inverted RNN layers in order to enable the network to see future tokens as well as the past ones, called BiRNN (bidirectional RNN).

At the time, the NLP (natural language processing) community used a stack of RNN layers that grabs the contextual information out of the input (*encoder*), and another stack of layers that processed this information for the output (*decoder*) for tasks involving sequence transformation (i.e. translation). The current token is accepted at each layer, whereas the decoder layer output is used as the prediction for the next token.

Having different variants, (*cross*) *attention* between vectors  $a, b$  is (with a similarity matrix  $S$ )

$$S_{ij} = a_i b_j \quad (10)$$

$$A(a, b)_i = \bar{b}_i = \sum_j S_{ij} b_j \quad (11)$$

The attended vector  $\bar{b}$  takes into account the similarity between both vectors.

Bahdanau et al. obtain candidate vectors at each layer of a gated BiRNN encoder with a full sweep. Then they use attention between the candidate vectors and the current decoder hidden state to obtain a context vector. Finally, they feed the context vector alongside the current hidden state to obtain the next hidden state for the current gated decoder layer [4]. ELMo (Embeddings from Language Models, Peters et al.) on the other hand uses deep BiRNNs and concatenates the hidden state vectors. These vectors then could be further concatenated with the input for other NLP tasks [5]. Parikh et al. and Seo et al. use bidirectional (two-way) cross-attention [6, 7].

Lastly, it is important to mention two different training modes for RNNs. When training at the current step, we could feed the current label (correct output) to the network. This is called *teacher forcing* as the name implies. When generating its own output, the network does not have the labels. Teacher forcing resembles spoon learning in an informal sense. There is an alternative called *curriculum learning* that feeds the network its current prediction as the next token. This approach is slower yet achieves better results [8].

### 2.2.3 Transformers

The transformers paper (Vaswani et al.) brought a new paradigm. The first transformer uses an architecture similar to the aforementioned encoder-decoder (*seq2seq*) model. Transformer encoder and decoder blocks process the inputs in parallel unlike RNNs' which operates sequentially. In order to obtain a single output token, the last decoder layer is connected to a FFN (feed forward neural network). The parallelism enables a different attention mechanism at each layer, where the

sentence would be compared to itself (self-attention). This allows for faster inference. However, due to its parallel nature, the cross-attention in decoder layers may include future tokens as well. We are not interested in the future tokens that are not available yet, hence the parts corresponding to them are masked [9].

The transformer boom brought different variants. BERT (Bidirectional Encoder Representations from Transformers, Devlin et al.) is the encoder part of the transformer. It is trained on two tasks. The first task is MLM (masked language modeling) where the network is trained to predict a number of masked tokens for each given sentence. In a typical transformer layer, the future tokens also see the past tokens. In the following layers the current token would indirectly link to itself and the similarity matrix entry corresponding to it would have the highest activation (which is reasonable as it is the same token). Masking the tokens prevents this phenomenon. There is also NSP (next sentence prediction) task where the network is trained to predict the next sentence as the name implies. The authors claim and show, the same *pretrained* network excels at wide range of tasks "such as question answering and language inference" after only little more training, called *fine-tuning* [10]. Due to requiring less compute-time, fine-tuning becomes a widely adopted method in the community. GPT (Generative Pretrained Transformer, Radford et al.) on the other hand, uses the decoder part of the transformer [11]. Three branches (encoder only, decoder only and encoder-decoder) of transformers continue developing.

GPT-3 (Brown et al.) removes the fine-tuning step altogether, proposing *manual few-shot prompting* with hard labels and frozen weights after pretraining [12]. Prompt engineering becomes another paradigm of research. Schick and Schütze propose *prompt tuning* modelling the problem as a sequence classification task. They fine-tune an ensemble of PLMs (pretrained language models) to predict the relevant masked part, alleviating the contextual information in the prompt given beside the query. Then the PLMs soft-label a larger prompt dataset, giving rise to automatic prompting [13].

### 2.3 The Paper: Unifying Large Language Models and Knowledge Graphs: A Roadmap

The paper divides the approaches into three parts. KG-enhanced LLMs aim to increase accuracy of the answers (i.e. prevent hallucinations) by introducing KGs as structured information to the LLM. LLM-augmented KGs, on the other hand utilizes LLMs in the construction and refinement of KGs. There is another approach called Synergized LLMs + KGs [14] to which there was not time to read and understand. I could only read KG-enhanced LLMs.

KG-enhanced LLMs are further divided. Firstly, there are works using KG information in the training objective [14]. GLM (Graph Guided Mask LM) uses KGs to mask each entity with a sample. Sampling weight for each entity depends on document frequency and shortest distance to the target entity. They also sample a *negative entity* as they call it. The similarity score between the positive-negative pair is added to the MLM loss to improve generalization [15]. EBERT has a masking method that adaptively switches from words (MLM) to more complex domain relevant phrases. It further uses an association graph to sample neighboring pairs, creating semantic-aware embeddings for each phrase of a pair using cross attention [16].

Some methods aim to fuse textual and knowledge information [14]. ERNIE adapts the encoder only transformer (i.e. BERT) to the structured knowledge. It stacks a number of textual encoder (T-encoder) layers which do what a typical encoder does with a sentence input. The difference comes with knowledgeable encoder (K-encoder) that comes after it which takes the entities as input. Each K-encoder layer mixes the text-knowledge pair, outputting another text-knowledge pair that are more aware of each other for the next layer [17]. On the other hand, there are methods that include KG information in the input. For instance, ERNIE 3.0 is trained with each relation triple concatenated with its corresponding text as input [18]. Some works are prompt-tuning based. KP-PLM approach uses knowledge-aware prompts next to contextual text, then fine-tunes a PLM on this input [19].

The methods above do not adapt well to the changing data, requiring retraining for the new samples. There is another branch of studies focused on using the knowledge during inference [14]. In the REALM paper, Given a context  $x$ , encoded as dense vectors as query; top scoring encoded document variables  $z$  are obtained via MIPS (Maximum Inner Product Search, which works like the attention mechanism) from a KB (knowledge base). Later,  $x$  and  $z$  are concatenated as input to a generator that is pre-trained on MLM and fine-tuned on open domain QA. The (input and document) encoders and the generator are BERT models. The pipeline is trained end-to-end. For a stable optimization, the document encoder's parameters are updated periodically [20]. RAG (retrieval augmented generation) uses an encoder-decoder (seq2seq) transformer (BART) as the generator and fixed document encoders [21]. Story fragments adds a short-term memory in which newly seen passages are added. This memory indirectly contributes salience information [22]. EMAT uses MIPS search between a query and a number of keys each mapping to a value in a memory. The obtained key-value pairs are incorporated into the encoder part of a seq2seq transformer [23].

## 3 Task: Relation Extraction via Manual Prompting

The supervisor suggested a simple idea. I was expected to have an LLM model output triples for a given passage via prompt engineering. The LLM was going to process the prompt without intermediary steps during inference. I started searching along the lines of *graph construction via prompting*, expecting to find abundant materials. The works that are found if any, either had prompts that are not suitable for task, or were using complex approaches that are not end-to-end. Most of the works are only skimmed through. One of such works is by Carta et al. which uses multiple steps (as the name "iterative"

in the title) [24]. The scarcity of the results was probably due to the terms given to the search engine. I could have been more descriptive or used different words. In the last month of the internship period, replacing the phrase *graph construction* with its formal counterpart *relation extraction* in the search term, resulted in more promising results. I was not familiar with the term *relation extraction* at the start of the internship. Results were not checked due to limited time.

Query part of the most promising prompt that Zhu et al. is used with some modifications. With their prompt Zhu et al. claim to achieve a micro-F1 score of 4.4 and 7.2 using ChatGPT web interface and the GPT-4 model on SciERC dataset respectively. However, they do not describe whether they fine-tuned GPT-4 and which model they used for the ChatGPT and so forth [25]. The prompts and their results are given in part 6.1. List of predicates could grow with number of edge classes a KG has. This could be a problem when *context window* (maximum number of tokens input has) of an LLM is smaller than the prompt, though this never happened with the models that are tried. Nevertheless, another version of the prompt which does not have the list of predicates is also used in the experiment.

Before proceeding, there are problems to point out on accuracy of the outputs. Firstly, it is challenging to find a suitable ground truth. Due to the constrained nature (i.e. schema and ontology) of KGs, the number of classes that the nodes and edges could take is finite. Assuming CWA, if a KG represents the real world accurately (without any negatives in the real world data that the LLM is trained on), it could be used as a ground truth for evaluation. In our case, list of predicates is part of such constraints, yet they may not be able to cover all predicates in the real world. Secondly, the LLM may give a set of outputs that entail a set of triples in the label set. In this case, the KG is accurate yet the LLM performs worse as these outputs are not true-positive. In other words, an LLM output could either be negative or positive, yet not in the label set. For brevity, CWA is assumed, excluding the problematic positive outputs from true-positives. Lastly, the *power* or how accurately the LLM represents the ground truth is important as well. The number of parameters is one of the factors that influence this variable to some extent. The LLMs with higher number of parameters generally tend to be more powerful (perform better).

Few inputs with 0, 1 and 2-shot prompts are tested. A prompt may differ due to its number of *demonstrations* where each demonstration is an input-label pair, similar to a supervised setting. An N-shot prompt uses N demonstrations. The query input is given at the end. TACRED dataset is preferred as the ground truth which has plenty of competition entries for F1 score evaluation [26, 27]. It still has some inherent problems as well, discussed in detail in Re-TACRED paper [28]. ChatGPT free plan (which used the GPT-4o at the time) and Phi-3 mini are used. The authors show that with only 3.8 billion parameters, Phi-3 mini performs close (or sometimes better) compared to models as large as 7 billion parameters. The variant with smaller context window of 4000 tokens is used (there is also a variant with 128000) [29]. Calculating the token length for the prompts, the maximum value was 668, making larger context windows unnecessary due to memory concerns.

Without the list of predicates, the models were creating their own *explicit* relations. The model is not forced to match phrases to any TACRED predicates. Adding the list of predicates to constrain the model did not prevent models from outputting irrational outputs as well. For instance, ChatGPT outputted predicate values where they should not be (i.e. in place of tail values). Phi-3-mini performed worse. For some of the prompts, it did not comply with the prompts (i.e. plain Python list), giving differently formatted or incomplete outputs. Because Phi-3 is a family of open-source models, it is possible to adjust their generation parameters such as *temperature*. Temperature = 0 as suggested in the paper without sampling (i.e. greedy search). The generation parameters could have been manually fine-tuned. However, I decided to see if automatic fine-tuning is possible. There are existing hyperparameter tuning methods for cross-validation which seemed tricky to implement for manual prompting.

## 4 Task: Prompt Tuning via Light-weight LLM

Due to the problems in the previous part, another idea was conceived. A lightweight LLM was going to be fine-tuned on a set of sentence-triple pairs, then the model performance was going to be evaluated. Separate model wasn't needed for prompt generation. Only a single model would be trained end-to-end, bearing the name *end-to-end prompt tuning* for the project. Good enough fine-tuning could make a lightweight model more powerful. The scale of the model could be a plus for individual applications.

It is possible to train a NN using plain supervised learning. However, I did not want to stop here and wanted to experiment with different training methods. There is a ton, yet starting simple would be more preferable. LLMs and RL methods are known to have Markov property. I decided to read on RL. I was looking for a book that also included implementations with the theory, so I came across *Deep Reinforcement Learning Hands On* by Maxim Lapan. I read the book until the end of *Chapter 14: Training Chatbots with RL* which took about 2 weeks. The implementation is based on seq2seq RNNs which is adapted to a decoder only LLM in the task. After Deepseek-R1 came out, its authors claimed that they were also using a similar technique to train their model. The technique is a *policy gradient method*, called GRPO (*Group Relative Policy Optimization*) [30, 31, 32]. *REINFORCE* which is one of the least sophisticated methods to implement in this family is adopted. At the time, the implementation was already ready and reimplementing GRPO seemed risky.

In the upcoming two parts, the concepts from the book and other resources that are relevant to the task are explained. In the final two parts, the metrics used during training and evaluation, with the implementation details at the end are described.

## 4.1 Reinforcement Learning Concepts

RL agent acts on a tuple  $(s_t, r_t)$  at time step  $t$  where  $s_t$  is the current state and  $r_t$  is the reward resulting from  $s_t$ . Each action  $a_t$  taken by the agent results in another tuple at time step  $t + 1$ . It is easy to see that this has the Markov property.

For any initial state  $s_t$ , the cumulative reward is defined as

$$g_t = \lim_{N \rightarrow \infty} \sum_{\tau=0}^N \gamma^\tau r_{t+\tau} \quad (12)$$

$$\sum_{\tau=0}^N \gamma^\tau r_{\min} \leq \sum_{\tau=0}^N \gamma^\tau r_{t+\tau} \leq \sum_{\tau=0}^N \gamma^\tau r_{\max} \quad (13)$$

$$\frac{\gamma^{N+1} - 1}{\gamma - 1} r_{\min} \leq \sum_{\tau=0}^N \gamma^\tau r_{t+\tau} \leq \frac{\gamma^{N+1} - 1}{\gamma - 1} r_{\max} \quad (14)$$

$$\lim_{N \rightarrow \infty} \frac{\gamma^{N+1} - 1}{\gamma - 1} r_{\min} \leq \lim_{N \rightarrow \infty} \sum_{\tau=0}^N \gamma^\tau r_{t+\tau} \leq \lim_{N \rightarrow \infty} \frac{\gamma^{N+1} - 1}{\gamma - 1} r_{\max} \quad (15)$$

$$\frac{r_{\min}}{1 - \gamma} \leq g_t \leq \frac{r_{\max}}{1 - \gamma} \text{ if } 0 \leq \gamma < 1 \quad (16)$$

As long as the rewards are bounded and  $0 \leq \gamma < 1$ , the cumulative reward will always be bounded as well, which is preferred. Getting diminishing returns as a result prevents the agent to go for infinite loops for infinite gain, encouraging to finish as soon as possible.

### 4.1.1 Bellman Equation

$$V(s_t) = \sum_{g_t} p(g_t | s_t) g_t \quad (17)$$

$$= E(g_t | s_t) \quad (18)$$

$$= E(r_t + \gamma g_{t+1} | s_t) \quad (19)$$

$$= E(r_t | s_t) + \gamma E(g_{t+1} | s_t) \quad (20)$$

$$= \sum_{r_t} p(r_t | s_t) r_t + \gamma \sum_{g_{t+1}} p(g_{t+1} | s_t) g_{t+1} \quad (21)$$

$$= \sum_{r_t, a_t, s_{t+1}} p(r_t, a_t, s_{t+1} | s_t) r_t + \gamma \sum_{g_{t+1}} \sum_{r_t, a_t, s_{t+1}} p(g_{t+1} | r_t, a_t, s_{t+1}, s_t) p(r_t, a_t, s_{t+1} | s_t) g_{t+1} \quad (22)$$

$$= \sum_{r_t, a_t, s_{t+1}} p(r_t, a_t, s_{t+1} | s_t) \left[ r_t + \gamma \sum_{g_{t+1}} p(g_{t+1} | r_t, a_t, s_{t+1}, s_t) g_{t+1} \right] \quad (23)$$

$$= \sum_{r_t, a_t, s_{t+1}} p(r_t, a_t, s_{t+1} | s_t) \left[ r_t + \gamma \sum_{g_{t+1}} p(g_{t+1} | s_{t+1}) g_{t+1} \right] \quad (24)$$

$$= \sum_{r_t, a_t, s_{t+1}} p(a_t | s_t) p(r_t, s_{t+1} | a_t, s_t) [r_t + \gamma V(s_{t+1})] \quad (25)$$

$$= \sum_{a_t} p(a_t | s_t) \sum_{r_t, s_{t+1}} p(r_t, s_{t+1} | a_t, s_t) [r_t + \gamma V(s_{t+1})] \quad (26)$$

$$= \sum_{a_t} \pi(a_t | s_t) Q(a_t, s_t) \quad (27)$$

$$V(s_t) = E_\pi [Q(a_t, s_t)] \quad (28)$$

$$Q(a_t, s_t) = E_{r_t, s_{t+1}} [r_t + \gamma V(s_{t+1})] \quad (29)$$

$V, Q, \pi$  are called *value function*, *Q-function* and *policy* respectively.

### 4.1.2 Value Based Methods

*Value based methods* assume a fixed policy (i.e. choosing the maximum value). In the expression 29, when we fix the policy such that the possibility of the maximum value is 1 we have

$$V^*(s_t) = \max_{a_t} [Q(a_t, s_t)] \quad (30)$$

Alternatively, we could visualize this phenomenon as a graph in figures 1 and 2. The agent maximizes the reward by climbing the tree in figure 2 in a maximin fashion. For real life scenarios construction the tree could be costly. Therefore, people generally prefer using sampling methods such as Monte Carlo.

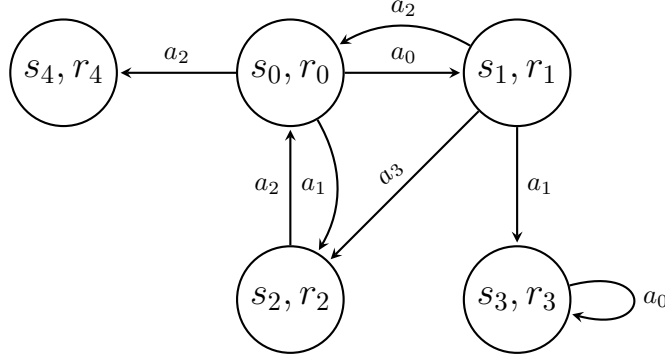


Figure 1: A sample graph of a state space

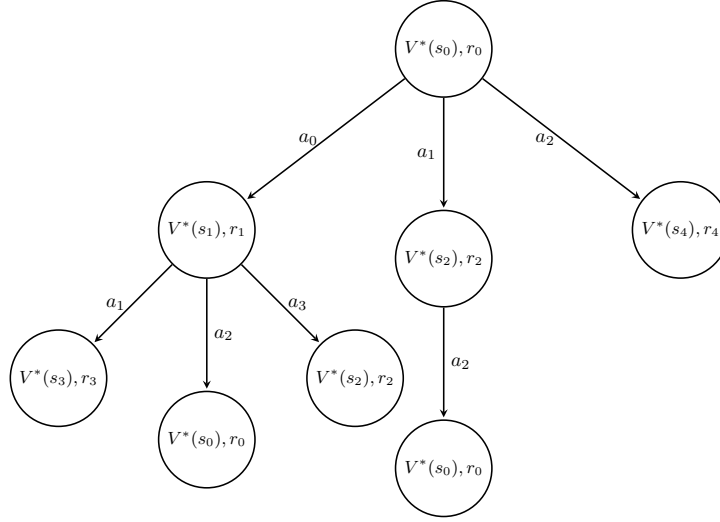


Figure 2: Tree of all paths in the graph in figure 1 with length less than or equal to 2

#### 4.1.3 Policy Gradient Methods

There is another branch of methods called *policy gradient methods* that uses gradient to adapt a model to a dynamic policy [8]. The accumulated reward independent of a state could be expressed as below

$$J(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) \quad (31)$$

$$= \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a) \quad (32)$$

$$= \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) \sum_{s'} P(s'|s, a) \nabla_\theta V^\pi(s') \quad (33)$$

Analytically calculating the gradient is tricky. Only policy is parameterized; however, the value terms include the policy as well. Following the derivation in Lilian Weng’s blog gives (please refer to the blog for full derivation).

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \pi_{\theta}(a|s) \quad (34)$$

$$\propto \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s) \quad (35)$$

$$= \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s, a) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} \quad (36)$$

$$= \mathbb{E}_{\pi} [Q^{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s)] \text{ because } (\ln x)' = 1/x \quad (37)$$

One policy gradient method (maybe the simplest) is *REINFORCE*, given as below.

1. Predict the reward  $G_t$
2. Perform a gradient update  $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_{\theta} \ln \pi_{\theta}(A_t|S_t)$  [33]

During the mean loss calculation, sample mean of the accumulated gradients are calculated. This gives an approximation for 37, in a Monte Carlo manner. As the predicted reward values may have high variance they lead to unstable optimization. For smoother gradient updates, the coefficient term is generally normalized with a *baseline* as  $Q(a, s) - b(s)$ . For a baseline, the mean  $V(s) = E_a[Q(a, s)]$  could be used. The book uses the optimal value  $V^*(s) = \max_a[Q(a, s)]$  from the SCST (*self-critical sequence training*) paper [8, 34]. It is possible to use a NN to approximate the rewards as well, which is done with methods such as *actor-critic*. The book’s implementation uses BLEU metric as the reward estimation [8].

## 4.2 LoRA (Low-Rank Adaptation of Large Language Models)

When training NNs, the parameters and gradients could be sparse. Their matrix representation could be rewritten as sum of products of matrices with smaller ranks as below.

$$W = \sum_i A_i B_i \quad (38)$$

Hu et al. choose to approximate the gradient of a layer with a single product. The product could be modeled as a forward pass of two smaller layers called *adapters*. In previous works, the adapters were injected between the original parameters. However, Hu et al. train the adapters alongside the parameters. This ensures the network depth stays the same, reducing the number of sequential operations allowing for faster inference. The expression for a composite linear-LoRA layer structure is given below.

$$h = (W + \Delta W)x = (W + \gamma_r AB)x \quad (39)$$

Instead of large number of network parameters, adapter parameters correspond to only a small portion of these parameters. The adapters could be removed and attached. In this regard, LoRA is a modular and lightweight approach. The authors also use a coefficient  $\gamma_r = \frac{\alpha}{r}$ . Tuning  $\alpha$  is not important as it corresponds to roughly the learning rate when optimizing with Adam [35]. The authors divide by the rank  $r$  as the dot product term becomes larger with larger ranks.

### 4.2.1 RS-LoRA (Rank-Stabilized LoRA)

RS-LoRA paper shows that for a network with  $m$  layers, after  $n$  gradient updates, the complexities for the expressions of forward pass output and gradients are  $\Theta_r((\gamma_r^2 r)^m)$  and  $\Theta_r(\gamma_r^2 r)$  respectively. The vanilla coefficient results in diminishing values (i.e.  $\Theta_r(\frac{1}{r^m})$  and  $\Theta_r(\frac{1}{r})$ ), leading to instability and worse performance. Instead, RSLoRA uses  $\gamma_r = \frac{\alpha}{\sqrt{r}}$  for stability (i.e.  $\Theta_r(1)$ ) [36].

### 4.2.2 Q-LoRA(Quantized LoRA)

Differentiation libraries (e.g. autograd of PyTorch) use intermediary variables such as the gradients for their calculations. We roughly need double the size of the model itself only to store gradients. Q-LoRA quantizes the stored parameters to 4-bit, as NF-4 (*NormalFloat*) type. During forward/backward pass, the parameters of the computed layer are dequantized to 16-bit, as bfloat (*BrainFloat*) type [37]. Bfloat maps to the same range with 32-bit representation with less precision as shown in figure 3 to prevent overflow errors. NF4 uses a lookup table of 16 *q-values* in the range [-1, 1], index of which is the stored 4-bit value. As long as the weights are normally distributed, NF4 equally distributes the *q-values* as illustrated in figure 4 [38].



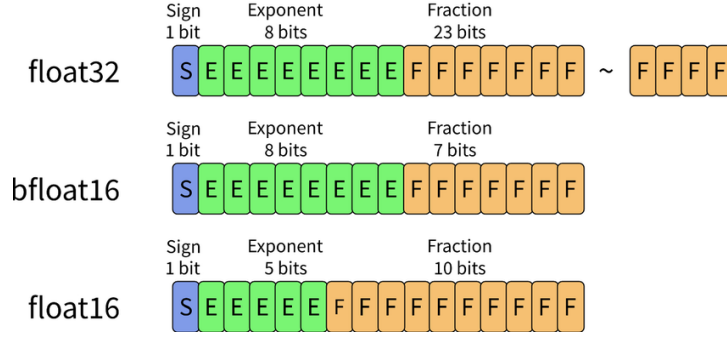


Figure 3: Depiction of three floating point representations in memory [39]

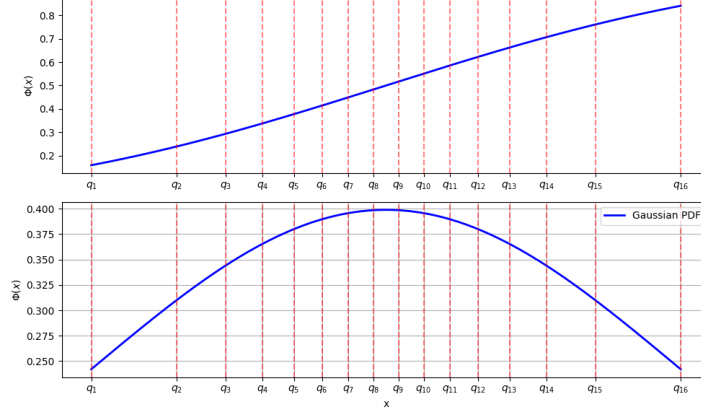


Figure 4: Gaussian PDF and CDF plot from top to bottom in the Interval  $[-1, 1]$  with  $q$  values

#### 4.2.3 LoftQ (LoRA-Fine-Tuning-aware Quantization)

LoRA initializes  $A$  with a normal distribution, and  $B = 0$ . LoftQ aims to find an initialization such that the inference error is minimized. It achieves it by using an iterative approach.

$$Wx = (Q + AB^T)x \quad (40)$$

$$(W - Q - AB^T)x = 0 \quad (41)$$

$$\|W - Q - AB^T\| = 0 \quad (42)$$

Where  $\|$  denotes the matrix norm we are trying to minimize [40].

### 4.3 Metrics

#### 4.3.1 Evaluation Metric: F1 score

For any task, one could form a confusion matrix by saving the actual and predicted values as in figure 5. A target class  $c$  could have set of *true positives*, *true negatives*, *false positives* and *false negatives* with cardinalities  $TP_c$ ,  $TN_c$ ,  $FP_c$  and  $FN_c$  respectively as shown in the figure. Note that we are interested in multiple classes.

		Predicted values				
		A	B	C	D	E
Actual values	A	3	7	2	9	5
	B	6	1	8	4	2
	C	9	5	3	7	6
	D	4	8	1	2	9
	E	7	3	6	5	4

Figure 5: A sample confusion matrix of 5 classes. The areas colored blue, green, red and yellow sum up to  $TP_B$ ,  $TN_B$ ,  $FP_B$  and  $FN_B$  respectively

*Precision* and *recall* are defined as

$$\text{precision}_c = \frac{TP_c}{TP_c + FP_c} = \frac{TP_c}{\text{number of predictions for } c} \quad (43)$$

$$\text{recall}_c = \frac{TP_c}{TP_c + FN_c} = \frac{TP_c}{\text{number of instances for } c} \quad (44)$$

$$(45)$$

*F1 score* incorporates both these values via harmonic sum.

$$\text{F1 score}_c = \frac{2}{\frac{1}{\text{precision}_c} + \frac{1}{\text{recall}_c}} \quad (46)$$

Macro and micro F1 scores are defined as follows [41].

$$\text{F1 score}_{\text{macro}} = \frac{1}{C} \sum_c \text{F1 score}_c \text{ where } C \text{ is the number of classes} \quad (47)$$

We define  $TP = \sum_c TP_c$  and similarly for other values as well.

$$\text{F1 score}_{\text{micro}} = \frac{2}{\frac{1}{\text{precision}_{\text{micro}}} + \frac{1}{\text{recall}_{\text{micro}}}} \quad (48)$$

$$= \frac{2}{\frac{TP+FP}{TP} + \frac{TP+FN}{TP}} \quad (49)$$

$$= \frac{2}{\frac{2TP+FN+FP}{TP}} \quad (50)$$

$$= \frac{2TP}{2TP+FN+FP} \quad (51)$$

$$= \frac{TP}{TP + \frac{1}{2}(FN+FP)} \quad (52)$$

$$= \frac{TP}{TP+FP} = \text{precision}_{\text{micro}} \quad (53)$$

$$= \frac{TP}{TP+FN} = \text{recall}_{\text{micro}} \text{ since } FN=FP \quad (54)$$

$$= \frac{TP}{\text{number of total instances}} \quad (55)$$

$$= \text{accuracy} \quad (56)$$

#### 4.3.2 Reward Metrics

**BLEU (Bilingual Evaluation Understudy)** is a method introduced for MT (machine translation) tasks to assess similarity of two sequences, namely the predicted *candidate* and the ground truth *reference*. Similar to convolution operation, a window capturing a subsequence of  $n$  tokens (i.e. *n-gram*) is slid over the sequence. Then the intersecting  $n$ -grams form the set of true positives. As in expression 43, dividing by total number of candidate  $n$ -grams gives the precision for  $n$ ,  $g_n$  which we call *n-gram score*. Intuitively, duplicate  $n$ -grams should not contribute to the score. To alleviate this, number of duplicate

matches are clipped with a value (usually 1). This is done only when calculating TP (not the total predictions) in order to penalize duplicate predictions. A candidate sequence could also be compared against multiple references as well (there could be multiple translations giving the same meaning). To give an aggregated similarity metric for multiple references, the total TP is used. Calculating similarity score of a candidate sequence (i.e. passage of text) composed of multiple subsequences (i.e. sentences) separated by a specific token (i.e. dot) is also straightforward. The total TP over the subsequences is divided by the total number of n-grams.

The n-gram scores could be weighted over different  $n$  values to give an overall n-gram score as below.

$$G_N = \prod_{n=1}^N g_n^{w_n} \quad (57)$$

$$\log G_N = \sum_{n=1}^N w_n \log g_n \quad (58)$$

We want the candidates to have lengths as close as possible to the references. N-gram score already penalizes the longer predictions. For shorter predictions, we use *brevity penalty* as below

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } c \leq r \end{cases} \quad (59)$$

where  $c$  refers to the total length of all candidates.  $r$  denotes the total reference length. For each candidate the closest reference length is chosen, and the chosen lengths are summed over the candidates, which gives *effective reference corpus length*  $r$ .

BLEU score is defined as

$$\text{BLEU}_N = BP \cdot G_N \quad (60)$$

$$\log \text{BLEU}_N = \min(1 - \frac{r}{c}, 0) + \sum_{n=1}^N w_n \log g_n \quad (61)$$

Usually  $N = 4$  is used as introduced in the original paper [42].

**ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** is an umbrella term for a set of approaches. ROUGE-N calculates an n-gram score between a candidate-reference pair similar to BLEU where  $N$  is an integer denoting the length of the n-gram. When aggregating n-gram scores over multiple references, it only chooses the maximum scoring reference instead of summing as BLEU does.

$$\text{ROUGE-N} = \arg_r \max g_n(c, r) \text{ where } (c, r) \text{ is a candidate-reference pair} \quad (62)$$

ROUGE-L uses LCS (*longest common subsequence*) length between two sentences as TP. Plain ROUGE-L only considers a single LCS. There is another variant ROUGE-L<sub>sum</sub> that takes the union of LCSs for each reference, summing up the union lengths to obtain TP.

ROUGE methods could use recall and F1 scores as the metric unlike BLEU, which only uses precision. ROUGE has other variants as well such as weighted ROUGE-L and ROUGE-S that uses strided n-grams [43]. These methods are not described here as they are not used in the task.

## 4.4 Implementation

### 4.4.1 Dataset

A modified version of the prompt in part 3 is used

List of predicates is ['org:founded', 'org:subsidiaries', 'per:date\_of\_birth', 'per:cause\_of\_death', 'per:age', 'per:stateorprovince\_of\_birth', 'per:countries\_of\_residence', 'per:country\_of\_birth', 'per:stateorprovince\_of\_death', 'org:website', 'per:cities\_of\_residence', 'per:parents', 'per:employee\_of', 'NA', 'per:city\_of\_birth', 'org:parents', 'org:political/religious\_affiliation', 'per:schools\_attended', 'per:country\_of\_death', 'per:children', 'org:top\_members/employees', 'per:date\_of\_death', 'org:members', 'org:alternate\_names', 'per:religion', 'org:member\_of', 'org:city\_of\_headquarters', 'per:origin', 'org:shareholders', 'per:charges', 'per:title', 'org:number\_of\_employees/members', 'org:dissolved', 'org:country\_of\_headquarters', 'per:alternate\_names', 'per:siblings', 'org:stateorprovince\_of\_headquarters', 'per:spouse', 'per:other\_family', 'per:city\_of\_death', 'per:stateorprovince\_of\_death', 'org:founded\_by']. What Subject-Predicate-Object triples are included in the following sentence?

List of predicates is included for more degrees of freedom. This allows giving different list of predicates for other graphs.

Preprocessing of the dataset is shown in part ?? . Firstly, the dataset is mapped so that there is only two columns (sentence and labels, sample 1). Then ChatML (*Chat Markup Language*, introduced by OpenAI) template is applied to the dataset (samples 2 and 3). The final step is the tokenization, which is not shown. When training, the triples are injected into the input sequence for reference. However, the model is expected to generate the data during evaluation. The triple is extracted as a separate column to compare the output at this stage. Because model is trained to give its answer after the generation prompt (<|assistant|>), it gives outputs irrelevant to the prompt without it at the end. This occurs during generation, which occurs during evaluation.

#### 4.4.2 Model and Training

Web APIs are not preferred as the task requires sending costly bulks of data. Therefore, closed-source GPT variants and alike are avoided. Google Colab is used with occasional switches to Kaggle Notebook. With these tools, access to a compute power to host a large LLM is not possible. Therefore, lightweight LLMs with fewer number of parameters are preferred. Microsoft’s Phi, Google’s Gemma, Facebook’s Llama are a few. For comparison, GPT-3 has about 175 billion parameters requiring 350GB [12] and is possibly not larger than GPT-4, whereas Phi-3-mini only has 3.8 billion parameters [29] (2GB memory requirement according to the output of a program written for the work). Phi-3-mini is chosen due to the reasons in part 3. Due to the memory concerns pointed out in part 4.2.2, lightweight models become extra important for training. PyTorch library enables one to disable unnecessary computations such as gradient calculations and achieve significant memory gain during evaluation. As a side note, the codebase was built around Phi-3-mini. Therefore it was not cost effective to have models such as Phi-4 and Deepseek-R1 work seamlessly. These models were released toward the end of the internship period.

HuggingFace’s `transformers` and `bitsandbytes` libraries for the model and Q-LoRA implementations are used respectively. Aside the parameters discussed in part 4.2.2, the configuration class also takes the maximum sequence length which is 4096 (the context window size). Flash attention implementation could not be enabled for performance gain. Flash attention supports Ampere, Ada, or Hopper GPUs [44]. Google Colab offers T4 which is Tesla; therefore, the plain implementation is preferred.

At the beginning, I attempted to use Trainer API of the `transformers` library. Due to encountered errors, a complete trainer API was written from scratch, which was a bad decision.

The training mostly occurs in the loss function. The model has little information before the training. As training progresses, the model becomes *wiser* (i.e. As it is introduced to more samples, it *knows* more). Therefore, the book’s implementation starts with teacher forcing (i.e. label is inputted as the next token), gradually transitioning to curriculum learning (i.e. predicted output is inputted as the next token). Transformers process the inputs in parallel, not allowing the use of curriculum learning. Teacher forcing is preferred during training [8].

The difference between a number (which is a HP) of sampled logits from the model’s output distribution and the maximum logit (as the baseline) gives the reward value. The book only considers BLEU as the reward metric [8]. BLEU and ROUGE as HPs (hyperparameters) are used for comparison. Default parameters for both are used. For ROUGE, F1 scores of ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-L<sub>sum</sub> are tuned. There are also other metrics such as METEOR that are not used. Due to the fact that TACRED gives a single reference without a separation token, the one-to-one single sequence variants of these metrics are used. There is also the *multiple* (another HP) variant that is built that compares a candidate against all references in the batch. CE (*Cross Entropy*) is used as usual for supervised learning.

The loss function supports *label smoothing*, which is a regularization technique that adds noise to the loss value. The returned loss is a linear interpolation between the original loss and the mean loss to make model decisions less certain for better generalization [45]. The interpolation constant *label smoothing factor* is another HP.

For the LoRA parameters, Tuning  $\alpha$  is unnecessary as mentioned in 4.2. Only the rank  $r$  is tuned. The value closest to exponential median is used (The range is  $r \in [1 = 2^0, 64 = 2^6]$  with median  $8 = 2^3$ , 32 is chosen). LoRA adapters are type of NNs that could use regularization techniques. Therefore, they support dropout regularization, which is another HP. Even though LoftQ is described in the report, it is very challenging to use it. It is an iterative LoRA initialization method with complexity proportional to the number of adapter parameters. Because NNs tend to be more powerful with higher number of parameters, LoRA adapters are injected to all linear layers. Therefore, it took considerable time to compute LoftQ initialization. For each different LoRA rank, the parameters had to be reinitialized. There are precomputed LoftQ adapters available on the internet for some LoRA configurations, which is not matching the specified configurations of the task. RS-LoRA is used as well.

Lastly, *AdamW* is used for the optimizer [46]. The learning rate, betas, and weight decay are tuned.

#### 4.4.3 Evaluation

The papers in part 3 were using F1 score. Micro F1 is used for evaluation. A unique triple is considered as a class. Instead of a CxC confusion matrix ( $O(C^2)$ ), a counter of classes ( $O(C)$ ) is constructed. The memory gain from this approach could be beneficial for larger KGs. For each prediction-label pair, the intersection gives true positives. The total number of true positives and the total length of both predictions and labels are used in the calculation.

There are multiple ways to generate an output in a sequential manner. Greedy search chooses the token with the maximum value. Beam search with a number of beams  $B$  expands for the top  $B$  scoring tokens at each level, decrementing  $B$  by 1. Deterministic techniques such as these could lead to repetitive outputs. The tokens could be sampled from the distribution. Alternatively, values below a certain probability threshold  $p$  could be eliminated [47]. A number of values in the API (do\_sample, num\_beams, top\_p) are tuned to compare between such techniques. Lastly, temperature works similarly to label smoothing factor, preventing the network from going too certain.

## 4.5 Hyperparameter Tuning

At the last month, implementing a HP tuning library was risky. Instead of repeating the same mistake with the trainer, Optuna is used. TPE (Tree-Structured Parzen Estimator), which is a *Bayesian optimization* algorithm to automatically tune HPs is used [48]. Due to the encountered errors, no results could be obtained.

## 5 Self-Reflection and Future Directions

Looking at the past, the trainer implementation costed significant amount of time. An external library could be worth the debugging effort. If an implementation is still necessary, libraries that simplify the task by handling the training loop such as *PyTorch Lightning* could be useful.

There is room for few improvements to be addressed. The given approach could be improved to include more advanced constraints. For instance, without specific implementation details, a number of ontological rules could be included in the dataset as well. Further comparison between the fine-tuned model and a model using a knowledge base (i.e. RAG) could be made. Instead of Phi-3, a more recent version (Phi-4 at the time) could be used and compared against alternatives such as Deepseek-R1.

## 6 Appendix

### 6.1 Manual Prompting Prompts

#### 6.1.1 ChatGPT

Prompt 1: List of predicates is ['org:founded', 'org:subsidiaries', 'per:date\_of\_birth', 'per:cause\_of\_death', 'per:age', 'per:stateorprovince\_of\_birth', 'per:countries\_of\_residence', 'per:country\_of\_birth', 'per:stateorprovinces\_of\_residence', 'org:website', 'per:cities\_of\_residence', 'per:parents', 'per:employee\_of', 'NA', 'per:city\_of\_birth', 'org:parents', 'org:political/religious\_affiliation', 'per:schools\_attended', 'per:country\_of\_death', 'per:children', 'org:top\_members/employees', 'per:date\_of\_death', 'org:members', 'org:alternate\_names', 'per:religion', 'org:member\_of', 'org:city\_of\_headquarters', 'per:origin', 'org:shareholders', 'per:charges', 'per:title', 'org:number\_of\_employees/members', 'org:dissolved', 'org:country\_of\_headquarters', 'per:alternate\_names', 'per:siblings', 'org:stateorprovince\_of\_headquarters', 'per:spouse', 'per:other\_family', 'per:city\_of\_death', 'per:stateorprovince\_of\_death', 'org:founded\_by']. What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). 'predicate' must be from the list of predicates only. Only use 'NA' when you cannot find any reasonable predicates. Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: The series leading to the may 1 derby will begin march 27 on usa network with the louisiana derby from new orleans, and will also include the lane 's end stakes from turfway park in florence, ky..

Triple: [('USA Network', 'NA', 'Turfway Park')]

Sentence: The bond insurers declined to comment on friday, though on thursday, mbia 's chief financial officer, charles e. chaplin, vigorously defended his company at a hearing in congress and said it did not need any help.

Triple: [('MBIA', 'org:top\_members/employees', 'Charles E. Chaplin')]

Sentence: A result is that for a variety of reasons, several prominent corporations that typically promote chief executives from within have turned to the outside to fill their top spot in recent years, including boeing, chrysler, con-agra, ford, hewlett- packard and 3m .

Triple:

Label: [('Con-Agra', 'NA', 'Chrysler')]

Prediction:

[('Boeing', 'NA', 'top spot'),  
( 'Chrysler', 'NA', 'top spot'),  
( 'Con-Agra', 'NA', 'top spot'),  
( 'Ford', 'NA', 'top spot'),  
( 'Hewlett-Packard', 'NA', 'top spot'),  
( '3M', 'NA', 'top spot')]

Prompt 2: List of predicates is ['org:founded', 'org:subsidiaries', 'per:date\_of\_birth', 'per:cause\_of\_death', 'per:age', 'per:stateorprovince\_of\_birth', 'per:countries\_of\_residence', 'per:country\_of\_birth', 'per:stateorprovinces\_of\_residence', 'org:website', 'per:cities\_of\_residence', 'per:parents', 'per:employee\_of', 'NA', 'per:city\_of\_birth', 'org:parents', 'org:political/religious\_affiliation', 'per:schools\_attended', 'per:country\_of\_death', 'per:children', 'org:top\_members/employees', 'per:date\_of\_death', 'org:members', 'org:alternate\_names', 'per:religion', 'org:member\_of', 'org:city\_of\_headquarters', 'per:origin', 'org:shareholders',

'per:charges', 'per:title', 'org:number\_of\_employees/members', 'org:dissolved', 'org:country\_of\_headquarters', 'per:alternate\_names', 'per:siblings', 'org:stateorprovince\_of\_headquarters', 'per:spouse', 'per:other\_family', 'per:city\_of\_death', 'per:stateorprovince\_of\_death', 'org:founded\_by']. What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). 'predicate' must be from the list of predicates only. Only use 'NA' when you cannot find any reasonable predicates. Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: Just earlier david gregory was interviewing cantor, and he talked about the virginia primary ballot and said " there 's no one else on the ballot with romney "

Triple: [('David Gregory', 'NA', 'Virginia')]

Sentence: The 23-year-old halim told the jakarta globe that she is a member of the church, for which she did humanitarian work after the asian tsunami in 2004, the mail and guardian said.

Triple:

Label: [('Jakarta Globe', 'NA', '2004')]

Prediction:

[('Halim', 'per:age', '23'),  
('Halim', 'org:member\_of', 'Church')]

Prompt 3: List of predicates is ['org:founded', 'org:subsidiaries', 'per:date\_of\_birth', 'per:cause\_of\_death', 'per:age', 'per:stateorprovince\_of\_birth', 'per:countries\_of\_residence', 'per:country\_of\_birth', 'per:stateorprovinces\_of\_residence', 'org:website', 'per:cities\_of\_residence', 'per:parents', 'per:employee\_of', 'NA', 'per:city\_of\_birth', 'org:parents', 'org:political/religious\_affiliation', 'per:schools\_attended', 'per:country\_of\_death', 'per:children', 'org:top\_members/employees', 'per:date\_of\_death', 'org:members', 'org:alternate\_names', 'per:religion', 'org:member\_of', 'org:city\_of\_headquarters', 'per:origin', 'org:shareholders', 'per:charges', 'per:title', 'org:number\_of\_employees/members', 'org:dissolved', 'org:country\_of\_headquarters', 'per:alternate\_names', 'per:siblings', 'org:stateorprovince\_of\_headquarters', 'per:spouse', 'per:other\_family', 'per:city\_of\_death', 'per:stateorprovince\_of\_death', 'org:founded\_by']. What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). 'predicate' must be from the list of predicates only. Only use 'NA' when you cannot find any reasonable predicates. Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: Directed by david gordon green; written by green, based on the novel by stewart o'nan; director of photography, tim orr; edited by william anderson; music by david wingo and jeff mcilwain; production designer, richard wright; produced by dan lindau, paul miller, lisa muskat and cami taylor; released by warner independent pictures.

Triple:

Label: [('William Anderson', 'NA', 'Paul Miller')]

Prediction:

[  
('david gordon green', 'per:employee\_of', 'NA'),  
('green', 'per:employee\_of', 'NA'),  
('stewart o\'nan', 'per:employee\_of', 'NA'),  
('tim orr', 'per:employee\_of', 'NA'),  
('william anderson', 'per:employee\_of', 'NA'),  
('david wingo', 'per:employee\_of', 'NA'),  
('jeff mcilwain', 'per:employee\_of', 'NA'),  
('richard wright', 'per:employee\_of', 'NA'),  
('dan lindau', 'per:employee\_of', 'NA'),  
('paul miller', 'per:employee\_of', 'NA'),  
('lisa muskat', 'per:employee\_of', 'NA'),  
('cami taylor', 'per:employee\_of', 'NA'),  
('warner independent pictures', 'org:alternate\_names', 'NA')  
]

Prompt 4: What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: The series leading to the may 1 derby will begin march 27 on usa network with the louisiana derby from new orleans, and will also include the lane 's end stakes from turfway park in florence, ky..

Triple: [('USA Network', 'NA', 'Turfway Park')]

Sentence: The bond insurers declined to comment on friday, though on thursday, mbia 's chief financial officer, charles e. chaplin, vigorously defended his company at a hearing in congress and said it did not need any help.

Triple: [('MBIA', 'org:top\_members/employees', 'Charles E. Chaplin')]

Sentence: A result is that for a variety of reasons, several prominent corporations that typically promote chief executives from within have turned to the outside to fill their top spot in recent years, including boeing, chrysler, con-agra, ford, hewlett- packard and 3m .

Triple:

Label: [('Con-Agra', 'NA', 'Chrysler')]

Prediction:

[('Boeing', 'org:promote', 'chief executives'),  
('Chrysler', 'org:promote', 'chief executives'),  
('Con-Agra', 'org:promote', 'chief executives'),

```
( 'Ford', 'org:promote', 'chief executives'),
( 'Hewlett-Packard', 'org:promote', 'chief executives'),
( '3M', 'org:promote', 'chief executives')]
```

Prompt 5: What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: Just earlier david gregory was interviewing cantor, and he talked about the virginia primary ballot and said " there 's no one else on the ballot with romney ".

Triple: [( 'David Gregory', 'NA', 'Virginia')]

Sentence: The 23-year-old halim told the jakarta globe that she is a member of the church, for which she did humanitarian work after the asian tsunami in 2004, the mail and guardian said.

Triple:

Label: [( 'Jakarta Globe', 'NA', '2004')]

Prediction:

```
[( 'Halim', 'told', 'The Jakarta Globe'),
( 'Halim', 'is a member of', 'the church'),
( 'Halim', 'did humanitarian work', 'after the Asian tsunami in 2004'),
( 'The Mail and Guardian', 'said', 'Halim did humanitarian work after the Asian tsunami in 2004')]
```

Prompt 6: What Subject-Predicate-Object triples are included in the task sentence? Each triple is in the form ('subject', 'predicate', 'object'). Some reference sentence-triple pairs will be given. For each pair, sentence is given after 'Sentence: ' and the corresponding triples are given in a python list after 'Triple: ' word. Only return the triples in a Python list. The task sentence is the last sentence with no triples.

Sentence: Directed by david gordon green; written by green, based on the novel by stewart o'nan; director of photography, tim orr; edited by william anderson; music by david wingo and jeff mcilwain; production designer, richard wright; produced by dan lindau, paul miller, lisa muskat and cami taylor; released by warner independent pictures.

Triple:

Label: [( 'William Anderson', 'NA', 'Paul Miller')]

Prediction:

```
[
( 'david gordon green', 'directed by', None),
( 'green', 'written by', None),
( 'green', 'based on', 'the novel by stewart o\'nan'),
( 'tim orr', 'director of photography', None),
( 'william anderson', 'edited by', None),
( 'david wingo', 'music by', None),
( 'jeff mcilwain', 'music by', None),
( 'richard wright', 'production designer', None),
( 'dan lindau', 'produced by', None),
( 'paul miller', 'produced by', None),
( 'lisa muskat', 'produced by', None),
( 'cami taylor', 'produced by', None),
( 'warner independent pictures', 'released by', None)
]
```

## 6.1.2 Phi-3-mini

### Constraint Predicates :

Prompt 7: [{ 'content': "List of predicates is ['org:founded', 'org:subsidiaries', "
" 'per:date\_of\_birth', 'per:cause\_of\_death', 'per:age', "
" 'per:stateorprovince\_of\_birth', 'per:countries\_of\_residence', "
" 'per:country\_of\_birth', 'per:stateorprovinces\_of\_residence', "
" 'org:website', 'per:cities\_of\_residence', 'per:parents', "
" 'per:employee\_of', 'NA', 'per:city\_of\_birth', 'org:parents', "
" 'org:political/religious\_affiliation', 'per:schools\_attended', "
" 'per:country\_of\_death', 'per:children', "
" 'org:top\_members/employees', 'per:date\_of\_death', 'org:members', "
" 'org:alternate\_names', 'per:religion', 'org:member\_of', "
" 'org:city\_of\_headquarters', 'per:origin', 'org:shareholders', "
" 'per:charges', 'per:title', 'org:number\_of\_employees/members', "
" 'org:dissolved', 'org:country\_of\_headquarters', "
" 'per:alternate\_names', 'per:siblings', "
" 'org:stateorprovince\_of\_headquarters', 'per:spouse', "
" 'per:other\_family', 'per:city\_of\_death', "
" 'per:stateorprovince\_of\_death', 'org:founded\_by']. What "
" Subject-Predicate-Object triples are included in the task "
" sentence? Each triple is in the form ('subject', 'predicate', "
" 'object'). 'predicate' must be from the list of predicates only. "

```

    "Only use 'NA' when you cannot find any reasonable predicates. "
    "Only return the triples in a Python list.",
    'role': 'system'},
{'content': " [('USA Network', 'NA', 'Turfway Park')]", 'role': 'user'},
{'content': 'The series leading to the may 1 derby will begin march 27 on usa '
'network with the louisiana derby from new orleans, and will also '
'include the lane 's end stakes from turfway park in florence, "
'ky.. ',
'role': 'assistant'},
{'content': " [('MBIA', 'org:top_members/employees', 'Charles E. Chaplin')]",
'role': 'user'},
{'content': 'The bond insurers declined to comment on friday, though on '
'thursday, mbia 's chief financial officer, charles e. chaplin, "
'vigorously defended his company at a hearing in congress and '
'said it did not need any help. ',
'role': 'assistant'},
{'content': 'A result is that for a variety of reasons, several prominent '
'corporations that typically promote chief executives from within '
'have turned to the outside to fill their top spot in recent '
'years, including boeing, chrysler, con-agra, ford, hewlett- '
'packard and 3m. ',
'role': 'assistant'}}]

```

Label: [('Con-Agra', 'NA', 'Chrysler')]

Prediction:

```

Prompt 8: [{'content': "List of predicates is ['org:founded', 'org:subsidiaries', "
" 'per:date_of_birth', 'per:cause_of_death', 'per:age', "
" 'per:stateorprovince_of_birth', 'per:countries_of_residence', "
" 'per:country_of_birth', 'per:stateorprovinces_of_residence', "
" 'org:website', 'per:cities_of_residence', 'per:parents', "
" 'per:employee_of', 'NA', 'per:city_of_birth', 'org:parents', "
" 'org:political/religious_affiliation', 'per:schools_attended', "
" 'per:country_of_death', 'per:children', "
" 'org:top_members/employees', 'per:date_of_death', 'org:members', "
" 'org:alternate_names', 'per:religion', 'org:member_of', "
" 'org:city_of_headquarters', 'per:origin', 'org:shareholders', "
" 'per:charges', 'per:title', 'org:number_of_employees/members', "
" 'org:dissolved', 'org:country_of_headquarters', "
" 'per:alternate_names', 'per:siblings', "
" 'org:stateorprovince_of_headquarters', 'per:spouse', "
" 'per:other_family', 'per:city_of_death', "
" 'per:stateorprovince_of_death', 'org:founded_by']. What "
'Subject-Predicate-Object triples are included in the task '
'sentence? Each triple is in the form ('subject', 'predicate', "
"'object'). 'predicate' must be from the list of predicates only. "
'Only use 'NA' when you cannot find any reasonable predicates. "
'Only return the triples in a Python list.',
'role': 'system'},
{'content': " [('David Gregory', 'NA', 'Virginia')]", 'role': 'user'},
{'content': 'Just earlier david gregory was interviewing cantor, and he '
'talked about the virginia primary ballot and said " there 's no "
'one else on the ballot with romney '. ',
'role': 'assistant'},
{'content': 'The 23-year-old halim told the jakarta globe that she is a '
'member of the church, for which she did humanitarian work after '
'the asian tsunami in 2004, the mail and guardian said. ',
'role': 'assistant'}}]

```

Label: [('Jakarta Globe', 'NA', '2004')]

Prediction:

Triples:

```

[('Halim', 'per:age', '23'),
('Halim', 'per:gender', 'NA'),
('Halim', 'per:nationality', 'Indonesian'),
('Halim', 'per:schools_attended', 'NA'),
('Halim', 'per:city_of_birth', 'NA'),
('Halim', 'per:city_of_death', 'NA'),
('Halim', 'per:stateorprovince_of_birth', 'NA'),
('Halim', 'per:stateorprovinces_of_residence', 'NA'),
('Halim', 'per:country_of_residence', 'NA'),

```



```
( 'Halim', 'per:country_of_death', 'NA'),
( 'Halim', 'per:countries_of_residence', 'NA'),
( 'Halim', 'per:cause_of_death', 'NA'),
( 'Halim', 'per:date_of_birth', 'NA'),
( 'Halim', 'per:date_of_death', 'NA'),
( 'Halim', 'per:employee_of', 'NA'),
( 'Halim', 'per:employer', 'NA'),
( 'Halim', 'per:education_level', 'NA'),
( 'Halim', 'per:ethnicity', 'NA'),
( 'Halim', 'per:family', 'NA'),
( 'Halim', 'per:gender_of_spouse', 'NA'),
( 'Halim', 'per:hometown', 'NA'),
( 'Halim', 'per:hometown_of_parents', 'NA'),
( 'Halim', 'per:hometown_of_spouse', 'NA'),
( 'Halim', 'per:income', 'NA'),
( 'Halim', 'per:income_of_family', 'NA'),
( 'Halim', 'per:income_
```

```
Prompt 9: [{ 'content': "List of predicates is ['org:founded', 'org:subsidiaries', "
    "'per:date_of_birth', 'per:cause_of_death', 'per:age', "
    "'per:stateorprovince_of_birth', 'per:countries_of_residence', "
    "'per:country_of_birth', 'per:stateorprovinces_of_residence', "
    "'org:website', 'per:cities_of_residence', 'per:parents', "
    "'per:employee_of', 'NA', 'per:city_of_birth', 'org:parents', "
    "'org:political/religious_affiliation', 'per:schools_attended', "
    "'per:country_of_death', 'per:children', "
    "'org:top_members/employees', 'per:date_of_death', 'org:members', "
    "'org:alternate_names', 'per:religion', 'org:member_of', "
    "'org:city_of_headquarters', 'per:origin', 'org:shareholders', "
    "'per:charges', 'per:title', 'org:number_of_employees/members', "
    "'org:dissolved', 'org:country_of_headquarters', "
    "'per:alternate_names', 'per:siblings', "
    "'org:stateorprovince_of_headquarters', 'per:spouse', "
    "'per:other_family', 'per:city_of_death', "
    "'per:stateorprovince_of_death', 'org:founded_by']. What "
    'Subject-Predicate-Object triples are included in the task '
    "sentence? Each triple is in the form ('subject', 'predicate', "
    "'object'). 'predicate' must be from the list of predicates only. "
    "Only use 'NA' when you cannot find any reasonable predicates. "
    'Only return the triples in a Python list.',
    'role': 'system'},
{ 'content': 'Directed by david gordon green; written by green, based on the '
    "novel by stewart o'nan; director of photography, tim orr; edited "
    'by william anderson; music by david wingo and jeff mcilwain; '
    'production designer, richard wright; produced by dan lindau, '
    'paul miller, lisa muskat and cami taylor; released by warner '
    'independent pictures. ',
    'role': 'assistant'}]
```

Label: [('William Anderson', 'NA', 'Paul Miller')]  
Prediction:

```
[('directed by', 'org:founded_by', 'david gordon green'),
('written by', 'org:founded_by', 'green, based on the novel by stewart o\'nan'),
('director of photography', 'org:founded_by', 'tim orr'),
('edited by', 'org:founded_by', 'william anderson'),
('music by', 'org:founded_by', 'david wingo and jeff mcilwain'),
('production designer', 'org:founded_by', 'richard wright'),
('produced by', 'org:members', 'dan lindau, paul miller, lisa muskat and cami taylor'),
('released by', 'org:members', 'warner independent pictures')]
```

## No Predicate Constraint :

```
Prompt 10: [{ 'content': 'What Subject-Predicate-Object triples are included in the task '
    "sentence? Each triple is in the form ('subject', 'predicate', "
    "'object'). Only return the triples in a Python list.",
    'role': 'system'},
{ 'content': " [('USA Network', 'NA', 'Turfway Park')]", 'role': 'user'},
{ 'content': 'The series leading to the may 1 derby will begin march 27 on usa '
    'network with the louisiana derby from new orleans, and will also '
    'include the lane 's end stakes from turfway park in florence, "
    'ky.. ',
    'role': 'assistant'},
{ 'content': " [('MBIA', 'org:top_members/employees', 'Charles E. Chaplin')]",
```

```

    'role': 'user'},
{'content': 'The bond insurers declined to comment on friday, though on '
            'thursday, mbia 's chief financial officer, charles e. chaplin, "
            'vigorously defended his company at a hearing in congress and '
            'said it did not need any help. ',
 'role': 'assistant'},
{'content': 'A result is that for a variety of reasons, several prominent '
            'corporations that typically promote chief executives from within '
            'have turned to the outside to fill their top spot in recent '
            'years, including boeing, chrysler, con-agra, ford, hewlett- '
            'packard and 3m. ',
 'role': 'assistant'}]

```

Label: [('Con-Agra', 'NA', 'Chrysler')]

Prediction:

```

Prompt 11: [{'content': 'What Subject-Predicate-Object triples are included in the task '
            'sentence? Each triple is in the form ('subject', 'predicate', "
            '"object'). Only return the triples in a Python list.",
 'role': 'system'},
{'content': " [('David Gregory', 'NA', 'Virginia')]", 'role': 'user'},
{'content': 'Just earlier david gregory was interviewing cantor, and he '
            'talked about the virginia primary ballot and said " there 's no "
            'one else on the ballot with romney '. ',
 'role': 'assistant'},
{'content': 'The 23-year-old halim told the jakarta globe that she is a '
            'member of the church, for which she did humanitarian work after '
            'the asian tsunami in 2004, the mail and guardian said. ',
 'role': 'assistant'}]

```

Label: [('Jakarta Globe', 'NA', '2004')]

Prediction:

```

[('Halim', 'is a member of', 'the church')]
[('Halim', 'did', 'humanitarian work after the asian tsunami in 2004')]

```

```

Prompt 12: [{'content': 'What Subject-Predicate-Object triples are included in the task '
            'sentence? Each triple is in the form ('subject', 'predicate', "
            '"object'). Only return the triples in a Python list.",
 'role': 'system'},
{'content': 'Directed by david gordon green; written by green, based on the '
            'novel by stewart o'nan; director of photography, tim orr; edited "
            'by william anderson; music by david wingo and jeff mcilwain; '
            'production designer, richard wright; produced by dan lindau, '
            'paul miller, lisa muskat and cami taylor; released by warner '
            'independent pictures. ',
 'role': 'assistant'}]

```

Label: [('William Anderson', 'NA', 'Paul Miller')]

Prediction:

```

[('David Gordon Green', 'Directed by', 'Directed by David Gordon Green'),
 ('David Gordon Green', 'Written by', 'Written by David Gordon Green'),
 ('David Gordon Green', 'Based on the novel by', 'Based on the novel by Stewart O'Nan'),
 ('Tim Orr', 'Director of Photography', 'Tim Orr'),
 ('William Anderson', 'Editor', 'William Anderson'),
 ('David Wingo', 'Music by', 'David Wingo'),
 ('Jeff McIlwain', 'Music by', 'Jeff McIlwain'),
 ('Richard Wright', 'Production Designer', 'Richard Wright'),
 ('Dan Lindau', 'Produced by', 'Dan Lindau'),
 ('Paul Miller', 'Produced by', 'Paul Miller'),
 ('Lisa Muskat', 'Produced by', 'Lisa Muskat'),
 ('Cami Taylor', 'Produced by', 'Cami Taylor'),
 ('Warner Independent Pictures', 'Released by', 'Warner Independent Pictures')]

```

## References

- [1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and

- A. Zimmermann, “Knowledge Graphs,” *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, May 2022, arXiv:2003.02320 [cs]. [Online]. Available: <http://arxiv.org/abs/2003.02320>
- [2] B. Karaca, “AI-Book,” May 2024. [Online]. Available: <https://github.com/Dj-Polyester/AI-Book>
- [3] S. Hochreiter and J. Schmidhuber, “Long Short Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://deeplearning.cs.cmu.edu/S23/document/readings/LSTM.pdf>
- [4] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” May 2016, arXiv:1409.0473 [cs]. [Online]. Available: <http://arxiv.org/abs/1409.0473>
- [5] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” Mar. 2018, arXiv:1802.05365 [cs]. [Online]. Available: <http://arxiv.org/abs/1802.05365>
- [6] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, “A Decomposable Attention Model for Natural Language Inference,” Sep. 2016, arXiv:1606.01933 [cs]. [Online]. Available: <http://arxiv.org/abs/1606.01933>
- [7] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, “Bidirectional Attention Flow for Machine Comprehension,” Jun. 2018, arXiv:1611.01603 [cs]. [Online]. Available: <http://arxiv.org/abs/1611.01603>
- [8] M. Lapan, Ed., *Deep reinforcement learning hands-on: apply modern RL methods to practical problems of chatbots, robotics, discrete optimization, web automation, and more, Second edition*, 2nd ed. Place of publication not identified: Packt Publishing, 2020.
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” Aug. 2023, arXiv:1706.03762 [cs]. [Online]. Available: <http://arxiv.org/abs/1706.03762>
- [10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 2019, arXiv:1810.04805 [cs]. [Online]. Available: <http://arxiv.org/abs/1810.04805>
- [11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” 2018.
- [12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language Models are Few-Shot Learners,” Jul. 2020, arXiv:2005.14165 [cs]. [Online]. Available: <http://arxiv.org/abs/2005.14165>
- [13] T. Schick and H. Schütze, “Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference,” Jan. 2021, arXiv:2001.07676 [cs]. [Online]. Available: <http://arxiv.org/abs/2001.07676>
- [14] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, “Unifying Large Language Models and Knowledge Graphs: A Roadmap,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, Jul. 2024, arXiv:2306.08302 [cs]. [Online]. Available: <http://arxiv.org/abs/2306.08302>
- [15] T. Shen, Y. Mao, P. He, G. Long, A. Trischler, and W. Chen, “Exploiting Structured Knowledge in Text via Graph-Guided Representation Learning,” Apr. 2020, arXiv:2004.14224 [cs]. [Online]. Available: <http://arxiv.org/abs/2004.14224>
- [16] D. Zhang, Z. Yuan, Y. Liu, F. Zhuang, H. Chen, and H. Xiong, “E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce,” Dec. 2021, arXiv:2009.02835 [cs]. [Online]. Available: <http://arxiv.org/abs/2009.02835>
- [17] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, “ERNIE: Enhanced Language Representation with Informative Entities,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 1441–1451. [Online]. Available: <https://www.aclweb.org/anthology/P19-1139>
- [18] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu, W. Liu, Z. Wu, W. Gong, J. Liang, Z. Shang, P. Sun, W. Liu, X. Ouyang, D. Yu, H. Tian, H. Wu, and H. Wang, “ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation,” Jul. 2021, arXiv:2107.02137 [cs]. [Online]. Available: <http://arxiv.org/abs/2107.02137>
- [19] J. Wang, W. Huang, Q. Shi, H. Wang, M. Qiu, X. Li, and M. Gao, “Knowledge Prompting in Pre-trained Language Model for Natural Language Understanding,” Oct. 2022, arXiv:2210.08536 [cs]. [Online]. Available: <http://arxiv.org/abs/2210.08536>

- [20] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, “REALM: Retrieval-Augmented Language Model Pre-Training,” Feb. 2020, arXiv:2002.08909 [cs]. [Online]. Available: <http://arxiv.org/abs/2002.08909>
- [21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Apr. 2021, arXiv:2005.11401 [cs]. [Online]. Available: <http://arxiv.org/abs/2005.11401>
- [22] D. Wilmot and F. Keller, “Memory and Knowledge Augmented Language Models for Inferring Salience in Long-Form Stories,” Sep. 2021, arXiv:2109.03754 [cs]. [Online]. Available: <http://arxiv.org/abs/2109.03754>
- [23] Y. Wu, Y. Zhao, B. Hu, P. Minervini, P. Stenetorp, and S. Riedel, “An Efficient Memory-Augmented Transformer for Knowledge-Intensive NLP Tasks,” Oct. 2022, arXiv:2210.16773 [cs]. [Online]. Available: <http://arxiv.org/abs/2210.16773>
- [24] S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu, and S. G. Tiddia, “Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction,” Jul. 2023, arXiv:2307.01128 [cs]. [Online]. Available: <http://arxiv.org/abs/2307.01128>
- [25] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen, and N. Zhang, “LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities,” Dec. 2024, arXiv:2305.13168 [cs]. [Online]. Available: <http://arxiv.org/abs/2305.13168>
- [26] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, “Position-aware Attention and Supervised Data Improve Slot Filling,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 35–45. [Online]. Available: <http://aclweb.org/anthology/D17-1004>
- [27] “TACRED Benchmark (Relation Extraction) | Papers With Code.” [Online]. Available: <https://paperswithcode.com/sota/relation-extraction-on-tacred>
- [28] G. Stoica, E. A. Platanios, and B. Póczos, “Re-TACRED: Addressing Shortcomings of the TACRED Dataset,” Apr. 2021, arXiv:2104.08398 [cs]. [Online]. Available: <http://arxiv.org/abs/2104.08398>
- [29] M. Abdin, J. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, H. Behl, A. Benhaim, M. Bilenko, J. Bjorck, S. Bubeck, M. Cai, Q. Cai, V. Chaudhary, D. Chen, D. Chen, W. Chen, Y.-C. Chen, Y.-L. Chen, H. Cheng, P. Chopra, X. Dai, M. Dixon, R. Eldan, V. Fragoso, J. Gao, M. Gao, M. Gao, A. Garg, A. D. Giorno, A. Goswami, S. Gunasekar, E. Haider, J. Hao, R. J. Hewett, W. Hu, J. Huynh, D. Iter, S. A. Jacobs, M. Javaheripi, X. Jin, N. Karampatziakis, P. Kauffmann, M. Khademi, D. Kim, Y. J. Kim, L. Kurilenko, J. R. Lee, Y. T. Lee, Y. Li, Y. Li, C. Liang, L. Liden, X. Lin, Z. Lin, C. Liu, L. Liu, M. Liu, W. Liu, X. Liu, C. Luo, P. Madan, A. Mahmoudzadeh, D. Majercak, M. Mazzola, C. C. T. Mendes, A. Mitra, H. Modi, A. Nguyen, B. Norick, B. Patra, D. Perez-Becker, T. Portet, R. Pryzant, H. Qin, M. Radmilac, L. Ren, G. d. Rosa, C. Rosset, S. Roy, O. Ruwase, O. Saarikivi, A. Saied, A. Salim, M. Santacroce, S. Shah, N. Shang, H. Sharma, Y. Shen, S. Shukla, X. Song, M. Tanaka, A. Tupini, P. Vaddamanu, C. Wang, G. Wang, L. Wang, S. Wang, X. Wang, Y. Wang, R. Ward, W. Wen, P. Witte, H. Wu, X. Wu, M. Wyatt, B. Xiao, C. Xu, J. Xu, W. Xu, J. Xue, S. Yadav, F. Yang, J. Yang, Y. Yang, Z. Yang, D. Yu, L. Yuan, C. Zhang, C. Zhang, J. Zhang, L. L. Zhang, Y. Zhang, Y. Zhang, Y. Zhang, and X. Zhou, “Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone,” Aug. 2024, arXiv:2404.14219 [cs]. [Online]. Available: <http://arxiv.org/abs/2404.14219>
- [30] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” Jan. 2025, arXiv:2501.12948 [cs]. [Online]. Available: <http://arxiv.org/abs/2501.12948>

- [31] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, “DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models,” Apr. 2024, arXiv:2402.03300 [cs]. [Online]. Available: <http://arxiv.org/abs/2402.03300>
- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [33] L. Weng, “Policy Gradient Algorithms,” Apr. 2018. [Online]. Available: <https://lilianweng.github.io/posts/2018-04-08-policy-gradient/>
- [34] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, “Self-critical Sequence Training for Image Captioning,” Nov. 2017, arXiv:1612.00563 [cs]. [Online]. Available: <http://arxiv.org/abs/1612.00563>
- [35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-Rank Adaptation of Large Language Models,” Oct. 2021, arXiv:2106.09685 [cs]. [Online]. Available: <http://arxiv.org/abs/2106.09685>
- [36] D. Kalajdzievski, “A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA,” Nov. 2023, arXiv:2312.03732 [cs]. [Online]. Available: <http://arxiv.org/abs/2312.03732>
- [37] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs,” May 2023, arXiv:2305.14314 [cs]. [Online]. Available: <http://arxiv.org/abs/2305.14314>
- [38] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, “8-bit Optimizers via Block-wise Quantization,” Jun. 2022, arXiv:2110.02861 [cs]. [Online]. Available: <http://arxiv.org/abs/2110.02861>
- [39] F. Grobelaar, “PygmalionAI/pygmalion-7b · Reasoning behind bfloat instead of float?” May 2023. [Online]. Available: <https://huggingface.co/PygmalionAI/pygmalion-7b/discussions/3>
- [40] Y. Li, Y. Yu, C. Liang, P. He, N. Karampatziakis, W. Chen, and T. Zhao, “LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models,” Nov. 2023, arXiv:2310.08659 [cs]. [Online]. Available: <http://arxiv.org/abs/2310.08659>
- [41] K. Leung, “Micro, Macro & Weighted Averages of F1 Score, Clearly Explained,” Jan. 2022. [Online]. Available: <https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f/>
- [42] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, US: Association for Computational Linguistics, 2002, pp. 311–318. [Online]. Available: <https://aclanthology.org/P02-1040.pdf>
- [43] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81. [Online]. Available: <https://aclanthology.org/W04-1013.pdf>
- [44] Dao-AILab, “GitHub - Dao-AILab/flash-attention at 27f501dbe011f4371bff938fe7e09311ab3002fa,” May 2022. [Online]. Available: <https://github.com/Dao-AILab/flash-attention/tree/27f501dbe011f4371bff938fe7e09311ab3002fa>
- [45] R. Müller, S. Kornblith, and G. Hinton, “When Does Label Smoothing Help?” Jun. 2020, arXiv:1906.02629 [cs]. [Online]. Available: <http://arxiv.org/abs/1906.02629>
- [46] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” Jan. 2019, arXiv:1711.05101 [cs]. [Online]. Available: <http://arxiv.org/abs/1711.05101>
- [47] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The Curious Case of Neural Text Degeneration,” Feb. 2020, arXiv:1904.09751 [cs]. [Online]. Available: <http://arxiv.org/abs/1904.09751>
- [48] S. Watanabe, “Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance,” May 2023, arXiv:2304.11127 [cs]. [Online]. Available: <http://arxiv.org/abs/2304.11127>