Instituto Tecnológico y de Estudios Superiores de Monterrey

Monterrey Campus

School of Engineering and Sciences

Master of Science in Computer Science

Thesis Proposal

Integrating Knowledge Graph Data with Large Language Models for Explainable Inference

by

Carlos Efrain Quintero Narvaez

ID A00825555

Monterrey, Nuevo León, May, 2023

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

The committee members, hereby, recommend that the proposal presented by Carlos Efrain Quintero Narvaez to be accepted to develop the thesis project as a partial requirement for the degree of **Master of Science in Computer Science**.

Dr. Raúl Monroy Borja
Tecnológico de Monterrey
Principal Advisor

Dr. Lectorio Cansado V.
First Committee Member's institution
Committee Member

Dr. Hostígono Final P.
Second Committee Member's institution
Committee Member

Dr. José Carlos Ortiz Bayliss
Director of Program in Computer Science
School of Engineering and Sciences

Monterrey, Nuevo León, May, 2023

Contents

Abstract

Recent advancements in **Large Language Models (LLMs)** such as **OpenAI GPT, BERT, and LLaMA** have demonstrated their potential for complex reasoning tasks with natural language. However, there is still room for improvement as the inner workings of these models are not yet fully understood. In this line of thought, **Neurosymbolic Artificial Intelligence**, which combines **Symbolic Logic Reasoning** and **Deep Learning**, aims to create **explainable** inference models using the virtues of the two fields. **Knowledge Graphs (KGs)** are an essential component in this subject, since they provide concise representations of large knowledge bases, understandable for both users and models. Two significant challenges in this area are **query answering** from KGs, and the **integration of KG information** into the output of language models. Researchers have proposed a variety of approaches to address these last two, including the use of Deep Learning for complex queries on KGs and **Augmented Language Models** that integrate recognition of entities from a KG. In this thesis, we propose to modify and combine these approaches with recent LLM developments, creating an explainable way for LLMs to work with **data from any KG**. Our approach will use LLMs for the KG entity embedding steps utilized in existing techniques, while keeping the other parts of the methods intact. Furthermore, we will use this new querying method for executing KG queries in the internal inference process of LLMs. Our goal is to reduce the frequency of hallucinations and attempt to make it easier to detect them by allowing the model to provide informed explanations, making them more broadly useful for general contexts.

1 Introduction

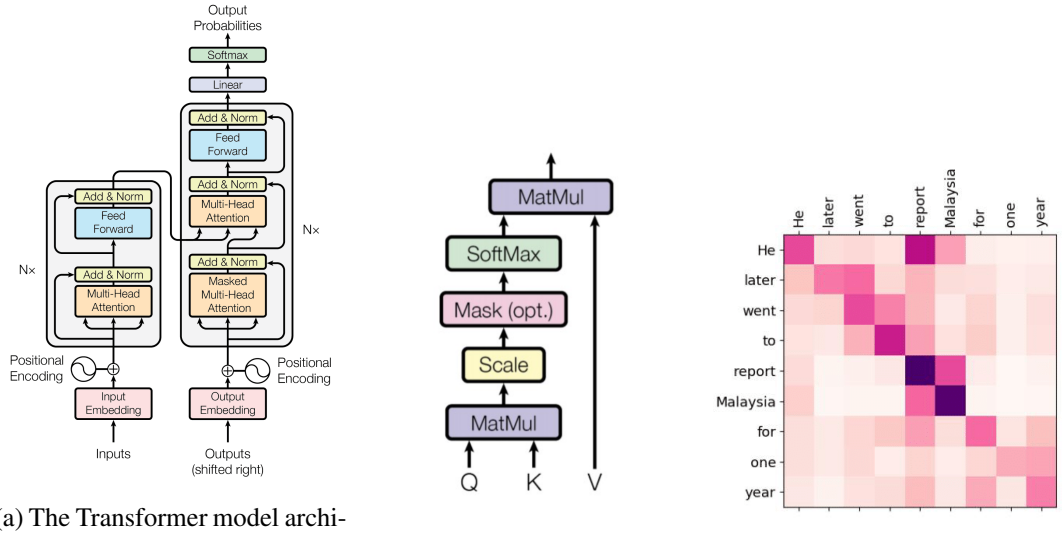
Impressive capabilities and even sparks of Artificial General Intelligence (AGI) [1] have emerged from the development of new **Large Language Models (PLMs)** such as **OpenAI GPT** [2, 3, 4], **BERT** [5], and **LLaMA** [6]. This seems to motivate continuation of work on this same line, dedicating more and more resources to training and execution of these types of models. However, it may be worth questioning whether these approaches that require massive corpuses of data and ever-increasing computational resources are the best methods to follow. Moreover, the inner workings of these models are not yet fully understood. This makes their use for critical fields such as medicine and data analysis, where one may need accountability and being able to correct mistakes, not yet possible.

In this introduction, we will first provide some background information on what LLMs are, how they are trained and used, and state more details and facts about the challenges and opportunities we have mentioned. We will also elaborate on the field of Neurosymbolic AI, specifically on some methods used to reason on Knowledge Graphs [7, 8]. After that, we will go over some examples of augmented language models, ERNIE and KEPLER [9, 10, 11]. Finally, we will outline the main objectives of this thesis proposal, and how they aim to address some of the open questions and problems related to LLMs. We begin by defining what LLMs are and how they work.

1.1 Large Language Models

Large Language Models are Deep Learning models that have been designed to work using the so-called **Transformer** architecture (Figure 1a) [12]. This is made of two components that work similarly, the so-called **encoders** and **decoders**. Both of these rely on **Self-Attention Blocks** (Figure 1b) which allow the models to rate the connections between parts of a sentence and evaluate from there, as seen in (Figure 1c). Parts of this architecture are later redefined by each language model, but in principle all of them consist of a number of encoder and decoder blocks as the ones seen in Figure 1a.

Given these architecture patterns, we are in a better position to talk about the size of notable LLMs. The first release of OpenAI GPT used 12 Transformer encoder blocks stacked on top of each



(a) The Transformer model architecture, with N encoders and N decoders.

(b) Scaled Dot-Product Attention Block.

(c) Self-attention matrix heatmap.

Figure 1: Architecture components of popular PLMs. First proposed in [12].

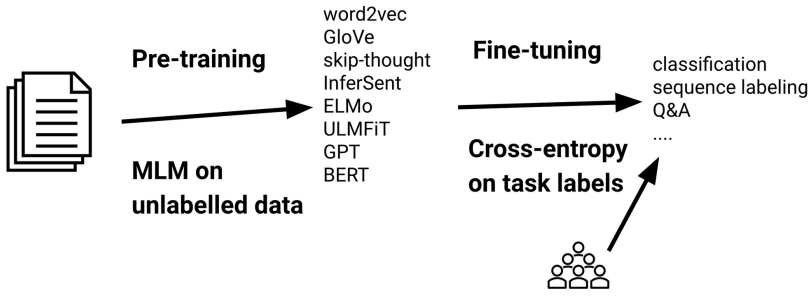


Figure 2: Standard pre-training and fine-tuning setting for PLMs [13].

other, having a total of 117 million parameters [2]. Similarly, the largest version of BERT has 345 million parameters. GPT-2 had four versions, with 12, 24, 36 and 48 blocks each, the largest of which had 1,542 million parameters [3]. The largest version of GPT-3 had 96 decoder blocks and 175 billion parameters [4]. Subsequently, the number of parameters in GPT-4, the latest version of OpenAI GPT, is not known, but it is reported to be about one trillion [14]. Furthermore, the recent release of the open-source LLaMA model has four versions with 7 billion, 13 billion, 33 billion, and 65 billion parameters [6]. Thus, the current tendency is to train ever-growing models to get better performance.

Now, let us examine the training processes for the before mentioned models, which are as important to their results as their architecture. As shown in Figure 2, training is done in two stages [2]. The first stage of training is done on a **large corpus of text** to obtain a high-capacity language model. The objective in this training stage is to **predict the next part of a sentence** or text given its predecessors. This is what is known as the **pre-training**. After obtaining this model that predicts how a piece of text will continue, the second stage of training is executed and the model uses **labeled data** to learn to do a discriminative task on its own output. That is, given example data on how to **rank its own outputs**, the objective is to predict that same ranking and use that to choose the best way to continue its inference.

Along with the development of these language generation models, their limitations and pitfalls come into question. Early work discovered that the usual maximum-likelihood metrics used in training could result in **degeneration**, that is, text that is bland, incoherent or gets stuck in incoherent loops [15, 16]. Indeed, it is a fact that LLMs often generate text that may be nonsensical or unfaithful to the provided input [17]. This behavior is referred to as **hallucination** [18]. Hallucinated text gives the impression of being fluent and based on factual information, despite being quite the contrary. It appears to be based on the real context provided, but the existence of such contexts is not immediately obvious. This in particular makes this behavior so dangerous. For example, in a question answering context, one may ask a specialized niche question and the model may answer with equally specialized terminology and even with the same writing structure as in relevant literature, but still give an incorrect answer without the user realizing.

The problem of models being able to disguise hallucinated answers as coherent ones arises from the black-box nature of the Deep Learning framework. Although it is possible to make evaluations on the inner layers of inference for smaller models, for more recent models with billions of parameters this becomes infeasible. This issue could be addressed by doing an effort to make models **explainable**. For this, a combination of the methods of **Deep Learning** with those of **Symbolic Logic Reasoning**, as is done in the emerging field of **Neurosymbolic AI**, may be an answer.

1.2 Neurosymbolic AI and Knowledge Graphs

Neurosymbolic AI is an area of research that is gaining more attention lately, as it seeks to combine Deep Learning with **Symbolic Logic Reasoning** methods to create models that are both highly predictive and somewhat comprehensible to humans [7]. With **Knowledge Graphs (KGs)** becoming more popular for representing complex data with multiple relationships, researchers have been exploring ways to reason on graph structures in a neurosymbolic fashion. Traditionally, this has involved either rule-based inference or generation of numerical embeddings for **Machine Learning** methods. However, recent studies have attempted to bridge this gap between rule-based and embedding-based methods in ways that preserve performance, enhance interpretability, and integrate expert knowledge.

Let us some background on what Knowledge Graphs are and how they are defined. A graph $G = (E, V)$ is formed by a set of vertices or nodes, V, and a set of ordered pairs of V called edges, $E \subseteq V \times V$. Each edge $e = (u, v) \in E$ connects two adjacent or neighboring vertices. The term

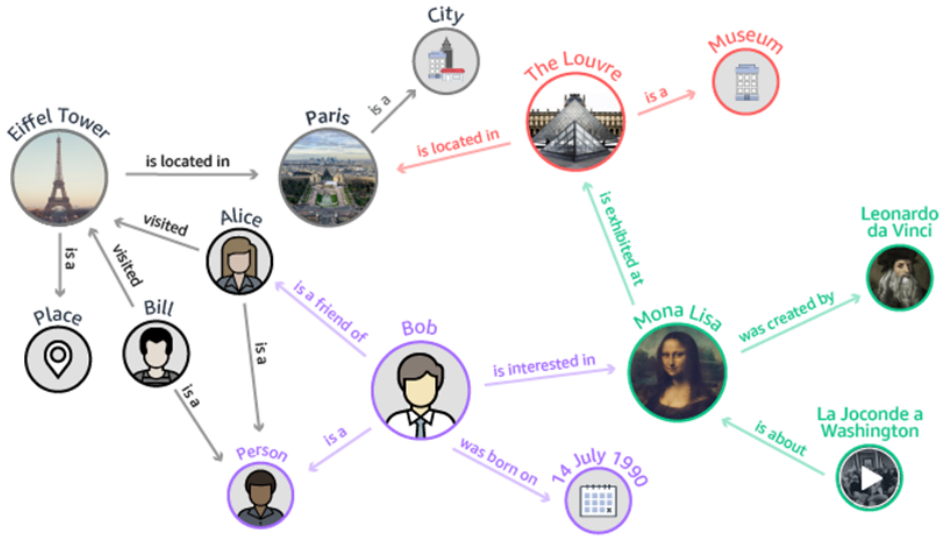


Figure 3: Knowledge Graphs capture relations between different types of entities in a domain [19, Figure 1].

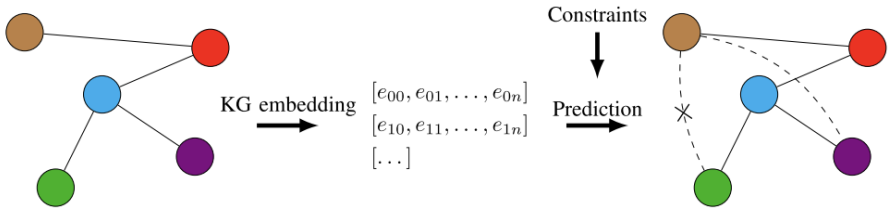


Figure 4: Embedding of entities and relations in a KG into real vectors of a space $\mathbb{R}^d$ for prediction of new relations [7, Figure 5].

triple denotes a pair of vertices connected by an edge. In particular, a **Knowledge Graph (KG)** is used to represent a knowledge base for a specific domain by representing objects or entities as vertices and the relations between them as edges with labels, as seen in Figure 3.

Now, we will go over some methods used to work with data from these graphs. KGs store human knowledge about a domain, and thus they are often incomplete. They can even contain false positives based on incorrect data. For this reason, **KG Completion (KGC)** methods have been proposed, designed to refine the graphs and uncover unseen information by adding new edges. To achieve this, algorithms for representation learning on graphs are used, which share a common goal of finding **node embeddings** as real vectors in a low dimensional space $\mathbb{R}^d$ such that similar nodes in a graph have close embeddings [7]. These embeddings are then inputted into distinct Machine Learning and Deep Learning models to process new information.

One of the problems for which the methods mentioned before are utilized is **Complex Query Answering** on KGs. Specifically, we focus on conjunctive queries, which are queries that use exis-

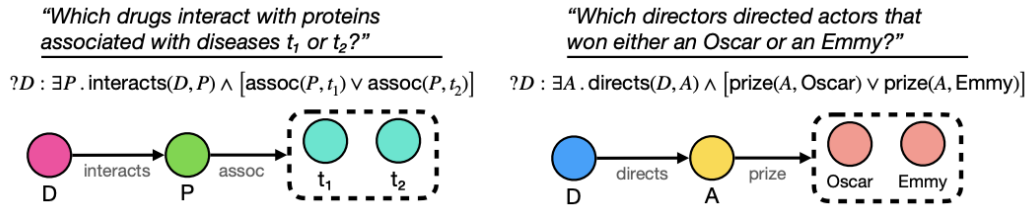


Figure 5: Concrete examples of conjunctive queries [8, Figure 1].

tential ($\exists$) and conjunctive ($\wedge$) operations in the form

$$\begin{aligned} \mathcal{Q}[A] \triangleq\ & ?A : \exists v_1, \ldots, v_m \in V \mid e_1 \wedge \ldots \wedge e_n \\ & \text{where } e_i = r(c, v_j),\ \ r \in E,\ c \in V \\ & \text{or } e_i = r(c, A),\ \ r \in E \\ & \text{or } e_i = r(v_k, v_j),\ \ r \in E, \end{aligned}$$

where we call each e_i an *atom* and $r(u, v)$ means that nodes u and v are connected by an edge labeled with relation r. The goal of answering this query is obtaining a set of entities $\mathcal{Q}[A]$ for which a set of conditions are held true. Concrete examples of this kind of query are shown in Figure 5.

In [8] a query answering model called **Continuous Query Decomposition (CQD)** is shown, relying on doing a **continuous reformulations** of queries to solve them. This means replacing the entities e_i of the graph with some real vector space embeddings $\mathbf{e_i}$ and the conjunctions and disjunctions with t-norms $\top : [0,1] \times [0,1] \longrightarrow [0,1]$, which are generalizations of logic operators for fuzzy logic. Thus, the conditions in conjunctive queries would yield a value in the range $[0,1]$ instead of true or false. The answer set for the query would then be the highest rated entities. A crucial step in this method is the generation of the embeddings for the entities, which in this case is done using a Deep Learning model. This could then be done using the same embedding steps done for LLMs, even reusing the same parameters.

1.3 Constrained Training with Temporal Logic

Going back to the need for **explainability** in Deep Learning models, we come across another Neurosymbolic method to make this possible. This is the customization of loss functions for different tasks using elements of logical reasoning. This could be using fuzzy logic as we have mentioned before or doing other types of continuous evaluations based on logic-based approaches. Thereby, **temporal logic** becomes relevant in the context of DL models that have an inherent succession structure in their outputs, such as models that determine a series of steps for a task. Indeed, developing specific loss functions in this way using the properties of temporal logic is possible, as is shown in [20].

Let us go over the functioning of the logic framework behind the customization of these loss functions. Temporal logic introduces new language components and symbology useful in the context of statements that are assigned truth values depending on the "time" at which they are evaluated. Some of these new components are:

- $\Box\rho$. ρ is *always* true.
- $\Diamond\rho$. ρ will *eventually* be true.
- $\mathcal{N}\rho$. ρ is true at the *next* time instant.

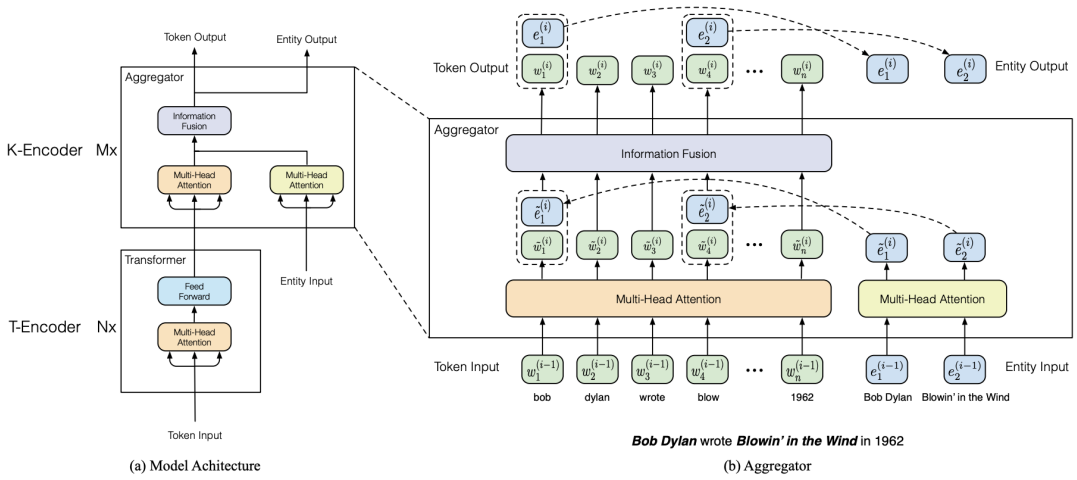


Figure 6: ERNIE model architecture [10, Figure 2].

- $\rho_1 \mathcal{U} \rho_2$. ρ_1 is true *until* ρ_2 is true.
- $\rho_1 \mathcal{R} \rho_2$. ρ_1 is true *since* ρ_2 is true.

These new formulae gain importance when training models with predictions with the form of succession events, as there arises a need for a framework to make statements with a dependence on past and future events. Thus, guiding the model using soft rules as those provided by a loss function turns more straightforward by using this method. This will be important for the work in this thesis, as we will need to gently guide the order in which our model accesses information to generate its answers.

1.4 Augmented Language Models

An emerging trend in research is the combination of LLMs, such as BERT and GPT, with structured knowledge contained in Knowledge Graphs (KGs), resulting in what is known as **Augmented Language Models** [9]. Since LLMs obtain their knowledge from vast amounts of unstructured training data, the integration of structured knowledge from KGs is worth considering. This infusion can occur at various stages of the language model, including input, architecture, output, or some combination of them. Some examples of notable augmented language models are **ERNIE** and **KEPLER**, which we will discuss further.

ERNIE is a language model that adds a preprocessing step where it recognizes informative entities from a KG on its natural language input and later enhances language representation with this knowledge. The model can then take full advantage of lexical, syntactic, and knowledge information. This is done by integrating embeddings of entities in the KG into the encoders of the model, as seen in Figure 6.

KEPLER is another language model that integrates KG data. This is done by training the same encoder in two different settings and combining the training losses from both so that the model balances the influences from the two processes. These two settings are the familiar KG Embedding and Language Generation tasks. What is done for KG Embedding is of particular interest here, as it uses descriptions of entities and relations instead of just the geometric properties of the graph as other models do. The general architecture of the model can be seen Figure 7.

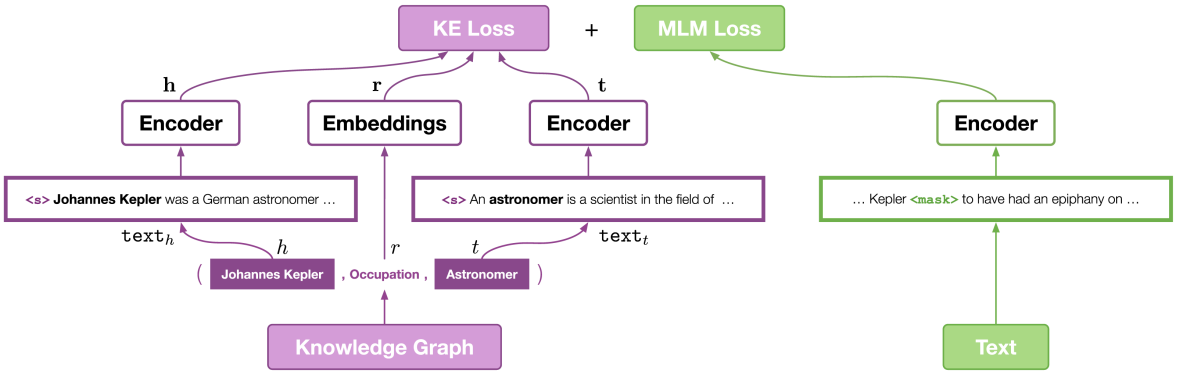


Figure 7: KEPLER model architecture [11, Figure 2].

Source	Hallucinated Output
The first vaccine for Ebola was approved by the FDA in 2019 in the US, five years after the initial outbreak in 2014. To produce the vaccine, scientists had to sequence the DNA Extrinsic of Ebola, then identify possible vaccines, and finally show successful clinical trials. Scientists say a vaccine for COVID- 19 is unlikely to be ready this year, although clinical trials have already started.	The first Ebola vaccine was approved in 2021. China has already started clinical trials of the COVID-19 vaccine.

Figure 8: Examples of two hallucinations from the same output, see that the source text has no mention of the year 2021 or China. Examples obtained from [17].

2 Problem Definition

In this thesis, we will aim to address the limitations and pitfalls of language models in generating coherent and faithful text. As mentioned before, while LLMs have shown great success in generating human-like text, they also exhibit a tendency to generate nonsensical or unfaithful text, which is referred to as hallucination, as mentioned in **Section 1.1**. This behavior poses a significant challenge in various applications, such as question-answering systems, where the model may generate an incorrect answer that appears to be based on the real context provided, as seen in the examples in Figure 8. The black-box nature of deep learning models exacerbates this problem, making it difficult to identify when the model is generating hallucinated text. For these reasons, it would be ideal to have a language model capable of explaining its output, offering a view of its internal decision processes.

Previous work [2] has shown that the use of Self-Attention Blocks in Transformer based LLMs gives important pieces of information when evaluating the output. This is because one can "dissect" the execution of the model and obtain a matrix as the one shown in Figure 1c. However, even though these matrix views can be helpful, their evaluation and visualization can quickly get out-of-hand as the number of tokens or words used by a model grows. Even when the size is not too large, it has been shown that for more complex tasks, such as those related to mathematical proofs and statements, self-attention can be quite superficial and inadequate, as happens in [21].

Some other attempts at improving the situation with explainability of language model outputs

have been DeepMind GopherCite [22], OpenAI WebGPT [23], and Google LaMDA [24]. These models are allowed to execute searches on external data and use it for their answers, displaying the actual data that was searched for and used. They have succeeded in allowing the user to make more accurate judgements on the validity of the generated text. However, it is worth noting that still does not solve the issue with the models sometimes hallucinating even with the given external information. Furthermore, they also fail to solve the problem of models failing to do more complex reasoning on different matters.

One of the reasons for why hallucinations are such a big problem in language generation is mentioned in [17, Section 4]. Indeed, it has been shown that state-of-the-art models, when assigned the task of summarizing data from a given source and evaluated with popular metrics such as ROUGE, BLEU, and METEOR, have hallucinated content in 25% of their generations [25]. Furthermore, it is mentioned in [26] that even if a summary contains a large amount of hallucinatory content, it can still have a high ROUGE score. These alarming facts motivate working on having explainable models on which outputs can be scored accurately with the actual source information.

3 Solution Proposal

As we delve into the discussion on LLMs, as mentioned in **Section 1.1**, we find that when faced with more complex questions or tasks, the issue of text degeneration and hallucinations becomes more prominent. Furthermore, detecting these hallucinations becomes harder with the increasing model complexity. For these cases, having the models provide explanations of their inference process and where they obtained their information may help reduce the frequency with which these hallucinations happen. Furthermore, these explanations may also help detect them. Thus, we propose a two-step solution to tackle this problem.

In the first step, we suggest integrating Neurosymbolic AI methods for query answering with a modified version of the KEPLER KG embedding model. For the second step, we will design a new model that can make KG queries on its internal inference, integrating the developments of the first step. This approach will allow for the integration of a KG into the new language model and enable it to explain its reasoning from that data.

To achieve the first step involving neurosymbolic querying, we propose using a slightly modified version of the KEPLER model presented in [8] and described at the end of **Section 1.2**. This modification will replace the original encoder, trained specifically for use with KEPLER [11], with an LLM such as LLaMA, BERT, or OpenAI GPT. This replacement is illustrated in Figure 9. While the original KEPLER itself could be used for this purpose, it is still open to discussion whether a more broadly trained version based on other models would be more effective.

Once the KG embeddings are generated, the overall model will integrate them into its internal inference, similarly to that of the ERNIE model mentioned in **Section 1.4**. Now, it is important to note that ERNIE only has one layer where entity embeddings are inputted into the system. Therefore, we propose adding this last feature into more layers so that the model can integrate queries at different stages of the inference process. This can be done by adding an "appendix" into the complete language model, as seen in Figure 10. Then, by logging these queries, the model should be able to output a meaningful explanation of its generated text.

Let us delve into how this query answering appendix we just mentioned will work. It will be designed to answer queries using the **CQD** method we mentioned before. This query answering model, however, will have the slight modification that the continuous reformulation will replace entities not with the specialized embeddings utilized in the original implementation, but with those generated

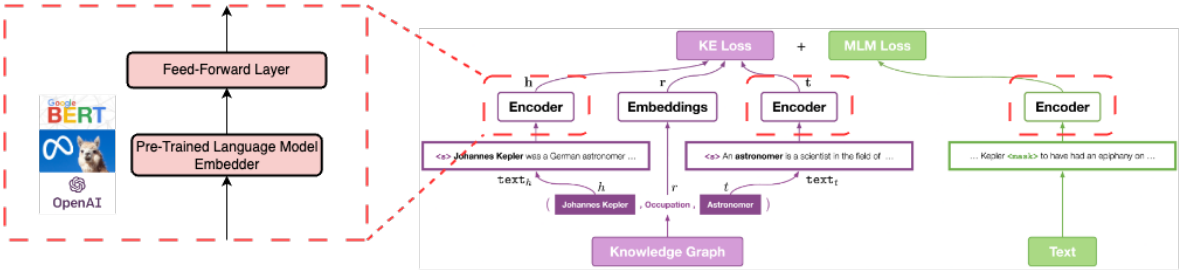


Figure 9: Proposed modification of the KEPLER KG embedder to integrate pre-trained LLMs. The encoders enclosed in red boxes are all copies of the same model with the same parameters.

by our version of KEPLER we mentioned earlier. This should allow the queries to be answered not only using geometric information from the connections in the Knowledge Graph, but also lexical and "common-sense" context of the entities, as KEPLER uses the descriptions of the entities rather than just their relations.

Furthermore, the distinct querying appendices in each block must be trained in such a way that their queries support each other, i.e. the central modified ERNIE part must be able to interconnect the information from the queries. Note that this part is designed to make a series of queries in linear order depending on information from the ones done before, giving it a notion of temporal logic. Thus, it would be helpful if the training was done in a way that reflects this successive nature. For this, we propose using a training method such as the one we mentioned in **Section 1.3** and shown in [20], which uses a series of temporal logical constraints defined formally using the Isabelle Higher-Order Logic (HOL) language to define an appropriate loss function that represents relevant constraints on the model. This constrained training method should allow the querying modules of the model to have more guidance on what information they should be looking for. Indeed, the constraints that can be translated into a loss function using this method are of the form:

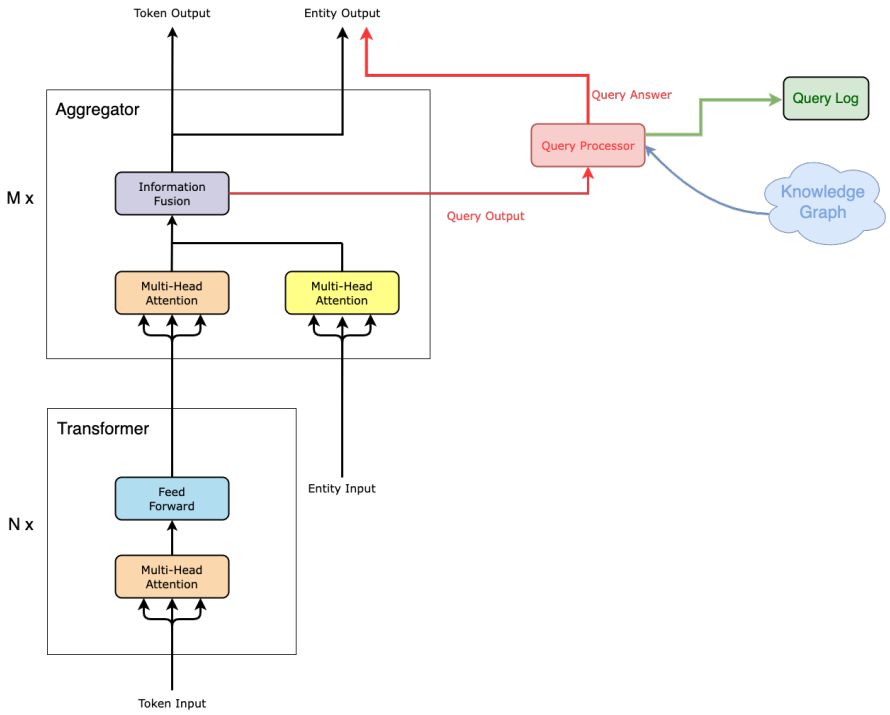
- "property X is always true in this series"
- "property Y is eventually true in this series"
- "property Z is true only after W happens in this series".

Which can be quite useful in our context as we need to have constraints such as:

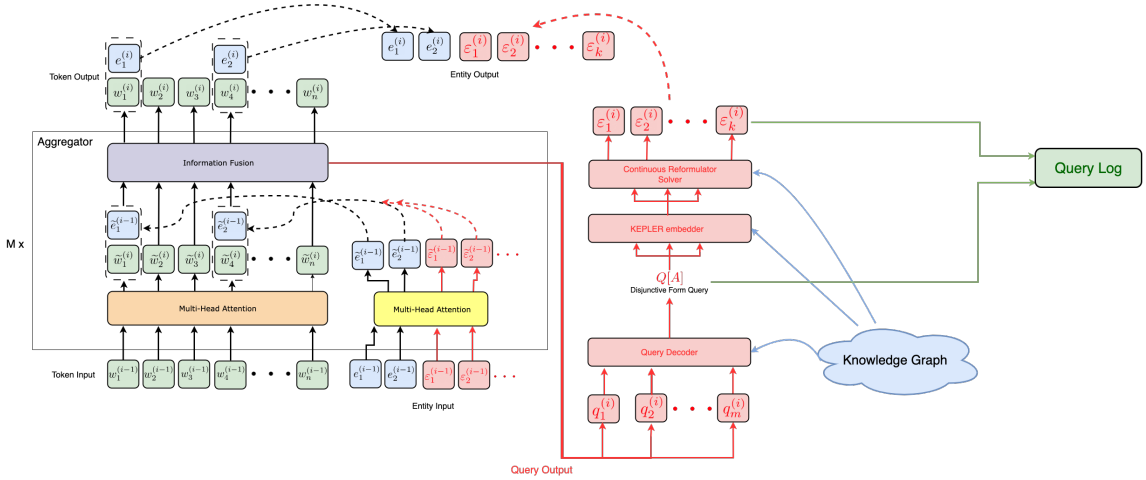
- "all queries should be on entities of type X"
- "the queries should eventually reach an entity related to type Y"
- "queries should only be related to type Z after it has been mentioned by a past query"

This kind of constrained learning should be of much help in our context, as it would allow for training to converge much faster and at the same time provide a much more **explainable** version of the model.

In summary, we propose building an Augmented Language Model with the ability to make multiple queries to a Knowledge Graph and provide an explanation of its text generation with those queries. This with the purpose of reducing the frequency with which hallucinations appear in text generation and at the same time making them more visible through the generated explanation.



(a) Modified ERNIE architecture with querying appendix in the Aggregator block.



(b) In-depth look at the architecture of the modified ERNIE Aggregator block with the querying appendix.

Figure 10: Modified ERNIE architecture with querying appendix, an overall look and a detailed look are shown.

4 Hypothesis and Research Questions

Our hypothesis is that it is possible to reduce the frequency with which hallucinations occur in the results of a language model if we provide it with the ability to query information from a Knowledge Graph also having it generate an explanation of its results through those queries. To accomplish this, our solution involves developing a query answering system utilizing Neurosymbolic AI methods, specifically a modified version of the CQD model [8]. This query answering model in turn will be integrated into the architecture shown in Figure 10a. The research questions which we intend to answer with this work are then:

- Does KG query answering improve by adding description-aware embedders based on an LLM?
- Which LLM is more effective for query answering?
- Does performance improve by using larger LLMs?
- Are hallucinations reduced by allowing the model to make queries on its inference?
- Is model training performance improved by using temporal logic constraints for loss functions?
- Are the explanations outputted by the queries done on the model useful for identifying hallucinations?
- How much is inference time impacted by the additional model overhead?

5 Objectives

The general objective in this thesis will be to develop a language model that can query a KG on its internal inference process and output the generated queries as an explanation of its reasoning. In turn, query answering should be supported by use of available pre-trained models such as BERT, GPT and LLaMA. This should reduce the rate at which hallucinations are generated and make it easier to identify them.

The particular sub-objectives of the research are then as follows:

1. Set up an appropriate hardware and environment to execute the models on which the work will be built on top of, such as [8, 10, 11, 6, 5]. Select which models can be appropriately evaluated and executed.
2. Implement an architecture similar to that of KEPLER [11] for generating embeddings of KG entities and relations from their descriptions and other KG properties.
3. Determine which metrics will be used to evaluate the new KG embedding model we just mentioned, consider the methods used in the original KEPLER article [11].
4. Evaluate the architecture mentioned before for different LLMs such as BERT, GPT and LLaMA using the metrics chosen in the last step. Determine which variation is more effective and fast.
5. Implement the new KG embedder (including all the variations with the models mentioned before) into the query answering method from [8].
6. Determine again which metrics will be used to evaluate the new query answering model, consider the methods used in the original Continuous Query Decomposition article [8].

7. Evaluate the new query answering model with different KG datasets using the metrics determined before. Find which variation of the embedder is more effective. Also evaluate the inference time for different parameters and model sizes, as this will be important for the final inference time and training of the overall model.

8. Design the architecture for allowing the new complete Augmented Language Model to make queries using the method mentioned before, using the pre-existing architecture of ERNIE [10].

9. Implement the new architecture and connect it to the query answerer appropriately.

10. Once again, determine the metrics that will be used to evaluate the complete Augmented Language Model, consider the methods used with ERNIE [10].

11. Evaluate using the metrics determined in the last step whether the frequency with which hallucinations appear improves when generating text from factual data. Evaluate the length and rate of hallucinations in the output.

12. Evaluate the usefulness for identifying hallucinations of the explanations provided by the model's internal query log.

13. Evaluate the hardware resources and inference time needed to train and execute all stages of the model.

6 Related Work

This thesis makes heavy usage of developments made in previous work on LLMs and Neurosymbolic AI. Previous work with the same aims as this one, obtaining explainable results to language model output and allowing it to access external data, has been done with different approaches. This leads us to LLMs with External Data Access mentioned in the first subsection. In the second subsection, we also go over the previous work from the field of Neurosymbolic Artificial Intelligence for query answering and integration of KGs into LLMs, mentioning a taxonomy for the work on this field. Overall, previous work on these subjects is very useful and serves as a framework for the development of this thesis' model.

6.1 LLMs with External Data Access

Previous work on providing language models with external data includes Google **LaMDA** [24], OpenAI **WebGPT** [23], and DeepMind **GopherCite** [22]. These models are allowed to execute searches on the web and use it for their answers, displaying the actual data that was searched for and used. Let us start with **LaMDA (Language Models for Dialog Applications)**. This is a family of Transformer-based deep learning language models trained for dialog. The largest version of these models has 137 billion parameters and is pre-trained on 1.56 trillion words of public dialog and web text data. In the introduction article for LaMDA the authors argued that while model scaling alone could improve quality, as is seen with GPT models, improvements on safety and factual grounding were not improved as much. With the realize of these models, it was shown that enabling language models to consult external knowledge sources can lead to significant improvements towards the two key challenges mentioned by them: safety and factual grounding. Indeed, the effectiveness of model scaling is measured with the three key metrics of quality, safety and groundedness.

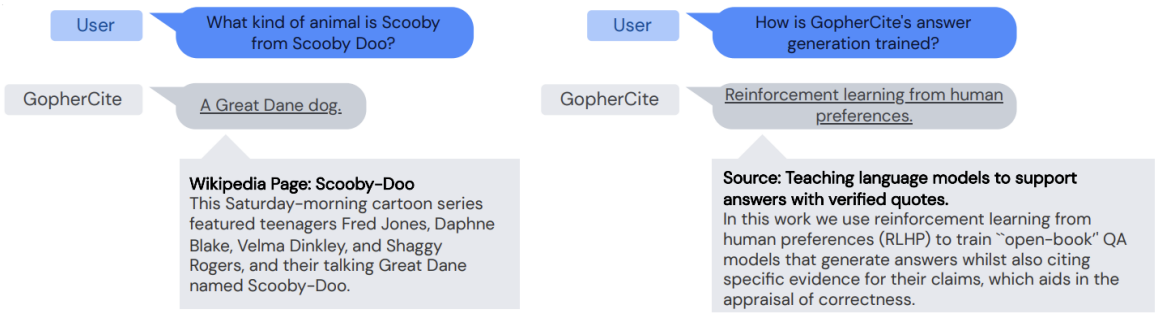


Figure 11: Answers and supporting quotes provided by GopherCite. Image obtained from [22].

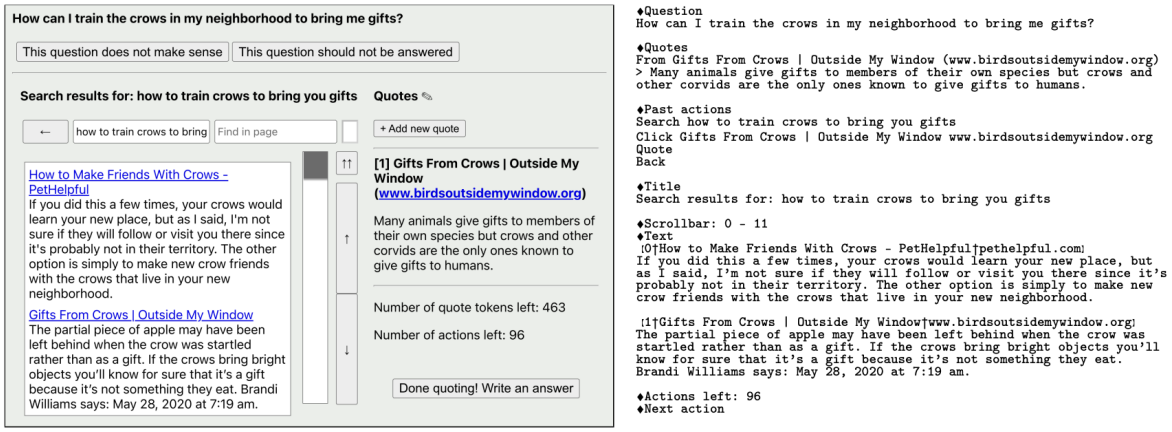
Another language model that uses external data in its generation is **GopherCite**, a 280 billion parameter model with the ability to produce answers with supporting evidence and to abstain from answering when there is not enough information. This evidence takes the form of a direct quote extracted from the original source retrieved by a web search or another suitable information retrieval system, as shown in Figure 11. These self-supporting answers can be seen as an explanation for the inference of the model, and thus puts this work into the subfield of explainable AI we mentioned earlier. For doing this, the model extracts related documents to the text of a dialog and presents the Transformer part of the model with a large context along with the dialog. Thus, having to process a much larger input and being encouraged to base its answers on the provided context rather than on the data used in training.

Finally, the most recent language model integrated with external data is **WebGPT** [23]. This is a version of OpenAI GPT-3 fine-tuned to answer long-form questions by using a text-based web browsing environment, once again allowing the model to navigate the web. The environment is designed in such a way that there is a version for humans to use and another for the model, as seen in Figure 12. This is done so that actual data on how a human would use the interface for searching and browsing is obtained. WebGPT then tries to replicate these behavior patterns with its fine-tuning and few-shot learning capabilities. Once again, the log of these actions and behaviors works as an explanation for the inference done by the model, as GopherCite does.

6.2 Neurosymbolic Artificial Intelligence

Little work has been done on integrating the querying on the internal inference of language models, as we propose. Nonetheless, a large amount of the existing work on Neurosymbolic Artificial Intelligence should be usable with our architecture. There is plenty of work in the matter, specifically related to query answering and edge completion on Knowledge Graphs. As mentioned in [9] the tasks on which the literature on Knowledge Graphs combined with Natural Language Processing have focused can be classified using the taxonomy shown in Figure 13. From this classification, we are particularly interested in the "Knowledge Application" class, which then branches into "Natural Language Generation" and also includes **Augmented Language Models (ALMs)** and **Question Answering (QA)**. These last two tasks concern the essential models for this thesis that are **KEPLER**, **ERNIE** and the **CQD** model mentioned in [8].

By integrating these last models and others in the category of ALMs and QA into a model that can provide similar explanations to those of the models with internet access we mentioned, we should be able to develop a model that can access and reliably explain specialized data such as medical



(a) Screenshot from the demonstration interface.

(b) Corresponding text given to the model.

Figure 12: WebGPT's text-browsing environment. The left one is what is shown to humans, and the right one to the model. Image obtained from [23].

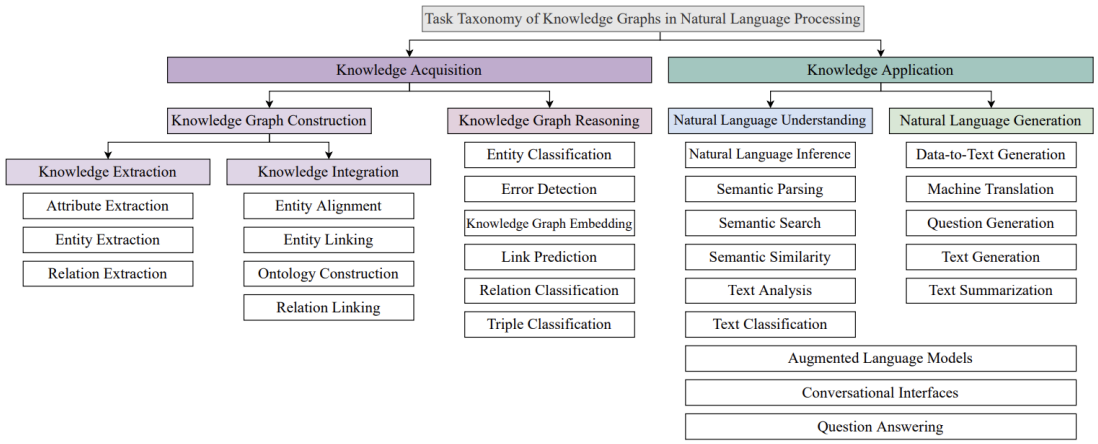


Figure 13: Taxonomy of tasks on Knowledge Graphs and Natural Language Processing as mentioned in [9].

knowledge and business intelligence logic. As mentioned in past work, [22, 23, 24] allowing models to query external data greatly increases the factual correctness of their answers as opposed to just increasing the scale of the models which, although it improves the quality and complexity of generated text, does not change the groundedness of the answers as much.

7 Methodology

Training and testing of the models we propose in this thesis is of utmost importance and the largest part of the work that will be performed. We have defined the following stages to achieve the objectives mentioned Section 4:

1. **Environment.** Set up appropriate hardware resources and software environments to load and train both the models and the datasets that will be used. This could use cloud platforms such as AWS SageMaker, Azure Machine Learning, or Google Colab. Another option would be to adjust model sizes and expectations so that training can be run on local hardware and resources, such as Tecnológico de Monterrey's GPU infrastructure or personal computers.

2. **Determine appropriate metrics for the new KG embedder.** Research the methods used for evaluation by the original KEPLER model and other similar KG embedding models. Determine which metrics are feasible and appropriate in the context of this thesis.

3. **Implement a KG embedder similar to KEPLER.** Use the architecture shown in Figure 7 and replace the encoder blocks with embedders from relevant language models such as Open AI GPT, BERT and LLaMA. For this, we will fork the KEPLER repository and replace and adjust the exact part of the Python code where the encoder is utilized. A class instantiation structure should be adopted so that different language models can be used easily.

4. **Train the modified KG embedder.** Run training of different versions of the embedder with different language model sizes on a suitable KG dataset such as Wikidata5M with the same objectives as the ones used in KEPLER. For this, a suitable notebook or script should be developed and set up to run on the hardware mentioned in the first stage. Suitable checkpoint, notification, continuous integration, and logging frameworks should be utilized to ensure this stage moves smoothly.

5. **Evaluate the different variations of the embedder.** Prioritize metrics such as quality of the embeddings and inference time of the final model. Ensure that evaluation is run with the same dataset partitions for each model, but in such a way that data is meaningfully distributed as to prevent overfitting of any kind. Compare to original KEPLER model to ensure that an improvement is indeed reached. In case none or little improvement is achieved with the modified embedders default to the original model or revisit and change the integrations of the language models.

6. **Write a technical report with the results.** Report the new integration developments and the performance of the embedder. Compare with state-of-the-art models and write an article if performance is improved.

7. **Implement the modified KG embedders into the Complex Query Answerer shown in [8].** For the CQD repository and replace the utilized KG embeddings with the ones obtained from the modified KEPLER model obtained in previous steps. Again, a suitable class instantiation structure should be adopted in order to test with different versions of the KG embedder.

8. **Design a suitable query-answer dataset.** Use the Wikidata5M dataset design a suitable set of query-answer pairs, also adapt the FB15k and FB15k-237 KG query-answer datasets used in the original version of the query answerer.

9. **Determine appropriate metrics for the new query answering model.** Research the methods used for evaluation by the original CQD model and other similar query answerers. Determine which metrics are feasible and appropriate in the context of this thesis.

10. **Train the new query answering model.** Use the *Hits at 3* (H@3) metric used to train the original CQD model to train the new model. Develop a suitable training script or notebook with appropriate checkpoint, notification, continuous integration, and logging frameworks.

11. **Evaluate all the variations of the query answering model.** Prioritize the H@3 metric and inference time to ensure that the following steps of the model get the best possible performance. Compare with performance of the original CQD model. In case this does not improve significantly, revisit the earlier stages and modify the integrations of the earlier models, consider using smaller variations.

12. **Write a technical report with the results.** Report the new integration developments and the performance of the query answerer. Add details of how the dataset was compiled and consider making a separate article for that. Compare with state-of-the-art query answerers and write an article if performance is improved.

13. **Design the architecture and implementation of the new Augmented Language Model based on ERNIE.** Use the schematics shown in Figure 10 to develop a model architecture that can

    - Generate queries in a suitable format inside model inference.
    - Execute the generated queries efficiently using the modified CQD model developed in previous steps.
    - Efficiently fuse the information obtained from the entities returned from queries generated by the model.

14. **Implement the designed Augmented Language Model architecture.** Develop a script that implements the architecture of the model described. Structure it in such a way that suitable hyperparameters and variations of KG embedders and query answerers can be tested seamlessly.

15. **Implement temporal logic-based loss functions.** Develop Isabelle HOL scripts for loss function implementation for the Augmented Language Model querying modules and integrate using PyTorch as is shown in [20].

16. **Determine appropriate metrics for the complete Augmented Language Model.** Research the methods used for evaluation of faithfulness and coherency by ERNIE and current state-of-the-art language models. Make emphasis on counting the frequency with which hallucinations and false data appear. Determine which metrics are feasible and appropriate in the context of this thesis.

17. **Train the complete model.** Write a pipeline for training the implementation of the new Augmented Language Model with a suitable dataset similar to that of the original ERNIE. Once again, use a framework with appropriate checkpoint, notification, continuous integration and logging functionalities.

18. **Evaluate the performance of the complete model.** Focus on metrics that measure the number and length of hallucinations. Also devise a metric that ensures that the explanation provided by the query logging is meaningful to the user. Use different variations of the model with different sizes and compare their performance.

19. **Write a technical report with the results of the overall model.** Report the new integration developments and the performance of the complete Augmented Language Model. Include extensive evaluations on different variations of the model. Compare with state-of-the-art models and write an article if performance is comparable or superior.

20. **Compile findings into final thesis.** Use the written articles along with all the relevant documentation and discussion to write the final thesis document. Include in-depth explanations of how each stage was connected.

8 Work Plan

In Figure 14 we show a three stage activity plan for the complete thesis project, designed to be finished in one year after its start.

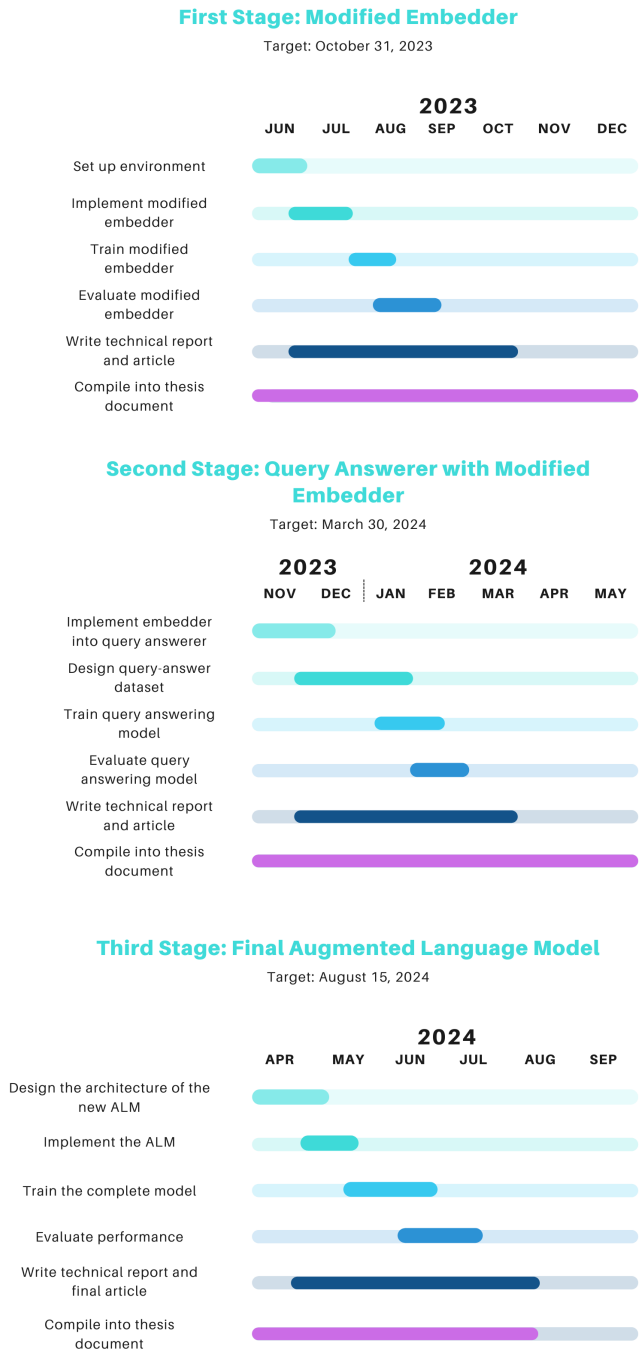


Figure 14: Three stage work plan for the thesis project.

References

[1] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. "Sparks of Artificial General Intelligence: Early Experiments with GPT-4". In: *arXiv preprint arXiv:2303.12712* (2023).

[2] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language Models Are Unsupervised Multitask Learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[5] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT*. Vol. 1. 2019, p. 2.

[6] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971* (2023).

[7] Lauren Nicole DeLong, Ramon Fernández Mir, Matthew Whyte, Zonglin Ji, and Jacques D. Fleuriot. *Neurosymbolic AI for Reasoning on Graph Structures: A Survey*. 2023. arXiv: 2302.07200 [cs.AI].

[8] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. *Complex Query Answering with Neural Link Predictors*. 2021. arXiv: 2011.03459 [cs.LG].

[9] Phillip Schneider, Tim Schopf, Juraj Vladika, Mikhail Galkin, Elena Simperl, and Florian Matthes. *A Decade of Knowledge Graphs in Natural Language Processing: A Survey*. 2022. arXiv: 2210.00105 [cs.CL].

[10] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. *ERNIE: Enhanced Language Representation with Informative Entities*. 2019. arXiv: 1905.07129 [cs.CL].

[11] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. *KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation*. 2020. arXiv: 1911.06136 [cs.CL].

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[13] Sebastian Ruder. *Recent Advances in Language Model Fine-tuning*. http://ruder.io/recent-advances-lm-fine-tuning. 2021.

[14] Matthias Bastian. *GPT-4 has a trillion parameters - report*. Mar. 2023. URL: https://the-decoder.com/gpt-4-has-a-trillion-parameters/.

[15] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL].

[16] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. *Neural Text Generation with Unlikelihood Training*. 2019. arXiv: 1908.04319 [cs.LG].

[17] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. "Survey of Hallucination in Natural Language Generation". In: *ACM Computing Surveys* 55.12 (Mar. 2023), pp. 1–38. DOI: 10.1145/3571730. URL: https://doi.org/10.1145%2F3571730.

[18] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. "On Faithfulness and Factuality in Abstractive Summarization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 1906–1919. DOI: 10.18653/v1/2020.acl-main.173. URL: https://aclanthology.org/2020.acl-main.173.

[19] *Knowledge Graphs - Amazon Neptune - Amazon Web Services*. 2021. URL: https://aws.amazon.com/neptune/knowledge-graphs-on-aws/.

[20] M. Chevallier, M. Whyte, and J.D. Fleuriot. *Constrained Training of Neural Networks via Theorem Proving*. English. WorkingPaper. ArXiv, July 2022. DOI: 10.48550/arXiv.2207.03880.

[21] Waylon Li, Yftah Ziser, Maximin Coavoux, and Shay B Cohen. "BERT is not The Count: Learning to Match Mathematical Statements with Proofs". English. In: *Proceedings of the Conference of the 17th European Chapter of the Association for Computational Linguistics 2023*. The 17th Conference of the European Chapter of the Association for Computational Linguistics, 2023, EACL 2023 ; Conference date: 02-05-2023 Through 06-05-2023. Jan. 2023. URL: https://2023.eacl.org/.

[22] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. *Teaching language models to support answers with verified quotes*. 2022. arXiv: 2203.11147 [cs.CL].

[23] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. *WebGPT: Browser-assisted Question-answering with Human Feedback*. 2022. arXiv: 2112.09332 [cs.CL].

[24] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. *LaMDA: Language Models for Dialog Applications*. 2022. arXiv: 2201.08239 [cs.CL].

[25] Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. "Ranking Generated Summaries by Correctness: An Interesting but Challenging Application for Natural Language Inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2214–2220. DOI: 10.18653/v1/P19-1213. URL: https://aclanthology.org/P19-1213.

[26] Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Paco Guzman, Luke Zettlemoyer, and Marjan Ghazvininejad. "Detecting hallucinated content in conditional neural sequence generation". In: *arXiv preprint arXiv:2011.02593* (2020).