ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12241
Τηλ: +30 210 538-1614
Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνητή Νοημοσύνη και Βαθιά Μάθηση
https://aidl.uniwa.gr/

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
DEPARTMENT OF INDUSTRIAL DESIGN AND PRODUCTION ENGINEERING
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
250, Thivon Str., Athens, GR-12241, Greece
Tel: +30 210 538-1614

Master of Science in
Artificial Intelligence and Deep Learning
https://aidl.uniwa.gr/

Master of Science Thesis

Augmentation of Large Language Model capabilities with Knowledge Graphs

Student: Tsakalakis, Viktor
Registration Number: AIDL-0034

MSc Thesis Supervisor
Dr. Kasnesis, Panagiotis

ATHENS-EGALEO, March 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΗΣ ΣΧΕΔΙΑΣΗΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
Θηβών 250, Αθήνα-Αιγάλεω 12241
Τηλ: +30 210 538-1614

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών
Τεχνητή Νοημοσύνη και Βαθιά Μάθηση
https://aidl.uniwa.gr/

UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING
DEPARTMENT OF INDUSTRIAL DESIGN AND PRODUCTION ENGINEERING
http://www.eee.uniwa.gr
http://www.idpe.uniwa.gr
250, Thivon Str., Athens, GR-12241, Greece
Tel: +30 210 538-1614

Master of Science in
Artificial Intelligence and Deep Learning
https://aidl.uniwa.gr/

Μεταπτυχιακή Διπλωματική Εργασία

Ενίσχυση ικανοτήτων Μεγάλων Γλωσσικών Μοντέλων με Γνωσιακούς Γράφους

Φοιτητής: Τσακαλάκης Βίκτωρ
ΑΜ: AIDL-0034

Επιβλέπων Καθηγητής
Δρ. Κασνέσης Παναγιώτης

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, Μάρτιος 2024

This MSc Thesis has been accepted, evaluated and graded by the following committee:

Supervisor	Member	Member
Panagiotis Kasnesis Digitally signed by Panagiotis Kasnesis Date: 2024.03.12 10:59:57 +02'00'	Digitally signed by Charalampos Patrikakis Date: 2024.03.12 13:37:22 +02'00'	Eleni Aikaterini Leligkou Digitally signed by Eleni Aikaterini Leligkou Date: 2024.03.12 13:41:38 +02'00'
Kasnesis, Panagiotis	Patrikakis, Charalampos	Leligou, Nelly
Lecturer	Professor	Professor
Department of Electrical & Electronics Engineering	Department of Electrical & Electronics Engineering	Department of Industrial Design and Production Engineering
University of West Attica	University of West Attica	University of West Attica

Copyright © Με επιφύλαξη παντός δικαιώματος. All rights reserved.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ και Τσακαλάκης Βίκτωρ,
Μάρτιος, 2024

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον/την συγγραφέα του και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις θέσεις του επιβλέποντος, της επιτροπής εξέτασης ή τις επίσημες θέσεις του Τμήματος και του Ιδρύματος.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος Τσακαλάκης Βίκτωρ του Θωμά, με αριθμό μητρώου AIDL-0034 μεταπτυχιακός/ή φοιτητής/ήτρια του ΔΠΜΣ «Τεχνητή Νοημοσύνη και Βαθιά Μάθηση» του Τμήματος Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών και του Τμήματος Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής, της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής,

δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Η εργασία δεν έχει κατατεθεί στο πλαίσιο των απαιτήσεων για τη λήψη άλλου τίτλου σπουδών ή επαγγελματικής πιστοποίησης πλην του παρόντος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.»

Ο/Η Δηλών/ούσα
Βίκτωρ Τσακαλάκης

Copyright © All rights reserved.

University of West Attica and Viktor Tsakalakis
March, 2024

You may not copy, reproduce or distribute this work (or any part of it) for commercial purposes. Copying/reprinting, storage and distribution for any non-profit educational or research purposes are allowed under the conditions of referring to the original source and of reproducing the present copyright note. Any inquiries relevant to the use of this thesis for profit/commercial purposes must be addressed to the author.

The opinions and the conclusions included in this document express solely the author and do not express the opinion of the MSc thesis supervisor or the examination committee or the formal position of the Department(s) or the University of West Attica.

Declaration of the author of this MSc thesis

I, Viktor Thomas Tsakalakis, with the following student registration number: AIDL-0034, postgraduate student of the MSc programme in "Artificial Intelligence and Deep Learning", which is organized by the Department of Electrical and Electronic Engineering and the Department of Industrial Design and Production Engineering of the Faculty of Engineering of the University of West Attica, hereby declare that:

I am the author of this MSc thesis and any help I may have received is clearly mentioned in the thesis. Additionally, all the sources I have used (e.g., to extract data, ideas, words or phrases) are cited with full reference to the corresponding authors, the publishing house or the journal; this also applies to the Internet sources that I have used. I also confirm that I have personally written this thesis and the intellectual property rights belong to myself and to the University of West Attica. This work has not been submitted for any other degree or professional qualification except as specified in it.

Any violations of my academic responsibilities, as stated above, constitutes substantial reason for the cancellation of the conferred MSc degree.

The author
Viktor Tsakalakis

This thesis is dedicated to the Open Source movement, community, and all its members, without whom it would not have been possible.

Abstract

This postgraduate thesis explores the possibility for augmentation of the abilities of Large Language Models (LLMs) in the task of Question Answering by incorporating the technique of Retrieval-Augmented Generation (RAG) in conjunction with Knowledge Graph triples. Leveraging the structured knowledge representation of semantic triples, the study aims to enhance the contextual understanding and precision of LLMs in responding to user queries. The methodology proposed consists of a pipeline for the extraction of triples from source documents (Wikipedia articles) using a fine-tuned Language Model, their subsequent storage and retrieval through RAG, and their inclusion as contextual information to assist the LLM with providing grounded and reliable answers which are based on the provided sources. The findings suggest that while the answers show an improvement compared to unaided LLM generation, this specific methodology is not competitive with more conventional RAG applications.

Keywords

artificial intelligence, deep learning, knowledge graphs, large language models, natural language processing

Περίληψη

Αυτή η μεταπτυχιακή διπλωματική εργασία εξερευνά τη δυνατότητα ενίσχυσης των ικανοτήτων Μεγάλων Γλωσσικών Μοντέλων (Large Language Models – LLMs) στο πρόβλημα της απάντησης ερωτήσεων μέσω της ενσωμάτωσης της τεχνικής παραγωγής κειμένου με ενίσχυση ανάκτησης (Retrieval-Augmented Generation – RAG) σε συνδυασμό με τριπλέτες Γνωσιακών Γράφων. Εκμεταλλεύοντας τη δομημένη αναπαράσταση γνώσης των σημασιακών τριπλέτων, η εργασία αποσκοπεί στο να βελτιώσει την κατανόηση του κειμένου όπως και την ακρίβεια των LLMs στην απάντηση ερωτήσεων χρηστών. Η μεθοδολογία που προτείνεται αποτελείται από μία ακολουθία για την εξαγωγή των τριπλετών από τις έγγραγες πηγές (άρθρα της Wikipedia) χρησιμοποιώντας ένα προσαρμοσμένο Γλωσσικό Μοντέλο, τη μετέπειτα αποθήκευση και ανάκτησή τους μέσω της RAG, και τη προσθήκη τους ως σχετική πληροφορία για την υποστήριξη του LLM προς την παροχή θεμελιωμένων και αξιόπιστων απαντήσεων οι οποίες βασίζονται στις παρεχόμενες πηγές. Τα πορίσματα υποδεικνύουν ότι παρόλο οι απαντήσεις δείχνουν μία βελτίωση σε σχέση με την παραγωγή απο το ανυποστήρικτο LLM, η συγκεκριμένη μεθοδολογία δεν ανταγωνίζεται πιο συμβατικές εφαρμογές RAG.

Λέξεις – κλειδιά

τεχνητή νοημοσύνη, βαθιά μάθηση, γνωσιακοί γράφοι, μεγάλα γλωσσικά μοντέλα, επεξεργασία φυσικής γλώσσας

Table of Contents

List of Tables **11**

List of Figures **12**

Acronym Index **13**

1 CHAPTER 1: INTRODUCTION **14**
1.1 Current situation **14**
1.2 Scope of the thesis **14**
1.3 Structure of the thesis **15**

2 CHAPTER 2: Background **16**
2.1 Language Models **16**
2.2 Transformers **17**
2.2.1 Attention 19
2.2.2 Encoder 19
2.2.3 Decoder 20
2.3 Large Language Models **21**
2.4 Hallucinations **22**
2.5 Retrieval-Augmented Generation **23**
2.5.1 Classification of RAG 24
2.5.2 Evaluation of RAG 26
2.5.3 Applications of RAG 27
2.6 Knowledge Graphs **28**

3 CHAPTER 3: Related works **31**
3.1 A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models **31**
3.2 RHO: Reducing Hallucination in Open-domain Dialogue with Knowledge Grounding **33**
3.3 Generalizing Question Answering System with Pre-trained Language Model Fine-tuning **34**
3.4 Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks **34**
3.5 FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge **36**
3.6 Language Model is all You Need: Natural Language Understanding as Question Answering **37**
3.7 Knowledge Graph Generation from Text **37**
3.8 QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering **37**
3.9 GreaseLM: Graph REASoning Enhanced Language Models **38**
3.10 EntGPT: Linking Generative Large Language Models with Knowledge Bases **39**

4 CHAPTER 4: Methodology **40**
4.1 Training of the triples extractor **41**
4.1.1 The T5 model 41
4.1.2 The WebNLG dataset 42
4.1.3 LoRA 42
4.1.4 Preprocessing 43
4.1.5 Training 44
4.1.6 Previous attempts 44
4.2 Extraction of triples from articles **45**

4.3 Creation and storage of triple embeddings..........45
4.4 LLM-based question answering..........46

5 CHAPTER 5: Results..........48
5.1 Triples generation..........48
5.2 Question Answering..........49
5.3 Quantitative analysis..........51
5.4 Qualitative analysis..........52

6 CHAPTER 6: Discussion..........56

7 CONCLUSIONS..........57

Bibliography – References – Online sources..........59

Appendix A: T5 training graphics..........63

Appendix B: WebNLG2020 evaluation results..........65

List of Tables

Table 1: WebNLG evaluation results and comparison..49

Table 2: The evaluation metrics for all three experiments..50

List of Figures

Figure 1: The Transformer architecture..18

Figure 2: A timeline of the explosion of new LLMs within the last years...............................22

Figure 3: Depiction of the RAG approach..24

Figure 4: Example of a Knowledge Graph generated from text through an NLP application. .29

Figure 5: The taxonomy of hallucination mitigation techniques..32

Figure 6: The FLEEK framework...36

Figure 7: The QA-GNN approach...38

Figure 8: The GreaseLM architecture...39

Figure 9: Depiction of the full KG-RAG pipeline developed for this thesis............................41

Figure 10: A sample from the WebNLG corpus...42

Figure 11: The Low-Rank Adaptation approach..43

Acronym Index

AI: Artificial Intelligence
CNN: Convolutional Neural Network
DL: Deep Learning
DNN: Deep Neural Network
GPU: Graphics Processing Unit
KG: Knowledge Graph
LM: Language Model
LLM: Large Language Model
LoRA: Low-Rank Adaptation
LSTM: Long Short-Term Memory
NLG: Natural Language Generation
NLP: Natural Language Processing
QA: Question Answering
RAG: Retrieval-Augmented Generation
RNN: Recursive Neural Network
TPU: Tensor Processing Unit
VRAM: Video Random Access Memory

1 CHAPTER 1: INTRODUCTION

1.1 Current situation

Question Answering (QA) is a significant Natural Language Processing (NLP) task that seeks to provide automatic responses to questions asked in natural language based on a given context. Recent advances in QA have been made using large pre-trained Large Language Models (LLMs), including BERT, GPT-4, and Llama. In addition to their already notable performance with zero-shot QA, these LLMs can be further improved by fine-tuning on a QA dataset.

However, despite the recent progress in QA using LLMs, several challenges remain. Arguably, the most significant obstacle is the LLMs' limited ability to reason over factual knowledge, which can lead to the production of incorrect or irrelevant responses (a phenomenon usually referred to as 'hallucination'). Another intrinsic limitation of LLMs is that their QA capabilities are confined to the knowledge contained in their training dataset, necessitating costly retraining to broaden their internal knowledge base.

1.2 Scope of the thesis

A popular solution which has emerged to address these problems is RAG, or retrieval-augmented generation. This refers to storing knowledge which the LM should have access to in a traditional storage medium such as a database, and adding the relevant information to the prompt as part of the question's context, using the techniques of embeddings and vector search to identify what data should be retrieved.

Another potential avenue to overcome these limitations is to combine LLMs with the use of knowledge graphs (KGs). KGs are structured representations of knowledge that contain entities, relationships, and attributes, which can provide a rich source of information for QA systems. By integrating LLMs with KGs, we can create more robust and accurate question answering systems that can leverage the strengths of both approaches. This combination has the potential to revolutionize the field of natural language processing and provide more effective and efficient question answering capabilities.

This thesis will define a pipeline in which documents in natural language get converted into knowledge graph triples, which are then passed into an LLM for QA. The goal is to examine whether we can achieve an improved, more reliable and grounded question answering compared to a simple LLM QA, without requiring access to the large amount of computational resources required for LLM fine-tuning, and without relying on the storage and interpretation of the full original unstructured text as is required by conventional RAG.

1.3 Structure of the thesis

In Chapter 2, we will explain some of the key concepts and technologies which this pipeline is built upon, including the evolution and current status of Language Models, the Transformer architecture, the RAG technique, the problem of hallucinations and the technology of Knowledge Graphs.

Chapter 3 will then present some of the latest developments in the field of NLP and Language Models which aim to ameliorate some of the same problems or otherwise augment the generation of LLMs through the incorporation of structured knowledge through KGs.

Chapter 4 constitutes the main contribution of this thesis, in describing in detail the methodology followed in implementing the KG-LLM-QA pipeline. This includes the specific tools and components used, the specific configuration of hyperparameters, what experiments were conducted, etc.

It is followed by Chapter 5, where the results of the experiments are presented in addition to an explanation of the chosen metrics.

Chapter 6 contains commentary on these results as well as some suggestions on possible future directions for this line of research.

Finally, the thesis ends with a conclusion, where the work otherwise detailed here is briefly summarized along with some parting thoughts.

2 CHAPTER 2: Background

In the rapidly evolving landscape of artificial intelligence, the emergence of Transformer architectures and Large Language Models (LLMs) marks a significant milestone in the quest for more sophisticated and versatile natural language processing capabilities. This chapter delves into the foundational principles of these technologies, their revolutionary impact on the field, and the challenges they introduce, such as the phenomenon of hallucinations in generated text. It further explores innovative solutions like Retrieval-Augmented Generation (RAG), which seeks to mitigate these challenges by enhancing model outputs with external knowledge. Lastly, the concept of Knowledge Graphs is introduced, a knowledge representation format with roots in the Semantic Web which can be integrated as a powerful tool for organizing and leveraging structured information in a way that complements the capabilities of LLMs. This discussion aims to provide a comprehensive background, setting the stage for the description of a pipeline making use of these technologies in Chapter 4.

2.1 Language Models

Language models (LMs) are foundational components in the field of natural language processing (NLP), enabling computers to understand, interpret, and generate human language. These models are probabilistic mechanisms designed to predict the likelihood of a sequence of words, forming the basis for a wide range of applications from speech recognition to text generation.

The journey of language models began in the 1980s, when the concept of statistical language models was first seriously considered [1]. The earliest statistical models were simplistic by today's standards, primarily focusing on the probabilities of short sequences of words without deep contextual understanding. However, they laid the groundwork for the evolution of language modeling by demonstrating the feasibility of using statistical methods for language prediction tasks.

By the turn of the millennium, the field of language modeling had significantly advanced through the adoption of various machine learning techniques. Among these, n-grams, decision trees, and context-free grammars became particularly prominent. N-gram, the dominant of the three models, analyze and predict the probability of a word based on the occurrence of its preceding $n-1$ words, offering a basic but powerful approach to understanding language structure. These models had already reached a level of sophistication to enable their application in fields such as document classification and automatic speech recognition [1].

As computational power increased and datasets grew, the limitations of traditional statistical models became apparent, especially in handling the complexity and variability of natural language. This realization paved the way for the adoption of neural networks in language modeling [2]. The 2010s marked a significant shift with the introduction of Recurrent Neural Networks (RNNs) and their derivative, Long Short-Term Memory (LSTM) networks. These architectures were capable of capturing longer-term dependencies in text, significantly improving the performance of language models on a variety of tasks such as machine translation, text summarization, and sentiment analysis [3], [4].

The most transformative advancement in language modeling came with the introduction of the transformer architecture by Vaswani et al. In 2017 [5]. Unlike its predecessors, the transformer model relies entirely on self-attention mechanisms to weigh the significance of different words in a sentence, without the need for sequential data processing. This breakthrough enabled the handling of long-range dependencies in text more effectively and efficiently, leading to unprecedented improvements in a wide array of NLP tasks.

Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), and their subsequent iterations, have set new benchmarks in language understanding and generation. These models are distinguished by their deep architectures, massive scale, and ability to perform 'transfer learning'—where a model trained on a vast corpus of text can be fine-tuned for specific tasks with relatively little additional training data.

Today, the field of language modeling is witnessing rapid advancements, with transformer models at the forefront. The scalability of these models has been demonstrated through versions with billions of parameters, capable of generating coherent and contextually relevant text over extended passages. Moreover, the application of language models has expanded beyond text to encompass multimodal tasks that include images, audio, and video inputs.

The current research is increasingly focused on improving the efficiency, reliability and interpretability of language models. Efforts are being made to make these models more accessible for real-world applications, improving the validity of their output and expanding their knowledge beyond that which was present in their training data.

The following section will focus on transformer Language Models with a deeper introspection into their mechanics.

2.2 Transformers

The advent of transformer neural networks marked a significant milestone in the evolution of deep learning architectures. Central to transformers is the attention mechanism, an innovative approach that allows the model to dynamically focus on specific segments of the input data. This process enables the model to prioritize information that it deems more relevant, thereby enhancing its ability to understand and generate contextually accurate outputs. This feature is particularly beneficial in complex NLP tasks such as machine translation, where discerning the nuanced context of each word within a sentence is crucial for accuracy and coherence.

Historically, neural network architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have incorporated mechanisms to handle sequential data and maintain context [6], [7]. However, the introduction of the transformer model, as detailed in “Attention is All You Need”, represented a paradigm shift. The architecture, relying solely on an attention mechanism, produced competitive results while removing the need for expensive recurrence or convolution entirely, proposing a more efficient and scalable approach to NLP.

Unlike RNNs, which process data sequentially while relying on an internal state to capture temporal dependencies, transformers can assess the entire input sequence concurrently. This characteristic not only facilitates the capture of long-range dependencies more effectively, but also significantly enhances computational efficiency through parallelization. The ability to process data in parallel makes transformers particularly well-suited to leverage the capabilities of modern GPUs and TPUs, contributing to their widespread adoption and success in a broad range of applications.

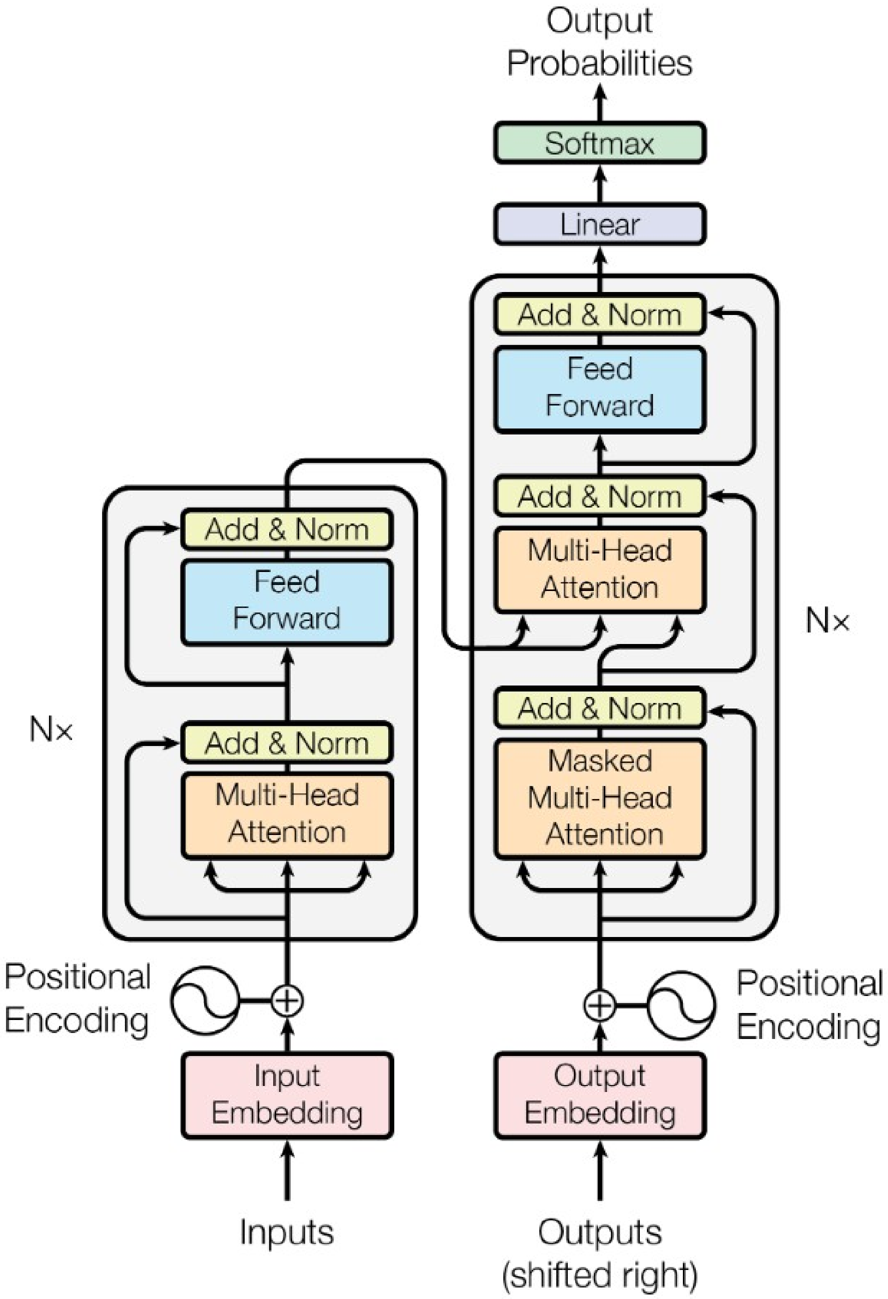


Figure 1: The Transformer architecture (source: Attention is All You Need, 2017, Vaswani et al. [5])

2.2.1 Attention

The operation of the attention mechanism within transformers can be delineated into three primary steps:

- **Creating Queries, Keys, and Values**: The mechanism begins by transforming the input data into three distinct vectors: queries, keys, and values. This transformation is achieved through learned linear mappings, facilitating a dynamic manipulation of the input information.

- **Computing Attention Scores**: The heart of the attention mechanism lies in calculating the attention scores, which are derived from the dot product of queries and keys. These scores quantify the relevance or similarity between different components of the input data, effectively determining how much focus the model should allocate to each part of the input when generating the output.

- **Applying Attention Scores**: The computed scores are then normalized using a softmax function to ensure they sum to one, signifying a probabilistic distribution of attention across the input elements. Subsequently, these scores are used to scale the value vectors, culminating in a weighted sum that represents the attention-modulated output.

To augment the capability of the attention mechanism, transformers employ a strategy known as multi-head attention. This technique involves duplicating the attention process across multiple "heads," each applying distinct learned linear transformations to the input. By aggregating the outcomes of these parallel processes, the model can assimilate information from various representational spaces, enriching its understanding and allowing for a more nuanced interpretation of the input data.

The transformer architecture is intricately designed with two primary components: the encoder and the decoder. **Figure 1** presents a schematic of the transformer architecture, with an encoder on the left side and a decoder on the right (both repeated N times in the actual model). These two components work in tandem to process and generate sequences, making transformers exceptionally versatile for tasks such as language translation, where both the understanding (encoding) of a source language and the generation (decoding) of a target language are required.

2.2.2 Encoder

The encoder's role in a transformer model is to process the input sequence and map it into a higher-dimensional space—a representation that captures the essence and context of the input. Each encoder layer, of which there can be multiple stacked on top of each other, performs two main operations:

- **Self-Attention Mechanism**: This allows the encoder to weigh the importance of different words in the input sequence relative to each other. Unlike RNNs or CNNs, this mechanism enables the model to look at the entire sequence simultaneously, capturing context from both immediate and distant elements in the sequence.

- **Position-wise Feed-Forward Networks**: After self-attention aggregates context from the entire sequence, the position-wise feed-forward networks apply further transformations to each position independently. This step enhances the encoder's ability to recognize patterns in the input and has been compared to a key-value memory [8].

Each encoder layer outputs a sequence of vectors that collectively encode the contextual relationships and semantic information of the input sequence, serving as the foundation for the decoder's operations.

2.2.3 Decoder

The decoder, in contrast, is designed to generate the output sequence from the encoded representation. It mirrors the encoder in structure but with a key difference in its attention mechanisms:

- **Masked Self-Attention**: In the decoder, the self-attention mechanism is "masked" to prevent future positions from being accessed. This ensures that the prediction for a particular position can only depend on known outputs at positions before it, maintaining the auto-regressive property necessary for sequential generation.
- **Encoder-Decoder Attention**: This layer allows the decoder to focus on different parts of the input sequence as it generates each word of the output. It does this by using the decoder's current state to query the encoder's output, integrating the context from the input sequence into the generation process.

These mechanisms enable the decoder to generate coherent and contextually relevant sequences based on the encoded input, with each layer further refining the generated output.

While the full transformer model proposed in “Attention is All you need” comprises both encoder and decoder, variations exist that utilize only one of these components for specific tasks:

- **Encoder-Only Architectures**: Models like BERT [9] utilize only the encoder part of the transformer. These models are exceptionally well-suited for tasks that require understanding or contextualizing input without generating new text, such as classification, entity recognition, or semantic similarity assessment.
- **Decoder-Only Architectures**: Conversely, models like GPT [10] operate solely with the decoder component, optimized for generating coherent and human-like sequences from a given prompt. This setup is ideal for tasks that require creativity and generation, such as text completion, QA, and more.
- **Encoder-Decoder Architectures**: In contrast to the encoder-only and decoder-only models, encoder-decoder architectures like T5 (Text-to-Text Transfer Transformer)

[11] integrate both encoder and decoder components to perform a wide range of text-based tasks. This hybrid model is designed to understand and generate text, making it highly versatile for both comprehension and production tasks.

These specialized architectures leverage the strengths of their respective components, demonstrating the transformer's adaptability and its profound impact on advancing the state-of-the-art in natural language processing and beyond.

Since their introduction, transformers have revolutionized the field of NLP, giving rise to a new generation of language models, including Large Language Models (LLMs) like GPT and BERT. These models have set new benchmarks in a range of tasks, from text generation to semantic analysis, demonstrating the versatility and power of the transformer architecture. Beyond NLP, the principles of transformers are also being explored in other domains, such as computer vision and audio processing, showcasing their potential to drive advancements across diverse areas of research and application [12].

2.3 Large Language Models

Large Language Models (LLMs) have emerged as a cornerstone in the advancement of artificial intelligence, particularly in the field of natural language processing (NLP). Among the pioneering models in this domain, BERT (Bidirectional Encoder Representations from Transformers), introduced by Google Research in 2018, marked a paradigm shift in how text-based models understand context and semantics. BERT's architecture, leveraging the transformer's bidirectional capabilities, enables it to grasp the full context of a word by considering both its preceding and following text, a significant departure from previous models that analyzed text in a unidirectional manner [9]. This innovation allowed BERT to achieve unprecedented accuracy in tasks such as sentiment analysis, named entity recognition, and question-answering, setting new standards for NLP applications.

In parallel, the development of GPT by OpenAI introduced the world to generative pre-trained transformers, which, unlike BERT's encoder-only structure, utilized an autoregressive approach to generate text [10]. GPT's design to predict the next word in a sequence laid the groundwork for a series of increasingly sophisticated generative models, including Google’s Bard or Meta’s LlaMA. This culminated in GPT-4, which is widely considered to represent the current state of the art in Natural Language Generation [13]. The iterative enhancements in GPT's versions reflect a broader trend in LLM development towards larger, more complex models capable of understanding and generating human-like text with astonishing accuracy. **Figure 2** shows a timeline of the multitude of LLMs released since 2019, and up to November of 2023. These models have not only expanded the horizons of text generation but have also demonstrated remarkable versatility across a spectrum of tasks including but not limited to machine translation, content creation, and even coding assistance [14].

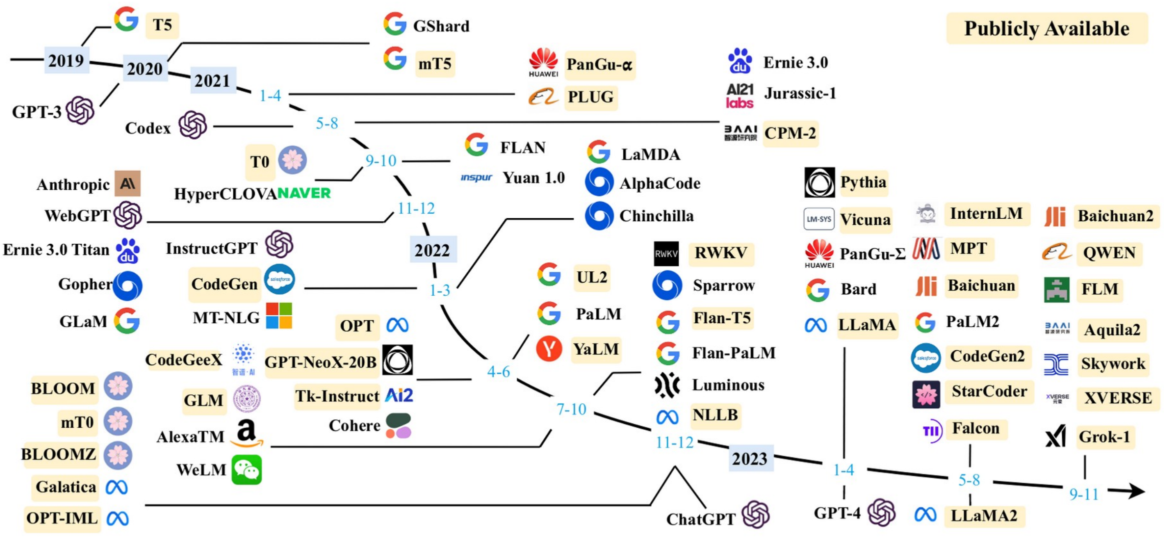


Figure 2: A timeline of the explosion of new LLMs within the last years (source: A Survey of Large Language Models, 2023, Zhao et al. [13])

Despite their impressive capabilities, LLMs are not without challenges. One of the primary issues is their tendency towards generating content that, while plausible, may not be anchored in factual accuracy, a phenomenon often referred to as "hallucination". This can lead to misinformation if the generated content is taken at face value without subsequent verification. Furthermore, these models require massive amounts of data for training, leading to significant environmental and financial costs. Additionally, LLMs can inadvertently perpetuate and amplify biases present in their training data, raising ethical concerns about their use in decision-making processes. Understanding these limitations is crucial for leveraging their potential responsibly, especially when addressing complex tasks that demand high degrees of accuracy and fairness.

2.4 Hallucinations

The phenomenon of hallucinations in LLMs currently presents a significant challenge in the field of artificial intelligence and natural language processing. Hallucinations refer to instances where LLMs generate incorrect, misleading, or nonsensical responses [15]. Unlike conventional knowledge management systems, which rely explicitly on databases to fetch factual information, LLMs operate on a fundamentally different principle. They are trained on vast corpora of text data, from which they learn patterns and associations to generate responses. However, these models do not have direct access to the data they were trained on during operation. Their outputs are generated based on the statistical likelihood of word sequences, making their reliability vary, especially so for less common queries.

The issue of hallucinations in LLMs is not isolated, but instead echoes challenges faced in other areas of AI and machine learning. For instance, in computer vision, models sometimes misinterpret images in ways that are nonsensical to humans, a problem known as "adversarial examples" [16]. Similarly, in predictive modeling, overfitting can lead models to make confident predictions based on irrelevant patterns in the data.

Neither is the problem of hallucinations merely academic; it has practical implications for the deployment of LLMs in critical applications. For example, in healthcare, an LLM might be used to provide information on drug interactions or treatment guidelines. A hallucinated response in such a context could lead to harmful recommendations. Similarly, in legal or financial advising systems, baseless or incorrect information could lead to misguided decisions with serious consequences.

Mitigation strategies for hallucinations in LLMs are diverse and still under active research. One approach is to improve the training data quality and coverage, ensuring the model has a broad and accurate base of knowledge to draw from. Another strategy involves fine-tuning the model on domain-specific datasets to increase its accuracy in particular areas of interest. Techniques like prompt engineering, where the queries to the model are carefully designed to reduce ambiguity and guide the model towards more accurate responses, have also shown promise [17].

Additionally, some solutions involve hybrid models that combine LLM outputs with rule-based systems or databases to verify facts before presenting them. This can be seen as a bridge between traditional knowledge management systems and modern LLMs, leveraging the strengths of both approaches. Another innovative solution is the development of models that can assess their own confidence in a given response, providing users with a reliability score alongside the answer [18]. This helps in flagging potentially hallucinated responses.

In conclusion, while hallucinations in LLMs present a formidable challenge, ongoing research and development efforts are aimed at mitigating these issues. Through a combination of improved training techniques, innovative model architectures, and hybrid systems that incorporate elements of traditional knowledge management, the goal is to enhance the reliability and utility of LLMs across a wide range of applications. As the field progresses, it is likely that we will see more sophisticated solutions to this problem, enabling the more widespread and safe use of LLMs in critical and everyday tasks alike.

The following section explains one such possible solution, which will also be utilized in the Methodology chapter of the thesis (Chapter 4).

2.5 Retrieval-Augmented Generation

Retrieval-Augmented Generation is a hybrid approach combining the generative capabilities of Large Language Models (LLMs) with the information retrieval process from an external knowledge base. Introduced by Lewis et al. 2020's "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [19], this technique can be used to counteract one of the fundamental limitations of LLMs: their lack of access to real-time or updated information. This limitation is responsible for what can be referred to as their "groundedness" issue and as such is a major factor in the emergence of hallucinations. By dynamically integrating retrieved information into the generation process, RAG systems can produce responses that are not only relevant and informative but also reflective of the latest knowledge.

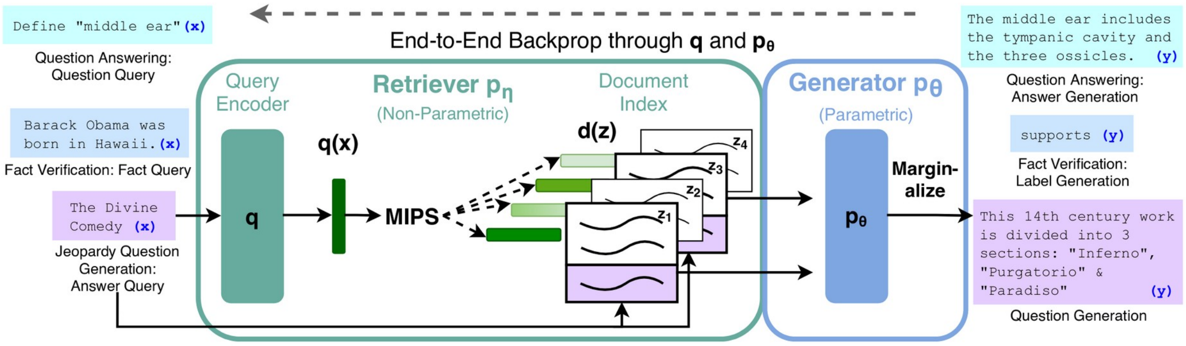

Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and p$_\theta$
Query Encoder
q
q(x)
Retriever p$_\eta$
(Non-Parametric)
MIPS
Document Index
d(z)
z$_4$
z$_3$
z$_2$
z$_1$
Generator p$_\theta$
(Parametric)
p$_\theta$
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Figure 3: Depiction of the RAG approach (source: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2020, Lewis et al. [19])

2.5.1 Classification of RAG

A significant resource in understanding RAG comes in the recent paper by Y. Gao et al., “Retrieval-Augmented Generation for Large Language Models: A Survey” [20]. It provides a comprehensive overview of the advancements in RAG technology. It delves into the evolution of RAG from its initial naive form to more advanced and modular frameworks, highlighting the integration of external knowledge to enhance the performance of Large Language Models (LLMs) in generating accurate and relevant content. The paper examines and categorizes the components of RAG, including retrieval, generation, and augmentation processes, and introduces metrics and benchmarks for RAG evaluation. Furthermore, it discusses future research directions, challenges, and the potential for expanding RAG's applications to multimodal contexts, aiming to further develop the RAG ecosystem and improve its efficacy in real-world applications. We will explore some of the authors’ work in depth in the following paragraphs.

As originally defined in the seminal paper by Lewis et al., the core mechanism of RAG typically involves two main modules, the retriever and the generator [19], depicted here in **Figure 3**.

- **Retriever**: This module is responsible for finding the most relevant pieces of information based on the input query. It uses a vector database to store embeddings of natural-language data along with the original text. When a query is received, it is converted into an embedding using a language model. This query embedding is then used to perform a similarity search in the vector database to identify the most relevant data points.

- **Generator**: Once the relevant information is retrieved, it is passed along with the original query to the generator. This module, typically a transformer-based language model, uses the combined input to generate a response. The retrieved context acts as an augmented input, providing the model with additional information that aids in producing more accurate, informed, and up-to-date outputs.

By taking into account later work in the field, Gao et al. Identify a third component in addition to those of Retrieval and Generation, that of **Augmentation** [20]. The augmentation process in

Retrieval-Augmented Generation (RAG) systems, as detailed in the paper, involves enhancing the capabilities of large language models (LLMs) through iterative, recursive, and adaptive retrieval mechanisms. This process seeks to refine the interaction between retrieval and generation components, thereby improving the overall performance and relevance of generated content. The key aspects of the augmentation process include:

- **Iterative Retrieval**: This approach enables the model to perform multiple cycles of retrieval and generation, enhancing the depth and relevance of the information used for content generation. By iteratively refining the context and content, the system can produce more accurate and contextually rich responses.
- **Recursive Retrieval**: This method involves using the outcomes of an initial retrieval as the basis for subsequent retrievals. It allows for a deeper exploration of relevant information, particularly useful in complex or multi-step queries. Recursive retrieval helps in gradually converging towards more precise and comprehensive answers by delving deeper into the subject matter.
- **Adaptive Retrieval**: Adaptive retrieval tailors the retrieval process to the specific demands of varying tasks and contexts. It involves dynamically adjusting the retrieval strategy based on the task at hand, optimizing the relevance and efficiency of the information sourced. This process allows the model to proactively determine the most appropriate moments and types of information for retrieval, thereby enhancing the pertinence and utility of the generated content.

These augmentation strategies contribute to overcoming the limitations of singular retrieval steps by enriching the model's understanding and response capabilities. They address challenges such as the “lost in the middle” phenomenon, where a single retrieval step may yield redundant or contradicting information, and the inadequacy of singular retrieval for complex problems requiring multi-step reasoning. Through iterative, recursive, and adaptive retrieval, RAG systems can achieve a more nuanced and effective integration of external knowledge, leading to improved accuracy, depth, and relevance in generated content.

The paper also introduces a distinction between Naive RAG and its more advanced counterparts, Advanced RAG and Modular RAG, which primarily lies in the sophistication and flexibility of their respective approaches to integrating retrieval into the generation process.

Naive RAG represents the earliest methodology in retrieval-augmented generation, arising shortly after the widespread adoption of models like ChatGPT. It follows a straightforward “Retrieve-Read” framework, where the process is divided into distinct stages:

- **Indexing**: Preparing and indexing data in a manner that makes it searchable. This includes data cleaning, chunking (breaking down data into manageable pieces), and embedding (converting chunks into vector representations for easier similarity comparison).
- **Retrieval**: When a query is received, the system retrieves the most relevant chunks based on similarity scores between the query vector and indexed data.
- **Generation**: The language model generates a response using the query and the retrieved documents as context.

However, Naive RAG faces several challenges, including issues with retrieval precision (leading to misaligned information retrieval), generation quality (risk of generating irrelevant or biased content), and the effective integration of retrieved content into the generation process (potentially resulting in disjointed or repetitive outputs).

Advanced RAG, in contrast, addresses some of Naive RAG's shortcomings through enhancements in the retrieval process and better integration of the retrieved information into the generation process. Specifically, it incorporates:

- **Pre-retrieval optimizations**: Enhancing the indexing and retrieval processes to improve the quality and relevance of retrieved information.
- **Fine-grained segmentation and metadata use**: For more accurate indexing and retrieval.
- **Post-retrieval strategies**: Including re-ranking and summarization to ensure that the most relevant information is used effectively in the generation process.

Finally, **Modular RAG** diverges from the traditional framework by offering greater versatility and adaptability. It allows for the customization of the RAG process to suit specific needs, integrating various methods to enhance different functional modules. This paradigm supports a serialized pipeline or an end-to-end training approach, making it possible to address specific issues more effectively.

Comparing the three categories directly:

- **Flexibility**: Modular RAG provides the most flexibility, allowing for customization and integration of various methods to enhance retrieval and generation processes. Advanced RAG, while also flexible, focuses on refining existing processes within a more structured framework.
- **Complexity**: Naive RAG is the simplest, with a straightforward process but limited by its rigidity and the quality of retrieval and generation. Advanced and Modular RAGs introduce complexity to overcome these limitations, improving the quality and relevance of the generation.
- **Use Case Suitability**: Naive RAG might be suitable for simpler applications where the limitations do not significantly impact performance. In contrast, Advanced and Modular RAGs cater to more complex scenarios requiring higher accuracy, relevance, and flexibility in handling retrieved information and generating responses.

2.5.2 Evaluation of RAG

The paper by Gao et al. outlines a detailed approach to evaluating RAG models, focusing on the assessment of both the retrieval and generation components. This comprehensive evaluation system encompasses several key aspects.

The authors recognize two major evaluation targets:

- **Retrieval Quality**: This involves assessing the effectiveness of the context provided by the retrieval module. The evaluation utilizes standard metrics from search engines, recommendation systems, and information retrieval systems such as Hit Rate, MRR

(Mean Reciprocal Rank), and NDCG (Normalized Discounted Cumulative Gain) to measure performance.

- **Generation Quality**: This focuses on the generator's ability to create coherent and relevant responses from the retrieved context. The evaluation is divided based on content objectives, including faithfulness, relevance, and non-harmfulness for unlabeled content, and accuracy for labeled content.

The evaluation of RAG models emphasizes three primary quality scores—context relevance, answer faithfulness, and answer relevance—and four essential abilities: noise robustness, negative rejection, information integration, and counterfactual robustness. These aspects collectively inform the evaluation of the two principal targets of RAG models:

- **Quality Scores**: Assess the efficiency of RAG models from different perspectives in the information retrieval and generation process. Context relevance ensures the precision and specificity of the retrieved context, answer faithfulness ensures consistency with the retrieved context, and answer relevance ensures the generated answers are pertinent to the posed questions.
- **Essential Abilities**: Include the model's capability to manage noise (noise robustness), discernment in refraining from responding when necessary (negative rejection), proficiency in synthesizing information from multiple documents (information integration), and ability to recognize and disregard inaccuracies (counterfactual robustness).

The paper also discusses various benchmarks and tools for RAG model evaluation, including RGB and RECALL for assessing essential abilities, and state-of-the-art automated tools like RAGAS, ARES, and TruLens, which use LLMs to adjudicate quality scores. These tools and benchmarks provide a framework for systematic evaluation, furnishing quantitative metrics that gauge RAG model performance and enhance comprehension of the model’s capabilities across various aspects.

This structured approach to RAG evaluation is designed to offer a comprehensive understanding of a model's performance, focusing on both the quality of retrieved context and the relevance and accuracy of the generated content. The evaluation framework plays a crucial role in identifying areas for improvement and guiding future enhancements to RAG models. The RAGAS framework is of particular interest for this thesis, as it is also the tool used for the evaluation of RAG output throughout. A more detailed explanation of RAGAS can be found in Chapter 5.

2.5.3 Applications of RAG

RAG has a wide array of applications across various domains. Some examples will be mentioned below:

- **Chatbots**: The application of retrieval-augmented generation (RAG) into chatbots represents a significant advancement in dialogue response generation, particularly within chit-chat systems. These systems face the challenge of generating meaningful and diverse responses to varied dialogue histories, a task complicated by the one-to-many nature of possible responses [21]. Traditional methods fall into two categories: retrieval-based models, which pull responses from a pre-curated dialogue corpus but struggle with novel or unique dialogue contexts, and generation-based models, which can handle unseen contexts but often produce generic or uninformative responses. RAG approaches aim to combine the strengths of both by using retrieved responses to inform and enrich the generation process, thus aiming to produce more relevant, informative, and diverse dialogue responses. This approach addresses issues of dullness and lack of specificity in generated responses by infusing the generation process with contextually appropriate information from retrieved sources.

- **Machine Translation**: Retrieval-Augmented Generation (RAG) presents a groundbreaking advancement in the field of machine translation, effectively bridging the gap between traditional translation methods and the latest in artificial intelligence technology. This method can be seen as an evolution of the older technique of translation memory, wherein human translators would search for similar phrases within a bilingual database to aid in the translation of new sentences [21]. Unlike translation memory, which relies on exact or near-exact matches, RAG leverages a vast, pre-trained language model that dynamically retrieves and generates translation content. This is achieved by first searching a large database of text for contextually similar content to the input phrase and then using this retrieved information to inform the generation process, ensuring that the translation is both contextually relevant and linguistically accurate. The integration of RAG into machine translation workflows not only significantly enhances the efficiency and quality of translations by providing a richer context but also retains the essence of leveraging past translations to inform new ones. This method, therefore, represents a significant step forward in the application of deep learning models to overcome the challenges inherent in language translation, seamlessly combining the benefits of human-like intuition with the scalability and speed of AI.
- **Question Answering Systems**: RAG significantly improves the performance of QA systems by providing access to a vast external knowledge base, enabling the generation of answers that are more precise and contextually rich. RAG has been experimentally proven to increase the quality of responses in QA systems [22], thereby standing at the forefront of AI-driven solutions to complex information retrieval and processing challenges.

In all, Retrieval-Augmented Generation represents a significant step forward in making LLMs more informative, accurate, and up-to-date. By effectively integrating external knowledge bases, RAG models not only address the inherent limitations of pre-trained language models but also open up new possibilities for their application across various domains.

2.6 Knowledge Graphs

Knowledge Graphs (KGs) are directed labeled graphs whose nodes and edges are assigned specific meanings and representations reliant on the problem domain at hand [23]. They embody a sophisticated approach to managing and structuring information, offering a dynamic framework for the representation of data. Knowledge graphs enable the construction of a semantic network that encapsulates entities—such as individuals, locations, or objects—and the intricate web of relationships that connect them. This paradigm facilitates a deeper comprehension and investigation of datasets, transforming raw data into a rich tapestry of interconnected knowledge.

The basic building blocks of KGs are nodes and edges, representing entities and their interrelations respectively. This structure is not merely a means of data storage but a canvas for mapping the hidden connections within the represented system. It allows users to navigate through the data, unveiling patterns and linkages that might remain obscured in traditional data representations like tables or lists. For example, in medicine, KGs can integrate health data from a plethora of different sources into a single homogeneous knowledge base, or even incorporating metadata such as the level of belief physicians have about a specific assertion [24].

The versatility of KGs extends to their construction methodologies, which range from manual curation to semi-automated processes, and fully automated systems leveraging advanced computational techniques [23]. The automated generation of KGs often employs Natural Language Processing (NLP) to dissect unstructured text, extracting entities and their relationships, and Machine Learning (ML) models to predict and infer new connections based on the known graph. This automation is crucial for scaling KGs, enabling the assimilation of vast amounts of data from diverse sources, including scientific literature, web content, and database records. A example of a conversion of unstructured english text into a KG, here shown in its characteristic visual representation where nodes are depicted as circles and edges as arrows, can be seen below in **Figure 4**.

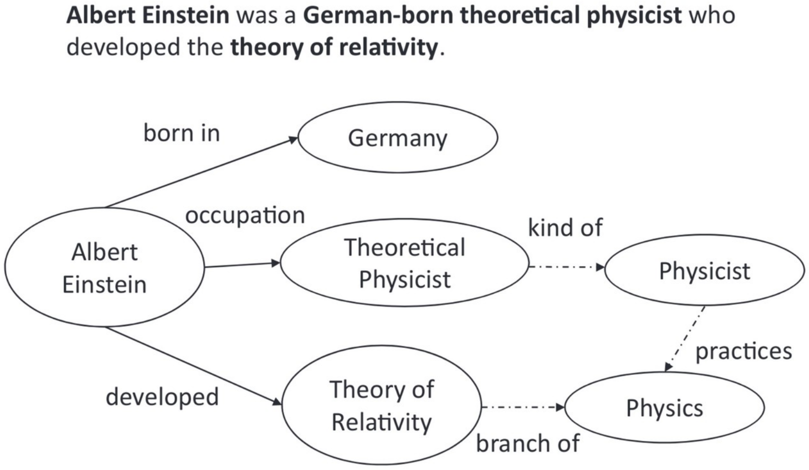


Figure 4: Example of a Knowledge Graph generated from text through an NLP application (source: Knowledge graphs: Introduction, history, and perspectives, 2022, Chaudhri et al. [23])

Furthermore, KGs are pivotal in enhancing the capabilities of artificial intelligence (AI) systems. They provide a structured knowledge base that AI can query, reason with, and augment, facilitating more informed decision-making processes. This has profound implications for various domains, including Question Answering, where KGs can enhance the abilities of a Language Model by mapping the relationships between different concepts that the LM is asked to reason upon.

In summary, Knowledge Graphs stand as a cornerstone in the evolution of data analysis and artificial intelligence. They encapsulate a move towards more intuitive, interconnected, and intelligent systems of knowledge representation, offering unprecedented opportunities for discovery, innovation, and application across a multitude of scientific and technological fields.

3 CHAPTER 3: Related works

This chapter showcases a diverse array of approaches aimed at overcoming the limitations of current LLMs, setting a foundation for future research endeavors in the field. Its goal is the exploration of advanced methodologies for enhancing the performance and reliability of large language models (LLMs) and their applications in various domains. The focus is placed on studies that have proposed innovative frameworks and techniques to address challenges such as hallucination, knowledge integration, and problem-solving capabilities in LLMs.

A significant portion of the research presented delves into the development of mechanisms aimed at reducing hallucinations in LLMs, with methodologies encompassing retrieval-augmented generation, feedback-based self-refinement, and the utilization of external knowledge sources, such as knowledge graphs, to ground model responses in factual information. Other studies propose frameworks that enhance the problem-solving abilities of LLMs by structuring reasoning processes in more sophisticated manners, thereby facilitating more complex and nuanced understanding and generation capabilities.

Furthermore, the incorporation of multi-task learning and fine-tuning on pre-trained models is highlighted as a means to extend the applicability and generalization of question answering systems. The chapter also discusses the integration of LLMs with knowledge graphs to improve question answering through enhanced reasoning and factual accuracy, alongside the introduction of novel models that leverage graph neural networks for a deeper synthesis of language understanding and structured knowledge.

3.1 A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models

The paper "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models" [17] by Tonmoy et al. (2024) presents a survey on techniques developed to mitigate hallucination in LLMs. It emphasizes the critical issue of hallucination in LLMs, highlighting how deep of an impact they have in a number of domains. The paper discusses recent advancements in detecting hallucinations, including mFACT, contextual information-based frameworks, as well as the ongoing research into whether self-contradiction acts as a possible cause contributing to hallucinations. It also discusses over thirty-two mitigation techniques, among them RAG, Knowledge Retrieval, CoNLI, and CoVe. The central contribution of this survey paper is the presentation of a detailed taxonomy categorizing these methods based on dataset utilization, common tasks, feedback mechanisms, and retriever types. The classification thus introduced aids in distinguishing the various different avenues meant for tackling problems with hallucination in LLMs. In addition, the authors analyze the drawbacks and challenges found in each of the examined techniques, to be used as a foundation in future research towards counteracting hallucinations or similar limitations faced by LLMs.

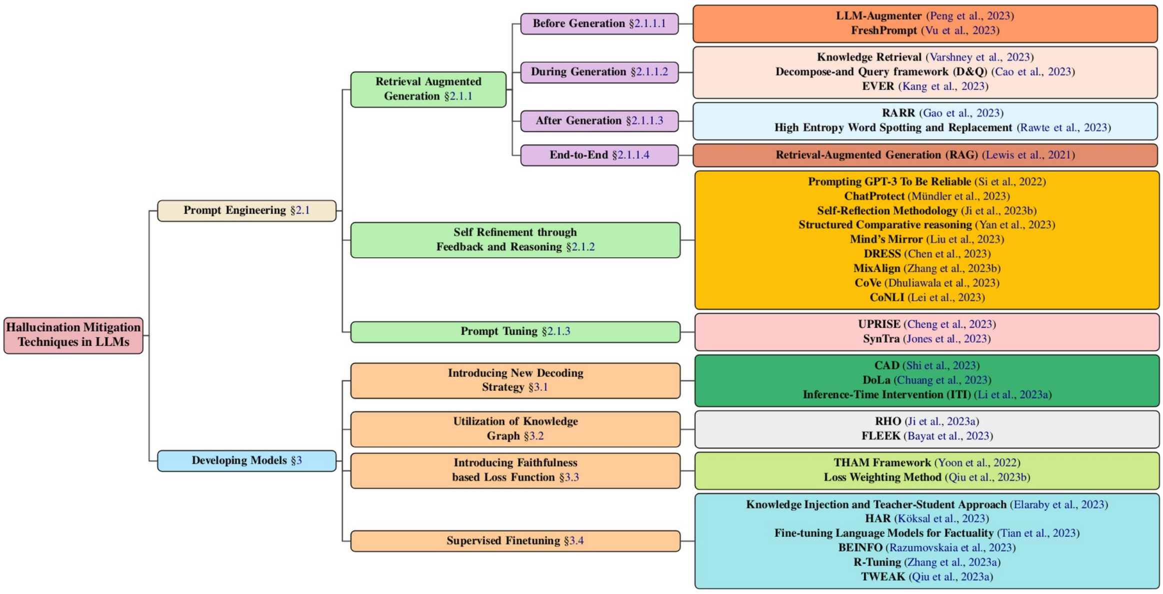

Hallucination Mitigation Techniques in LLMs
Prompt Engineering §2.1
Developing Models §3
Retrieval Augmented Generation §2.1.1
Self Refinement through Feedback and Reasoning §2.1.2
Prompt Tuning §2.1.3
Before Generation §2.1.1.1
During Generation §2.1.1.2
After Generation §2.1.1.3
End-to-End §2.1.1.4
LLM-Augmenter (Peng et al., 2023)
FreshPrompt (Vu et al., 2023)
Knowledge Retrieval (Varshney et al., 2023)
Decompose-and Query framework (D&Q) (Cao et al., 2023)
EVER (Kang et al., 2023)
RARR (Gao et al., 2023)
High Entropy Word Spotting and Replacement (Rawte et al., 2023)
Retrieval-Augmented Generation (RAG) (Lewis et al., 2021)
Prompting GPT-3 To Be Reliable (Si et al., 2022)
ChatProtect (Mündler et al., 2023)
Self-Reflection Methodology (Ji et al., 2023b)
Structured Comparative reasoning (Yan et al., 2023)
Mind's Mirror (Liu et al., 2023)
DRESS (Chen et al., 2023)
MixAlign (Zhang et al., 2023b)
CoVe (Dhuliawala et al., 2023)
CoNLI (Lei et al., 2023)
UPRISE (Cheng et al., 2023)
SynTra (Jones et al., 2023)
Introducing New Decoding Strategy §3.1
Utilization of Knowledge Graph §3.2
Introducing Faithfulness based Loss Function §3.3
Supervised Finetuning §3.4
CAD (Shi et al., 2023)
DoLa (Chuang et al., 2023)
Inference-Time Intervention (ITI) (Li et al., 2023a)
RHO (Ji et al., 2023a)
FLEEK (Bayat et al., 2023)
THAM Framework (Yoon et al., 2022)
Loss Weighting Method (Qiu et al., 2023b)
Knowledge Injection and Teacher-Student Approach (Elaraby et al., 2023)
HAR (Köksal et al., 2023)
Fine-tuning Language Models for Factuality (Tian et al., 2023)
BEINFO (Razumovskaia et al., 2023)
R-Tuning (Zhang et al., 2023a)
TWEAK (Qiu et al., 2023a)

Figure 5: The taxonomy of hallucination mitigation techniques (source: A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models, 2024, Tonmoy et. al [17])

The taxonomy provided in the paper, which can be seen above in **Figure 5**, categorizes hallucination mitigation strategies for LLMs into several key areas, each with specific approaches aimed at improving the reliability and accuracy of model outputs:

- **Retrieval-Augmented Generation (RAG):** Techniques that leverage external knowledge bases or databases during the generation process to ensure the information is grounded in reality.
- **Self-Refinement through Feedback and Reasoning:** Methods where the model iteratively refines its own outputs, using internal or external feedback loops to correct inaccuracies and reduce hallucinations.
- **Prompt Tuning:** Adjusting or optimizing the prompts given to the model to guide it towards generating more accurate and relevant responses.
- **Development of New Decoding Strategies:** Creating new algorithms for the decoding phase that prioritize factual accuracy and consistency in the generated content.
- **Utilization of Knowledge Graphs:** Incorporating structured knowledge representations, such as knowledge graphs, into the model to enhance its factual grounding.
- **Introduction of Faithfulness-Based Loss Functions:** Designing loss functions that specifically penalize the model for generating content that is not faithful to verified information or sources.
- **Supervised Fine-Tuning:** Employing annotated datasets to directly train the model on distinguishing between factual and hallucinated content, improving its ability to generate accurate outputs.

This structured approach allows researchers to systematically address the issue of hallucination in language models, targeting specific aspects of the problem with tailored solutions.

3.2 RHO: Reducing Hallucination in Open-domain Dialogue with Knowledge Grounding

The paper "RHO: Reducing Hallucination in Open-domain Dialogue with Knowledge Grounding" by Ji et al. (2023) [25] presents a novel framework aimed at minimizing hallucinations in dialogue systems by grounding responses in knowledge graphs (KGs). The authors define two types of hallucinations in open-domain dialogues: intrinsic and extrinsic. Intrinsic hallucinations occur when the generated response contradicts the dialogue history or the external knowledge. This type of hallucination is directly at odds with the information that has been provided or is well-established, making the response unfaithful to the actual context or facts. Extrinsic hallucinations, on the other hand, involve responses that are hard to verify with the dialogue history and external knowledge. These responses may not directly contradict known information but are speculative or unfounded given the available context, making them unreliable or unverifiable.

RHO incorporates local and global knowledge grounding techniques, enriching dialogue generation with structured knowledge and enabling multi-hop reasoning. Those techniques are briefly explained below:

- **Local Knowledge Grounding**: Local knowledge grounding focuses on token-level embedding from the dialogue context that can be linked to entities or relations within a KG. It involves identifying mentions of entities and relations in the dialogue history and linking these to the corresponding elements in the KG. The framework uses the TransE model for learning KG embeddings of entities and relation predicates. These embeddings are then used to generate locally grounded token embeddings for tokens in the dialogue history that are linked to specific entities or relations in the KG. This process ensures that the generated responses are contextually relevant and semantically enriched with specific, grounded knowledge, enhancing the dialogue system's ability to provide accurate and informative responses based on the immediate conversational context.
- **Global Knowledge Grounding**: Global knowledge grounding, on the other hand, goes beyond the immediate conversational context to consider the broader knowledge context. It involves aggregating the entire knowledge sub-graph related to the dialogue context into a memory bank. This is achieved by adopting an attention mechanism to draw global dependencies between the dialogue history and the memory bank that stores representations of all knowledge triples in the sub-graph. This comprehensive view allows the system to generate responses that are not only relevant to the immediate dialogue context but also informed by a broader understanding of the related knowledge domain. It enhances the multi-hop reasoning abilities of the dialogue system, enabling it to generate responses that are semantically richer and more contextually integrated over the entire conversation.

Both grounding techniques play crucial roles in the RHO framework, ensuring that the generated responses are both locally relevant and globally informed, thus significantly reducing the likelihood of hallucinations and improving the overall faithfulness and informativeness of the dialogue system.

This is demonstrated by substantial enhancements in performance metrics over existing state-of-the-art methods. Through both automatic and human evaluations, RHO shows a remarkable reduction in hallucination rates, demonstrating the efficacy of knowledge grounding in producing more accurate and reliable dialogue responses.

3.3 Generalizing Question Answering System with Pre-trained Language Model Fine-tuning

The paper by Su et al. (2019) [26] introduces a multi-task learning framework to enhance the generalization of question answering systems using large pre-trained language models like XLNet.

The study uses BERT-base and BERT-large models as baselines, following the implementation on the AllenNLP platform. These models employ a transformer architecture that receives a passage and a question, separated by an [SEP] token, and uses a linear layer to identify the span within the passage that answers the question. XLNet, a state-of-the-art pre-trained language model, is adopted for its ability to outperform BERT through a permutation language modeling objective and a two-stream attention mechanism. This allows XLNet to capture context more effectively on both sides of a given input sequence. The methodology also incorporates an attention-over-attention (AoA) mechanism, which generates an "attended attention" by considering the interactive information from both the query-to-document and document-to-query perspectives. This mechanism has proven effective in previous studies for enhancing the model's understanding of the relationship between the query and the document.

By fine-tuning on multiple question answering datasets, the framework achieves significant improvements over the BERT-large baseline, showcasing an effective approach to broadening the applicability of QA systems across diverse domains. The methodology involves rigorous experimentation with different configurations of the XLNet model, including variations in the fine-tuning approach, the use of the AoA mechanism, and the deployment on GPUs vs. TPUs. The performance of these configurations is evaluated on a range of in-domain and out-of-domain datasets to assess their generalization capabilities and overall effectiveness. The results underscore the potential of leveraging multi-task learning and pre-trained models to advance the state-of-the-art in question answering performance.

3.4 Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

The paper “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” by Lewis et al. (2020) [19], which as has already been mentioned in the previous chapter was seminal in introducing the eponymous technique, presented a novel approach in enhancing language models through the integration of retrieval-augmented generation (RAG). This methodology combines the capabilities of pre-trained sequence-to-sequence models with an external knowledge source, accessed via a dense vector index of Wikipedia.

As explained in Chapter 2, the two basic components for RAG are the generator and the retriever. For the 2020 paper, the retriever component is based on the Dense Passage Retriever (DPR) framework, which utilizes a bi-encoder architecture. This architecture comprises two main elements:

- **Document Encoder (BERT_d)**: Produces a dense representation of a document. This component encodes passages or documents into vector representations.
- **Query Encoder (BERT_q)**: Generates a query representation. It encodes the input query (e.g., a question in question-answering tasks) into a vector.

The DPR model is designed to retrieve documents relevant to a given query by calculating the similarity between the query representation and document representations. This is accomplished through a Maximum Inner Product Search (MIPS), efficiently identifying the top-k documents most relevant to the input query. The DPR retriever is pre-trained to find documents containing answers to questions from datasets like TriviaQA and Natural Questions, making it adept at selecting relevant information from a vast corpus.

The generator component employs BART-large, a pre-trained sequence-to-sequence (seq2seq) transformer model with 400 million parameters. BART is utilized for its strong performance in generating text based on input sequences, including the additional context provided by the retrieved documents from the DPR component. The integration of BART allows for the generation of coherent and contextually relevant responses or text outputs, capitalizing on BART's denoising pre-training and its capability to handle a variety of noising functions.

To combine the input with the retrieved content, the methodology simply concatenates the input sequence with the content retrieved by the DPR before feeding this combined sequence into BART for generation. This approach leverages BART's effectiveness in generating state-of-the-art results across diverse tasks, outperforming other models of comparable size.

The RAG system integrates the DPR and BART components by first using the DPR to retrieve relevant documents based on the input query and then passing both the query and the retrieved documents to BART for generating the final output. This retrieval-augmented generation approach allows the model to leverage external knowledge sources dynamically, enriching the generated output with information not contained within the model's pre-trained parameters.

The training process for this integrated system involves jointly training the retriever and generator components to minimize the negative log-likelihood of the target outputs given the input queries and the retrieved documents. This end-to-end training approach enables the model to learn how to effectively retrieve and utilize external knowledge to improve the quality and relevance of its generated outputs.

The authors demonstrate that this hybrid model significantly improves performance on a range of knowledge-intensive NLP tasks, including open-domain Question Answering and fact verification, by generating more specific, diverse, and factual content. Their experiments show state-of-the-art results on several benchmarks at the time of publication, illustrating the potential of RAG models to leverage external knowledge for more accurate and informative outputs.

3.5 FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge

The paper "FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge" [27] introduces FLEEK, a tool for detecting and correcting factual errors with evidence from external knowledge. It automates the extraction of factual claims, gathers evidence, evaluates factuality, and suggests corrections, showing promising initial results. This pipeline is illustrated below in **Figure 6**. FLEEK stands out by providing fact-level decisions with evidence, aiming for broader evaluations and enhancements in future work. The system faces limitations such as reliance on LLM-generated responses and small-scale datasets, with plans for expansion and more comprehensive evaluations.

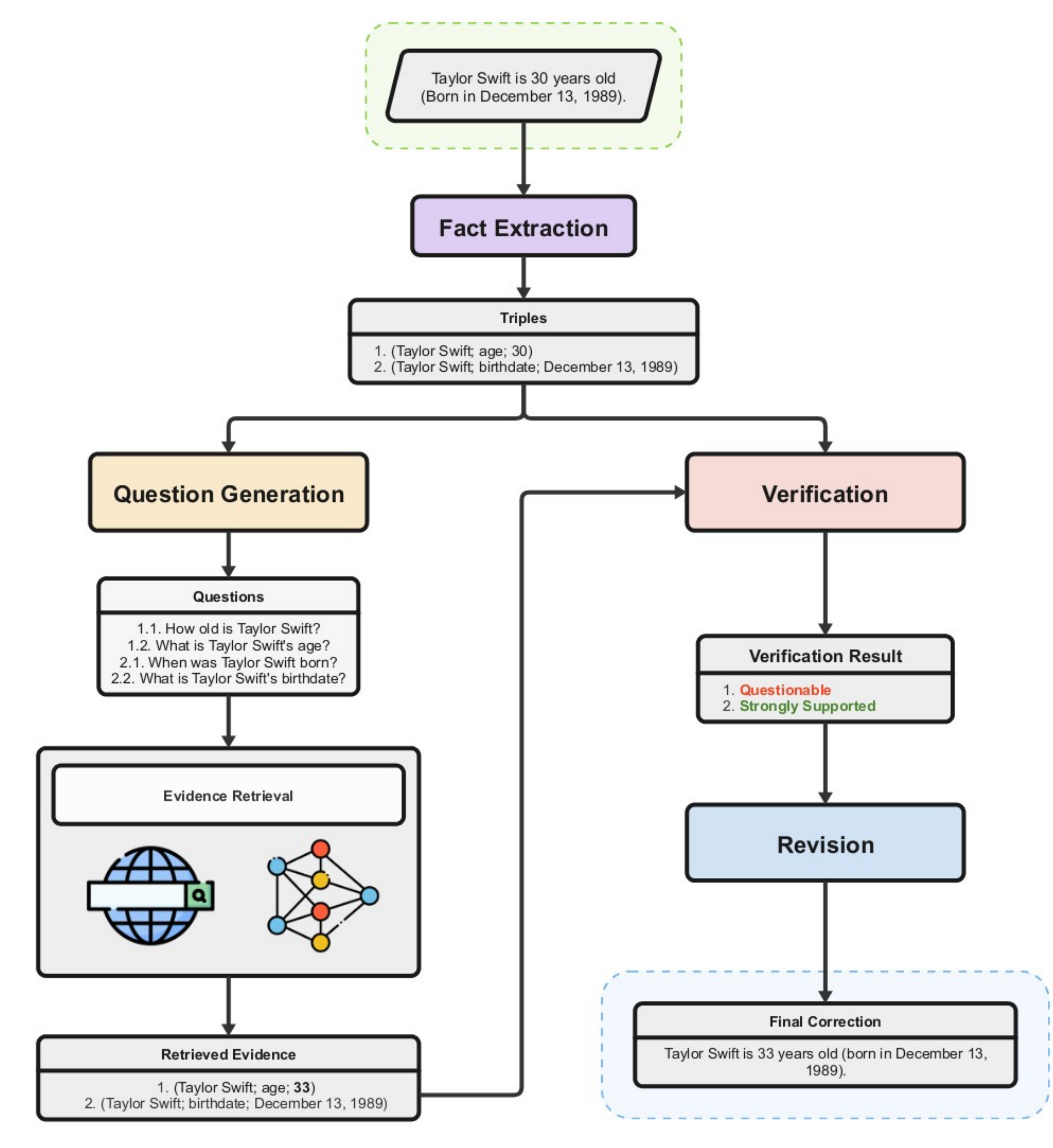

Taylor Swift is 30 years old (Born in December 13, 1989).
Fact Extraction
Triples
1. (Taylor Swift; age; 30)
2. (Taylor Swift; birthdate; December 13, 1989)
Question Generation
Questions
1.1. How old is Taylor Swift?
1.2. What is Taylor Swift's age?
2.1. When was Taylor Swift born?
2.2. What is Taylor Swift's birthdate?
Evidence Retrieval
Retrieved Evidence
1. (Taylor Swift; age; 33)
2. (Taylor Swift; birthdate; December 13, 1989)
Verification
Verification Result
1. Questionable
2. Strongly Supported
Revision
Final Correction
Taylor Swift is 33 years old (born in December 13, 1989).

Figure 6: The FLEEK framework (source: FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge, 2023, Bayat et al. [27])

3.6 Language Model is all You Need: Natural Language Understanding as Question Answering

The paper by Namazifar et al. (2021) [28] explores a novel approach to Natural Language Understanding (NLU) by mapping NLU tasks to QA problems. This methodology, termed QANLU, leverages transfer learning and demonstrates substantial improvements in intent and slot detection tasks within low-data regimes. Through extensive experiments on datasets like ATIS and Restaurants-8k, QANLU showcases significant enhancements over traditional NLU approaches. The findings suggest that QANLU not only sets new performance benchmarks but also efficiently reduces the data requirements for achieving high accuracy in NLU tasks. This work opens up new possibilities for the application of QA models in NLU, highlighting the potential of transfer learning in bridging the gap between different NLU domains and tasks.

3.7 Knowledge Graph Generation from Text

The paper “Knowledge Graph Generation from Text” by Melnyk et al. (2022) [29] proposes a novel, end-to-end multi-stage system for generating KGs from textual inputs. The system, named ‘Grapher’, is designed to first generate graph nodes using a pretrained LM. In a following step, a simple neural network module, for which the researchers propose either and LSTM or a Gated Recurrent Unit (GRU), is tasked with edge construction. The goal is the efficient extraction of KGs from text. The methodology offers several architectural choices based on available training resources, and the model's performance is evaluated using datasets like WebNLG 2020 Challenge, showing it matches or exceeds current state-of-the-art pipelines in the task of text-to-RDF generation. This system presents a viable alternative to existing graph generation approaches, aiming to streamline the process of converting unstructured text into structured knowledge.

3.8 QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering

The paper “QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering” by Yasunaga et al. (2021) [30] introduces an innovative approach that combines the strengths of pre-trained LMs and KGs through a Graph Neural Network (GNN) to improve question answering. This method, QA-GNN, addresses the challenge of effectively utilizing knowledge from both LMs and KGs by creating a joint graph that enables mutual enhancement of their representations. The model demonstrates significant performance improvements on various QA benchmarks, including CommonsenseQA [31], OpenBookQA [32], and MedQA-USMLE [33], by integrating the contextual understanding of LMs with the structured reasoning capabilities of KGs. The paper showcases how QA-GNN outperforms existing methods by facilitating deeper reasoning and providing a framework for explainable AI through its innovative architecture and relevance scoring mechanism. **Figure 7** demonstrates this pipeline.

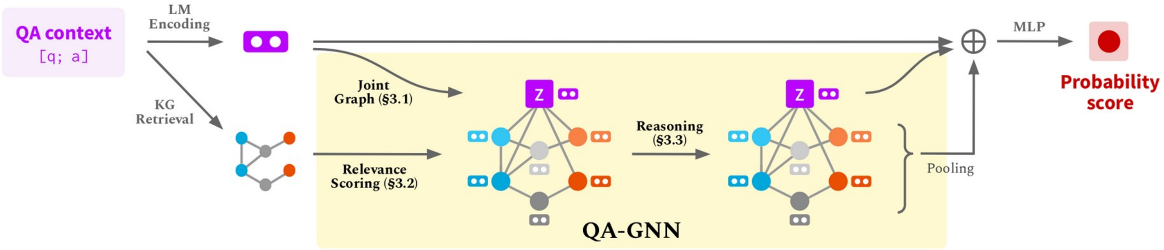


Figure 7: The QA-GNN approach (source: QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering, 2021, Yasunaga et al. [30])

In the QA-GNN approach, a user first asks a multiple-choice question. The question, along with the candidate answers, is used to search for relevant nodes in a preexisting KG using classical text search algorithms combined with the introduced LM-based relevance scoring. For most general knowledge benchmarks, Conceptnet [34] was the KG used. This creates a working graph, specific for each question-answer combination, featuring different types of nodes based on the entities' relationship to the question and answer context. This graph is then passed through an attention-based GNN, or Graph ATtention network [35], to produce a pooled representation. This representation, together with the LM's output, is used to make the prediction.

3.9 GreaseLM: Graph REASoning Enhanced Language Models

The paper "GreaseLM: Graph REASoning Enhanced Language Models" by Zhang et al. (2022) [36] builds upon QA-GNN and proposes an improved model that integrates KGs with LMs to enhance question answering capabilities. GreaseLM uses a graph neural network to enrich LM representations with structured knowledge from KGs, facilitating deep interaction between the two modalities across multiple layers, as illustrated in **Figure 8**. This approach allows for improved reasoning over questions that require understanding beyond the text, showing significant performance gains across several benchmarks, including state-of-the-art metrics in CommonsenseQA and OpenbookQA, and beating QA-GNN in MedQA-USMLE. The model outperforms existing methods by effectively blending language context with external knowledge, demonstrating its potential for complex question answering tasks that involve nuanced textual understanding and structured reasoning.

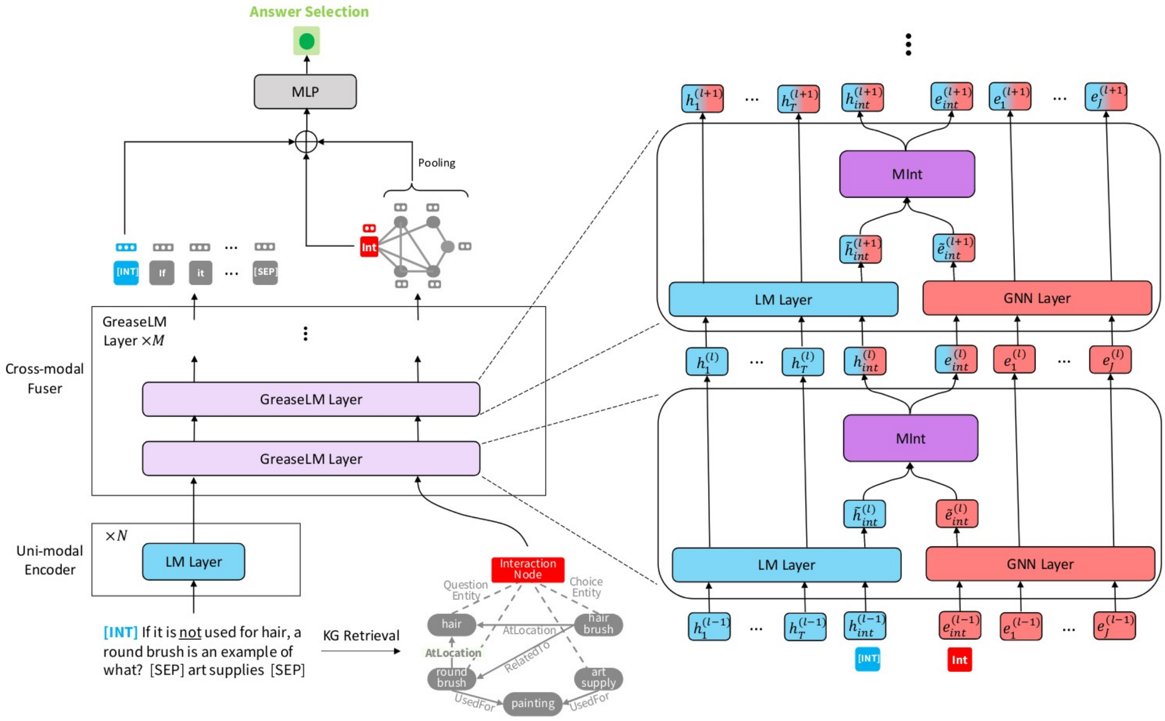


Figure 8: The GreaseLM architecture (source: GreaseLM: Graph REASoning Enhanced Language Models, 2022, Zhang et al. [36])

3.10 EntGPT: Linking Generative Large Language Models with Knowledge Bases

The paper "EntGPT: Linking Generative Large Language Models with Knowledge Bases" by Ding et al. (2024) [37] introduces and evaluates two variations of the Entity GPT model—EntGPT-P (prompt-based) and EntGPT-I (instruction-tuned). The core innovation of these models is their ability to link natural language mentions of entities generated by large language models to corresponding entities in a knowledge base, thereby reducing hallucinations commonly observed in generative models. The authors provide comprehensive experimental evidence showing that augmenting generative models with entity-based knowledge improves performance across both entity disambiguation and question answering tasks.

The study underscores the importance of entity-based knowledge in enhancing the accuracy and reliability of generative language models, especially in tasks that require precise entity understanding and reasoning. This research opens up new avenues for future exploration, particularly in extending the EntGPT framework to perform entity linking beyond disambiguation and in investigating the impact of entity disambiguation on question answering performance more deeply. The authors argue for the potential benefits of incorporating entity correlation more significantly into question answering models, pointing towards a promising direction for enhancing the capabilities of large language models through knowledge-based enhancements.

4 CHAPTER 4: Methodology

This chapter delineates the approach taken to develop a novel triples extraction system capable of converting natural language text into structured KGs. This system leverages the capabilities of Google's Flan-T5 model [38], fine-tuned to interpret and encode information into KG triples. A critical aspect of this process involved the utilization of the WebNLG 2020 dataset [39], facilitating the model's training by providing a corpus of factual statements paired with their RDF triple representations. This dataset served as a foundational element in training the model to accurately generate KG triples from natural language inputs.

The training process was designed to accommodate the computational constraints posed by the available hardware, employing the Low-Rank Adaptation [40] (LoRA) technique to significantly reduce memory requirements while maintaining the model's effectiveness. This adaptation allowed for the efficient fine-tuning of the model on a relatively modest setup, demonstrating the feasibility of conducting high-level NLP tasks without access to extensive computational resources. The methodology further details the introduction of special tokens to facilitate the model's understanding and generation of KG triples, ensuring clear delineation between distinct triples and their components.

In addition to model training, this chapter explores various attempts and methodologies employed prior to finalizing the approach, providing a holistic view of the project's developmental trajectory. Subsequent sections expand upon the extraction of triples from articles, detailing the processes and tools used to prepare text documents for KG conversion, and the creation and storage of triple embeddings to facilitate efficient retrieval and utilization in question-answering applications. The culmination of these efforts is a sophisticated question-answering system, leveraging the trained triples extractor and embedding database to improve the accuracy of answers to queries based on the relevant KG context. A depiction of the finished pipeline is in **Figure 9**.

This chapter not only highlights the technical methodologies employed but also underscores the adaptability and resourcefulness required to navigate the challenges of advanced NLP research within the constraints of available resources.

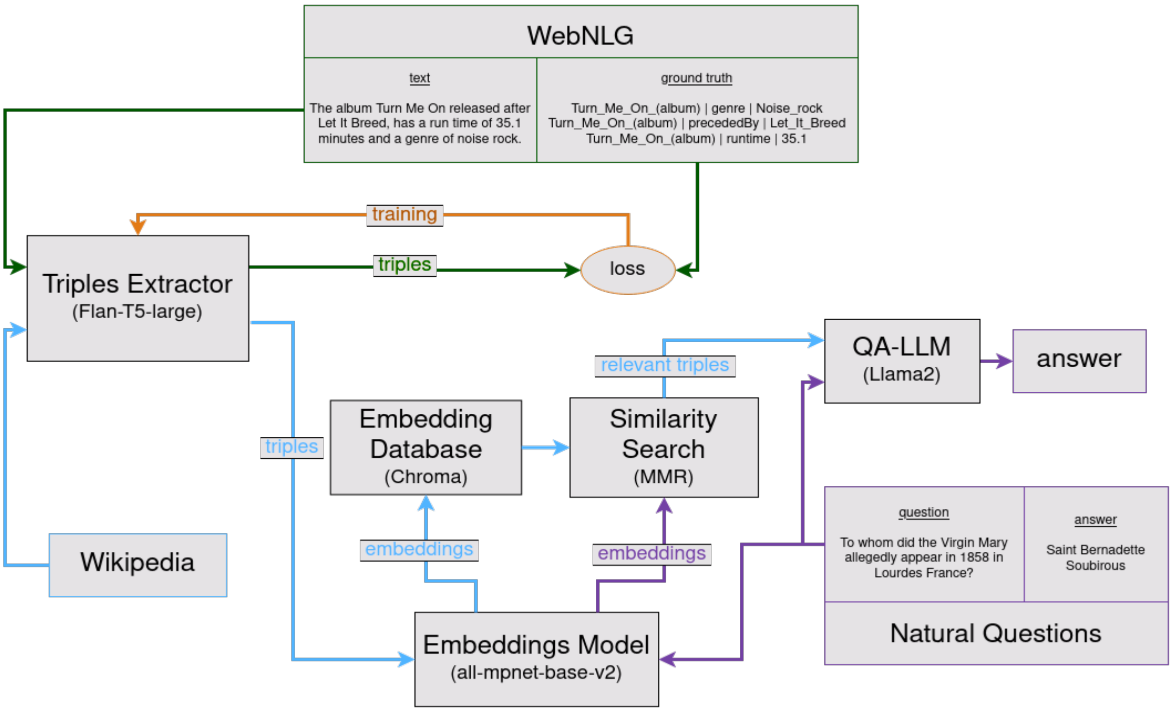

WebNLG
text
The album Turn Me On released after Let It Breed, has a run time of 35.1 minutes and a genre of noise rock.
ground truth
Turn_Me_On_(album) | genre | Noise_rock
Turn_Me_On_(album) | precededBy | Let_It_Breed
Turn_Me_On_(album) | runtime | 35.1
training
triples
loss
Triples Extractor
(Flan-T5-large)
relevant triples
QA-LLM
(Llama2)
answer
triples
Embedding Database
(Chroma)
Similarity Search
(MMR)
question
To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?
answer
Saint Bernadette Soubirous
Natural Questions
Wikipedia
embeddings
embeddings
Embeddings Model
(all-mpnet-base-v2)

Figure 9: Depiction of the full KG-RAG pipeline developed for this thesis

4.1 Training of the triples extractor

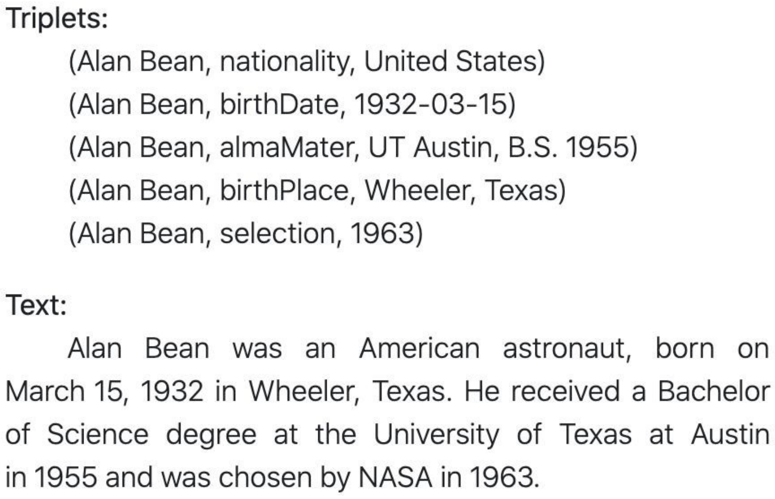
4.1.1 The T5 model

The first step in the process was acquiring an LM with the capability to produce a set of KG triples based on a given natural language English text, encoding the information contained therein. The architecture that was chosen for this model was Google's T5 [11], an encoder-decoder text-to-text transformer which was made publicly available as a pre-trained checkpoint and can be easily reassigned to a variety of NLP tasks with some slight fine-tuning. T5 repurposes all text-based language problems into a unified text-to-text format, where both the input and output are text strings. This approach allows it to handle tasks ranging from translation, summarization, question answering, and text classification to more complex tasks like simultaneous translation and synthesis of information across documents. Its flexibility and efficiency in processing and generating text make it a powerful tool for researchers and practitioners looking to tackle a diverse array of language processing challenges.

For the task at hand, both the 'small' and 'base' variants of the model were tried but found to be insufficient in the quality of their generation. The version of the model that was ultimately used was "Flan-T5 large", a 783-million parameter model that was further fine-tuned for following instructions [38]. The cross-entropy loss function was used throughout the training.

4.1.2 The WebNLG dataset

In order to train the model for the specific task of encoding KGs, it was also necessary to put it through a supervised learning process using a suitable dataset. Such a dataset is WebNLG, which provides factual statements in natural language along with an equivalent representation of those facts in the form of a RDF triples. An example is shown below in **Figure 10**. WebNLG was created as part of the 2017 paper *Creating Training Corpora for NLG Micro-Planning [39]* for the purpose of encouraging research in the field of Natural Language Generation and has since been updated a few times as part of the reoccurring WebNLG challenge in generating text from RDF data. The dataset used for this thesis was the latest available one at the time of implementation, WebNLG 2020.



Triplets:
(Alan Bean, nationality, United States)
(Alan Bean, birthDate, 1932-03-15)
(Alan Bean, almaMater, UT Austin, B.S. 1955)
(Alan Bean, birthPlace, Wheeler, Texas)
(Alan Bean, selection, 1963)

Text:
Alan Bean was an American astronaut, born on March 15, 1932 in Wheeler, Texas. He received a Bachelor of Science degree at the University of Texas at Austin in 1955 and was chosen by NASA in 1963.

Figure 10: A sample from the WebNLG corpus (source: Papers With Code)

4.1.3 LoRA

The machine that was used for the training was a home PC equipped with 16GB of DDR3 RAM and with a single NVIDIA GeForce GTX 1070ti GPU featuring 8GB of VRAM. Due to the limited resources, it would not have been possible to perform any kind of training on the full model, as the memory required to store the gradient for all the 783 million parameters would have far exceeded 8GB. For this reason, a particular training technique was used to improve memory efficiency, Low-Rank Adaptation or LoRA [40]. The idea behind LoRA is that the weights of the original model can be frozen so as not to take part in the fine-tuning, while some weight matrices can be decomposed into the product of two matrices A and B, sharing a dimension *r*, as illustrated in **Figure 11**. It has been shown that language models have an intrinsic dimension much lower than their actual weight matrices [41], so *r* can be smaller than the rank of the matrix that was decomposed. During training, only the parameters of the adapter are updated. Thus the dimensionality of the parameter space is reduced, leading to a significant decrease in memory requirements.

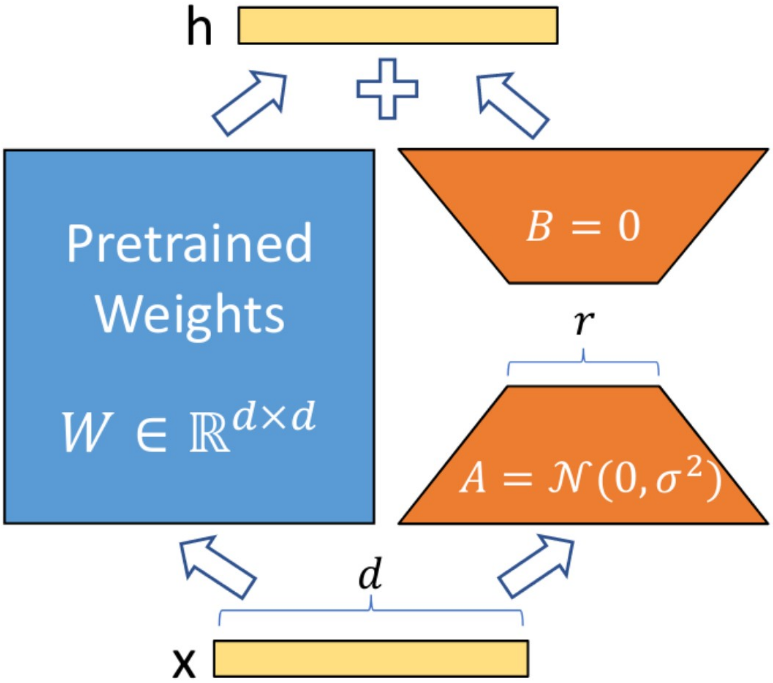

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 11: The Low-Rank Adaptation approach (source: LoRA: Low-Rank Adaptation of Large Language Models [40])

As in the original paper, the matrices targeted for decomposition for our triples extractor were the ‘query’ and ‘value’ modules of the transformer. A rank of 8 was chosen after proving sufficient experimentally. The result was that out of the more than 787 million parameters in the T5 model (including the weights of the adapter itself), a mere 4.7 million were active during the fine-tuning process, a reduction of approximately 99.4%. This made it possible to conduct the training within the available VRAM capacity of 8GB.

4.1.4 Preprocessing

Beside the original vocabulary, two special tokens were introduced to the T5 model and its tokenizer for the fine-tuning. Their purpose was to unambiguously designate the separation between one triple and the next (an inter-triple separator), as well as that between the terms of a single triple among themselves (an intra-triple separator). They were represented, respectively, as <inter_sep> and <intra_sep>.

The original dataset included several different fields of data for each sample, such as its size, a link to the original DBpedia entity the data was taken from, and the category of the entity. Of these fields, only two were considered relevant and kept, those being ‘lex.text` (the natural-language text) and ‘modified_triple_sets.mtriple_set’ (the equivalent list of triples). For each sample, in accordance with the procedure outlined by the model’s creators, a sentence giving a description of the task at hand in natural language was prepended to the paragraph of text as an instruction to the model (‘*answer the question based on the knowledge graph:* ’). For the target output, the triples were reformatted using the special tokens as noted above. The result would be an input-output pair like this:

extract knowledge graph triples: White rice is an ingredient of Arros negre which is a traditional dish from Spain.

Arròs_negre<inter_sep>country<inter_sep>Spain<intra_sep>Arròs_negre<inter_sep>ingredient<inter_sep>White_rice

Both the input and output strings were then padded and truncated to the maximum length found in the dataset, and an attention mask was created to have the model ignore the padding tokens from the input.

4.1.5 Training

For the training, the AdamW optimizer was used [42], with a learning rate reaching an initial value of 5e-5 over a warm-up period of 1000 steps and then dropping linearly to 0. An early stopping policy was also set to trigger after 3 epochs of no decrease to the validation loss. The main training run was 10 epochs, taking approximately 23.4 hours to finish and using 3.34KWh of energy. Plots showcasing the training and validation losses, the evolution of the learning rate and the power usage are included in Appendix A.

All code to perform the model training was written in Python, using the Pytorch library [43]. Pytorch Lightning was used for part of the model training [44]. The HuggingFace platform was also critical to this project, with the T5 model and its tokenizer being used through the *transformers* library, the WebNLG dataset downloaded through *datasets*, the LoRA adapter applied using *peft* and the model checkpoints being stored in HuggingFace Hub [45].

4.1.6 Previous attempts

Prior to reaching this final approach described above, there were some earlier attempts made in developing this KG encoder which will be briefly mentioned here in the interest of providing a comprehensive report. The first relied on utilizing IBM's Grapher project, a system composed of a language model and a second neural network, capable of generating complete knowledge graphs from text [29]. However, this approach was abandoned due to a combination of two reasons. First, the resources available for this project, in particular the size of the VRAM, weren't sufficient to train the whole system, even after altering the LM to use LoRA. In addition to that, despite the code being open-source, the maintainers have not released the weights of the model as a pre-trained checkpoint due to company policy [46].

Another idea which was explored was combining the LM tasked with encoding KG triples from natural language with a second LM tasked with answering questions whose answers could be found in the original text. By limiting the available QA context input to the triples and back-propagating the loss of the QA-LM to the triples extractor, it was postulated that the right conditions would be created to have the triples extractor learn to create KGs specifically suited to answering arbitrary questions. This setup proved infeasible due to the auto-regressive nature of LLMs, which makes it impossible to use the generations of one as input to another DNN while maintaining the gradient.

4.2 Extraction of triples from articles

With the triples extractor model having been trained on WebNLG to be able to generate triples, it would then be capable of extracting the knowledge in any arbitrary text document into a KG-compatible format. In preparation for the QA task outlined in section 3.4, a number of documents corresponding to the context required for answering the questions would have to be downloaded, broken into chunks and passed through the trained T5 to produce triples.

The benchmark chosen to assess the efficacy of the QA pipeline was Google's Natural Questions dataset [47]. This is composed of a number of questions posed in natural language, taken from actual user queries to the Google search engine. Each question is accompanied by one or more answers added by human annotators, either in short form of a few words or long form of a few sentences up to an entire paragraph. All answers in the dataset are in reference to a specific Wikipedia article where they are quoted from, and a link to that article is included for each sample. For the purposes of this project, only the short answers were considered. As such, samples for which short answers were not included were dropped. Furthermore, the initial subset of the dataset comprising the first 2,500 samples was selected. It should be noted that, due to time constraints, no manual selection or preliminary review of the dataset was made, and it can be observed that some of the samples are of somewhat lesser quality, with some of the questions being ambiguous without the context of a time period they are referring to or having more than one acceptable answer.

After the final dataset had been selected, the list of Wikipedia articles containing the answers was collected and downloaded. For this task, and most of the work described in sections 3.2, 3.3 and 3.4, the python library *langchain* was used [48]. Langchain is a popular open-source framework for the development of applications using LLMs, which offers various high-level tools for common use cases and pipelines. In addition to making it easy to download the articles using its *WikipediaLoader* class, its *RecursiveCharacterTextSplitter* enabled splitting the resulting documents into smaller chunks suitable for processing, while ensuring that each chunk was as close to the original meaning as possible by performing the split along semantically significant points such as paragraph or line breaks.

In an effort to improve the quality of the generated triples in cases where either the subject or the object chosen by the model became ambiguous outside the context of the original text, spacy's experimental co-reference resolver was tested [49]. However, it was observed that the quality of the replacements was somewhat erratic, with the model sometimes replacing a term with the wrong entity or with a more ambiguous term. For that reason, this step was not included in the final pipeline.

4.3 Creation and storage of triple embeddings

In order for the QA-LM to be able to retrieve the most relevant triples for each question, it was necessary to create vector embeddings for each triple and store them in an accessible data store. For our use case, the triples were encoded using the embeddings model *all-mpnet-base-*

v2 from the *sentence-transformers* library [50], after also experimenting with GPT4AllEmbeddings and LlamaCppEmbeddings with somewhat inferior results. The resulting 768-dimensional vectors were stored in an instance of the embeddings database Chroma [51].

For the retrieval of the triples, the algorithm used was Maximal Marginal Relevancy. In MMR, like most vector search algorithms, the embedding of the question is first created using the same model as that for the candidate documents (in this case the triples), then the cosine similarity between that embedding vector and each vector in the database is calculated. The innovation in MMR is that, instead of ranking the candidates by similarity and returning the top results, each vector is added one by one after ensuring that it is sufficiently dissimilar to the ones already added to the result set [52]. This promotes diversity among the returned documents, and for our case makes sure that the context passed to the LLM does not include any duplicate triples encoding the same information, giving the QA-LLM a greater chance of encountering data useful in answering the user's query. Five candidate triples were returned per query by the algorithm.

4.4 LLM-based question answering

The last step in the pipeline is the actual question-answering, with the aid of the relevant triples given as context. For this, a LLM capable of text generation and trained with the ability to chat would be needed. A number of models were considered for this role, including Mistral, Vicuna and Falcon. In the end however, the model chosen for its quality of generation and its overall status as something of a staple in the academic community was Meta's Llama.

The Llama family of models was developed by Meta AI and made public in February 2023 with the paper *LLaMA: Open and Efficient Foundation Language Models [53]*. It is an example of the GPT architecture, decoder-only transformers trained with massive amounts of unlabeled text to perform the task of text generation. Like many other LLMs, Llama came in a few variations, featuring from 7 billion up to 70 billion parameters. Thanks to its competitive performance, even against relatively much larger models, and due to being partially open-source and free to use for research purposes, Llama quickly became quite popular for academic use. In July of the same year, Meta released the updated Llama2 models, including version specifically trained for use as chatbots through Reinforcement Learning with Human Feedback [54].

The model used in our pipeline was the chat variation of Llama2-7b. As it has already been pre-trained for conversation, including QA, there was no need for further fine-tuning. The hyperparameters used for the generation were a temperature of 0.8, a top-K value of 40 and a top-P value of 0.9. The prompt template used was the following:

```
Use the following pieces of context to answer the question at the end. Keep
the answer very brief. If you don't know the answer reply "I don't know".
{context}
Question: {question}
Answer:
```

During the QA, the {question} tag would be filled in with the question at hand, and the {context} tag would be replaced with the triples returned by the similarity search on the vector database.

5 CHAPTER 5: Results

This Chapter presents a comprehensive evaluation of the developed system, focusing on two primary aspects: the generation of triples and the efficacy of the question answering (QA) pipeline. In the first part, the quality of triples generated by the model is assessed using the official evaluation script from the WebNLG2020 challenge. The fine-tuned model's performance is quantified using F1-scores, demonstrating a significant advancement over the 2020 competition's winner while highlighting areas for further improvement when compared to the state-of-the-art. The second part delves into the QA capabilities of the system through a series of experiments designed to evaluate the contribution of the triples extraction process and the RAG pipeline. Through the RAGAS framework, various dimensions of RAG performance are examined, including faithfulness, answer relevance, and context relevance, among others. This section not only presents quantitative findings from these evaluations but also sets the stage for a deeper discussion on the implications and potential enhancements for the system, as detailed in the following chapter.

5.1 Triples generation

For the assessment of the quality of the triples, it was possible to use the official evaluation script created for the WebNLG2020 challenge [55]. As explained in the tool's Github repo, four types of matching, explained below:

- **Strict**: An exact match is required between each of the three elements of the candidate triple (subject, predicate and object) with the respective elements of the reference triple.
- **Exact**: For this matching type, the three elements of the candidate triple must also be found in the reference, but not necessarily in the same order.
- **Partial**: This matching allows for each element of the candidate triple to be only a partial match to one of the elements in the reference. Similarly to exact matching, the type of each element does not matter.
- **Type**: Like partial matching, the strings composing the candidate triple need not be identical to their reference triple equivalents, however just as in strict matching they do have to be of the same type of element, subject, predicate or object.

In the below table is a comparison of the F1-score for each of these categories between the model trained in this thesis and the model that won the WebNLG2020 competition for the same category (WebNLG Text2RDF English All).

F1-score	WebNLG2020 Winner	Fine-tuned Flan-T5 (ours)
Strict	0.6864	0.8466
Exact	0.6892	0.8484
Partial	0.6964	0.8684
Type	0.7000	0.8853

Table 1: WebNLG evaluation results and comparison

As can be seen, the model trained for this thesis achieved a total F1-score ranging from 0.84 to 0.89 depending on category. This represents a significant improvement compared to the winner of the 2020 competition, whose equivalent scores were between 0.68 and 0.7, while still falling short of the current state of the art which has been reported as high as 0.93 [56], [57]. The full output of the evaluation tool is included as Appendix B.

5.2 Question Answering

For the purposes of evaluating the contribution of the triples extraction and RAG pipeline here constructed, three different experiments were conducted to compare the results thereof. The first saw the questions in the dataset posed to the LLM without the aid of any context provided through data retrieval. The model would answer the questions only based on the knowledge which it had absorbed through its original pre-training and had been thus encoded in its weights. The prompt template from section 3.4 was therefore also slightly altered to the following:

```
Use your knowledge to answer the question at the end. Keep the answer very
brief. If you don't know the answer reply "I don't know".
{context}
Question: {question}
Answer:
```

The second experiment was the same pipeline which is the subject of this thesis, as explained in full in Chapter 4. Lastly, the third experiment followed the pipeline in most regards, however the triples extraction step was not included. This means that the vectorization into embeddings as well as the RAG and inclusion of relevant context into the prompt was performed on the chunks of the original unaltered text as described in the third paragraph of section 3.2.

The problem of evaluating and quantifying RAG pipeline performance can be quite challenging due to open ended nature of LLM text generation. The framework used for the evaluation was the python library *ragas*, created by the Exploding Gradients company and introduced by them in collaboration with researchers at Cardiff University in the September

2023 paper *RAGAS: Automated Evaluation of Retrieval Augmented Generation [58]*. The RAGAS (Retrieval Augmented Generation Assessment) framework offers a set of metrics that cover various dimensions of RAG performance such as faithfulness, answer relevance, context precision, context relevance, context recall, and semantic similarity. These metrics are designed to provide insights into the quality of the generated responses, the relevance of the information retrieved, and the accuracy of the answers in relation to the provided context. Summarizing the paper's section on evaluation strategies, the three metrics that the authors consider to be of 'central importance' are the following:

- **Faithfulness**: how grounded the answer is with regard to the provided context
- **Answer relevance**: to what extent does the answer in fact address the question posed to the model
- **Context relevance**: to what extent is the provided context limited to information related to the question asked

In addition to these three metrics, **Answer Correctness** was also included, which is an evaluation of the alignment between the generated answer and the ground truth.

Since the first experiment did not include a RAG step, it would not be possible to evaluate it with the same metrics from the RAGAS framework. Instead, as a heuristic of the overall validity for the model's responses, the **cosine similarity** between the embeddings of the model response and the ground truth as taken from the original dataset was calculated for all three experiments. Table 1 below shows the final results.

A limitation of the framework is that the automated measurement of these metrics relies itself on the use of a LLM, which is passed specific prompts created by the researchers. In addition, the *ragas* library currently lacks full support for the use of arbitrary LLMs, with several metrics being accessible exclusively through the use of OpenAI's proprietary, closed-source GPT-3.5 and GPT-4 models and their corresponding API, which is offered as a paid service with the use of an OpenAI account. The evaluation of the output of the two RAG experiments was done using the GPT-3.5 model and incurred a total cost of 36€ in OpenAI credits.

	Cosine similarity	**Answer Correctness**	**Faithfulness**	**Answer Relevance**	**Context Relevance**
Experiment 1 (unaided LLM)	0.384	N/A	N/A	N/A	N/A
Experiment 2 (RAG on KG triples pipeline)	0.417	0.542	0.334	0.616	0.05
Experiment 3 (RAG on original text)	0.480	0.530	0.569	0.742	0.154

Table 2: The evaluation metrics for all three experiments

5.3 Quantitative analysis

As can be seen from the above data, the pipeline based on the triples extracted by the T5 model appears to be an improvement compared to the unaided LLM, however without reaching the performance of classic RAG in most metrics. Considering each metric individually:

- **Cosine similarity:** This is the only value which can be compared between all three experiments. It clearly shows that the embeddings of the answers generated by the KG-RAG model were more similar to the embeddings of the ground truth than those of the unaided LLM. This is used here as a rough heuristic for how close the LLM got to the actual answer. It can also be seen however that the simple RAG has a higher similarity than both by a fair margin.
- **Answer Correctness:** The answer correctness is slightly above 0.5 for both RAG pipelines, indicating that they are almost as likely to give an incorrect answer as they are to give the right one. To what extent that is the result of the RAG process itself could perhaps be hinted at by the three fundamental RAG metrics to follow.
- **Faithfulness:** This is the metric with the greatest deviation between the two RAG experiments in absolute value. Denoting how faithful the answer is to the provided context, one possible interpretation is that the LLM is more likely to use the RAG context when it is in the form of natural language. This could then be ameliorated by fine-tuning a model to take into account triples specifically.
- **Answer Relevance:** This metric, showing the relevance of the answer to the question, is also higher for the third experiment than for the second, while the difference is not as great as for faithfulness. It is also the highest value among all the metrics for both pipelines. Keeping that in mind, it could indicate that the model above all made an effort to answer the question, even when the other metrics suggest it did not quite have the resources for that and often ultimately failed to do so.
- **Context Relevance:** This metric has an exceptionally low value, with the evaluation model giving most samples less than 0.2 and very often a 0. This could be at least partially explained by the MMR algorithm. In cases where the relevant info only be found in at most on place in the source documents, the algorithm’s tendency to promote diversity in the results could mean that the other four out of five returned candidates are considered irrelevant by this metric. That said, it can still be observed that the simple RAG model had a relative value 3 times greater than that of the KG-RAG one.

Commentary on these results is included in the next chapter, along with a subjective assessment of the model outputs.

5.4 Qualitative analysis

A few interesting examples from the model generation are provided below. It should be noted that the Natural Questions dataset typically provides more than one acceptable answer as the ground truth. Only the most relevant answer has been included for these examples.

Question: who wrote the song photograph by ringo starr

Ground truth: George Harrison

Unaided LLM answer: I don't know.

LLM w/ RAG answer: Ringo Starr and George Harrison wrote the song "Photograph."

LLM w/ KG-RAG answer: George Harrison

Comments: In this example, we can see how RAG enabled the LLM to answer a question that it didn't know, or wasn't confident about knowing, the answer to. Both the raw text RAG and the triples-based RAG provided the relevant context to give the right answer.

Question: mandatory elements in the definition of an adverse event (ae)

Ground truth: any unfavourable and unintended sign (including an abnormal laboratory finding) , symptom , or disease temporally associated with the use of a medicinal (investigational) product , whether or not related to the medicinal (investigational) product

Unaided LLM answer: Mandatory elements in the definition of an adverse event (AE) include:

* Patient's identity and demographic information

* Date and duration of exposure to the suspected cause (e.g., drug, procedure)

* Nature and severity of the AE

* Whether the AE was reported to the regulatory authorities or not.

LLM w/ RAG answer: The mandatory elements in the definition of an adverse event (AE) are:

1. Untoward medical occurrence

2. In a patient or clinical investigation subject

3. Administered a pharmaceutical product

4. Does not necessarily have a causal relationship with this treatment

LLM w/ KG-RAG answer: Adverse event (AE) definition:

* Unfavourable and unintended sign or event.

Comments: Here we see that both RAG methods helped the model to come up with a correct answer, with the KG-RAG answer being briefer but closer to the ground truth.

Question: where was the world economic forum held this year

Ground truth: Davos

Unaided LLM answer: The World Economic Forum (WEF) was held this year in Davos, Switzerland.

LLM w/ RAG answer: Singapore

LLM w/ KG-RAG answer: Davos.

Comments: Here we can see a limitation of the Natural Questions dataset. Questions with subjective or ambiguous terms such as "this year" have not been filtered out, meaning that sometimes an answer can be factually correct from the perspective of a specific time period but still not match the ground truth. In this case, the RAG-LLM got confused after encountering a snippet about the 2021 World Economic Forum being held in Singapore, instead of its usual host city of Davos.

Question: star wars the clone wars anakin voice actor

Ground truth: Matt Lanter

Unaided LLM answer: Ahmed Best

LLM w/ RAG answer: James Arnold Taylor

LLM w/ KG-RAG answer: The voice actor for Anakin in Star Wars: The Clone Wars is James Arnold Taylor.

Comments: In this case, the unaided LLM hallucinated a totally incorrect answer, despite the prompt encouraging it to admit when it doesn't know something. The RAG-assisted models relied on the context and ended up with a more plausible, albeit still incorrect, answer: the name of the actor who played the same role in the films.

Question: who has made the most premier league appearances

Ground truth: Gareth Barry

Unaided LLM answer: I don't know.

LLM w/ RAG answer: Gareth Barry with 653 Premier League appearances.

LLM w/ KG-RAG answer: Mikel_Michael

Comments: In this case, the KG-RAG-LLM was misdirected by the seemingly malformed triple Mikel_Michael | appearanceIn | 22_(Premier_League_season)". It should be noted that an internet search reveals the only person with the name Mikel associated with the English Premier League is Mikel Arteta, manager of Arsenal F.C during the 2022-2023 season.

Question: a good that can be used in place of another good

Ground truth: A substitute good

Unaided LLM answer: A substitute good is a good that can be used in place of another good.

LLM w/ RAG answer: Substitute good.

LLM w/ KG-RAG answer: A complementary good.

Comments: This is an interesting example. The phrase ‘complementary good’, and in fact the word ‘complementary’ in general, does not appear anywhere among the retrieved triples. Instead, one of them even provides the correct answer: "Two_goods | substitutes | if_products_could_be_used_for_the_same_purpose". It appears that in this case, the model’s tendency to hallucinate overpowered the RAG technique.

Question: how old was sasuke when his clan died

Ground truth: seven

Unaided LLM answer: Sasuke was 9 years old when his clan died.

LLM w/ RAG answer: Sasuke was 7 years old when his clan died.

LLM w/ KG-RAG answer: Sasuke was born in 2007, so his clan died when he was around 6 years old.

Comments: In the absence of better data, the KG-RAG-LLM interpreted a triple about a ‘startYear’ of 2007 as the character’s date of birth.

Question: who wrote the book my country my life

Ground truth: L.K. Advani

Unaided LLM answer: Sure, I'd be happy to help! The book "My Country, My Life" was written by Winston Churchill.

LLM w/ RAG answer: L. K. Advani

LLM w/ KG-RAG answer: The book "My Country, My Life" was written by Ronald Reagan.

Comments: The correct context was given to the RAG-LLM, enabling it to answer this question. Instead, the most relevant triple that was passed to KG-RAG-LLM was about the authorship of the book “An American Life”. A manual search of the stored triples revealed that there was one encoding the relationship “My_Country_My_Life | author | L._K._Advani”, but it was not returned by the MMR search.

Question: who does eric end up with in gossip girl

Ground truth: Jenny

Unaided LLM answer: Eric ends up with Blair Waldorf in Gossip Girl.

LLM w/ RAG answer: Eric ends up with Blair Waldorf in Gossip Girl.

LLM w/ KG-RAG answer: Eric ends up with Blair Waldorf in Gossip Girl.

Comments: While, in Llama’s defense, the correct answer does not appear anywhere in the retrieved context, neither does the one it replied with. One could say that the model, like so many human fans, has its favorite couple.

Question: where did the saying monkey's uncle come from

Ground truth: Scopes Trial in the United States

Unaided LLM answer: I don't know the origin of the saying "monkey's uncle." It is a common idiom used to express surprise or disbelief, but its exact origins are unclear.

LLM w/ RAG answer: The origin of the saying "monkey's uncle" is unclear, but it is believed to have originated in Japan as a play on words with the phrase "see not, hear not, speak not" (見ざる、聞かざる、言わざる), which is often shortened to "monkey's uncle" in English. The phrase was later popularized in The Simpsons episode "Missionary: Impossible" and has since been used as a catchphrase in various contexts.

LLM w/ KG-RAG answer: United States.

Comments: In this rather humorous case, the RAG-LM got confused by the retrieved context containing snippets from the articles “Three wise monkeys” and “Bob’s your uncle”, merging them together into a wild hallucination. The KG-RAG-LLM chanced upon the correct answer based on a triple concerning a 1965 American comedy film titled “The Monkey's Uncle”.

---

6 CHAPTER 6: Discussion

The results of the metrics indicate that the methodology outlined in this thesis does not improve upon the performance of ordinary RAG using natural language text. Manual evaluation of the output from all three experiments shows that the unaided LLM is quite prone to hallucination, which is somewhat ameliorated in the two RAG experiments but still not completely eliminated. Even when the model is requested to answer a question based on the context provided, and reply with ‘I don’t know’ when the answer is unknown, it will still sometimes invent a response.

One possible use case for the triples pipeline would be the reduction of the space needed to store the documents, since converting them to triples reduces their size by about 24%. This could be further improved in production systems through a process of pruning, by dropping duplicate or highly similar triples. That said, it should also be noted that the space requirements for storing space would be considered relatively low by the standards of modern systems, with the chunks representing the full text of more than 3800 Wikipedia articles taking less than 130MB before any compression.

A limitation worth considering in the proposed methodology is that while the triples created by the extractor model were semantic triples as are typically used in knowledge graphs, the resulting graph in this case would be very sparsely connected since every triple was completely independent of all others. A second processing step to find triples containing the same entity as a subject or object and identifying those nodes with each other would fill out the missing connections and allow the QA system to take better advantage of the natural strengths of KGs, for example by adding to the context not only the triples returned by the RAG but also any triples defined by taking the neighbors of the relevant triples’ nodes.

Greater performance could also conceivably be achieved through the replacement of the models used. Given sufficient computational resources, a full implementation of IBM’s Grapher model (described in section 3.7) could be trained for the task of KG creation and integrated into the RAG pipeline proposed in this paper. This is a promising direction for future research. Similarly, the LLM used here had evident limitations in its utility for QA, including its apparent difficulty with following the given instructions to limit its responses to the provided context. This could potentially be ameliorated through more refined prompt engineering, fine-tuning a model for the QA task, or using a larger model.

7 CONCLUSIONS

This thesis aimed to explore the integration of Retrieval-Augmented Generation (RAG) with Knowledge Graph (KG) triples to enhance the performance of Large Language Models (LLMs) in Question Answering (QA). Through a specially designed pipeline, this project aimed to utilize the structured representation of KG triples in order to improve the contextual understanding and precision of LLMs, thereby addressing the challenges of factual accuracy and hallucinations often encountered in QA tasks.

The findings of this research offer mixed outcomes. On one hand, the development and evaluation of the KG-LLM-QA pipeline demonstrated a notable improvement in the quality of answers generated by LLMs, highlighting the potential of integrating structured knowledge in the form of KG triples with the dynamic and nuanced capabilities of LLMs. On the other hand, when compared to conventional RAG applications, the results of the proposed methodology did not outperform existing solutions. This outcome underscores the complexity of enhancing LLM performance for QA tasks and the challenges inherent in blending structured and unstructured knowledge sources.

Key Contributions

- **Innovative Pipeline Development:** The creation of a novel pipeline that converts natural language documents into KG triples and utilizes these triples to support LLMs in generating more accurate and grounded answers represents a significant technical achievement. This approach not only showcased the feasibility of integrating KGs with LLMs but also demonstrated a practical application of this integration in improving QA performance.
- **Empirical Insights:** The systematic evaluation of the KG-LLM-QA system provided valuable empirical insights into the strengths and limitations of leveraging KG triples in augmenting LLMs for QA tasks. The comparative analysis with conventional RAG applications enriched the understanding of the potential and constraints of retrieval-augmented approaches in natural language processing.
- **Resource Efficiency:** The exploration of a triples extraction process that reduces the storage requirements for supporting documents by approximately 24% presents an intriguing avenue for optimizing the storage and retrieval of knowledge in QA systems. This finding hints at the potential for more resource-efficient implementations of knowledge-based augmentation for LLMs.

Limitations and Future Directions

- **Improving Connectivity in KGs:** The sparse connectivity observed in the generated KGs suggests a need for further refinement in how KG triples are processed and integrated. Future work could explore algorithms for enhancing the interconnectedness of KG triples, thereby enriching the context available for QA tasks.

- **Optimizing Triple Storage and Retrieval:** The efficiency of storing and retrieving KG triples presents another area for improvement. Investigating more advanced compression and deduplication techniques could significantly reduce the storage footprint and enhance the speed of retrieval, contributing to more scalable and responsive QA systems.
- **Enhanced Understanding of Hallucination Mitigation:** Despite some reduction in hallucinations, the issue remains a challenge. Future research could delve deeper into the mechanisms by which structured knowledge influences LLM outputs and explore novel approaches to further mitigate hallucinations in LLM-generated content.
- **Comparative Studies with Emerging Technologies:** As the field of natural language processing continues to evolve rapidly, comparing the KG-LLM-QA approach with emerging technologies and methodologies could yield insights into new avenues for enhancing LLM performance in QA and beyond.

Concluding Thoughts

This thesis represents a step forward in the quest to harness the synergies between structured knowledge and the generative capabilities of LLMs. While the journey revealed both challenges and opportunities, the insights gained contribute to the broader discourse on enhancing the reliability, accuracy, and efficiency of QA systems. Looking to the future, the integration of KGs and LLMs holds promise for not only advancing the state of QA technologies but also for broadening the applicability of AI in accessing and leveraging knowledge across diverse domains.

Bibliography – References – Online sources

[1] R. Rosenfeld, “Two decades of statistical language modeling: where do we go from here?,” *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, Aug. 2000, doi: 10.1109/5.880083.

[2] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Deep neural network language models,” in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, in WLM ’12. USA: Association for Computational Linguistics, Jun. 2012, pp. 20–28.

[3] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, “Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems.” arXiv, Aug. 26, 2015. doi: 10.48550/arXiv.1508.01745.

[4] Y. Goldberg, “A Primer on Neural Network Models for Natural Language Processing,” *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, Nov. 2016, doi: 10.1613/jair.4992.

[5] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Jan. 27, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[6] S. Zagoruyko and N. Komodakis, “Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer.” arXiv, Feb. 12, 2017. doi: 10.48550/arXiv.1612.03928.

[7] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate.” arXiv, May 19, 2016. doi: 10.48550/arXiv.1409.0473.

[8] M. Geva, R. Schuster, J. Berant, and O. Levy, “Transformer Feed-Forward Layers Are Key-Value Memories.” arXiv, Sep. 05, 2021. doi: 10.48550/arXiv.2012.14913.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” 2018.

[11] C. Raffel *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” arXiv, Sep. 19, 2023. doi: 10.48550/arXiv.1910.10683.

[12] T. Lin, Y. Wang, X. Liu, and X. Qiu, “A survey of transformers,” *AI Open*, vol. 3, pp. 111–132, Jan. 2022, doi: 10.1016/j.aiopen.2022.10.001.

[13] W. X. Zhao *et al.*, “A Survey of Large Language Models.” arXiv, Nov. 24, 2023. doi: 10.48550/arXiv.2303.18223.

[14] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” 2019. Accessed: Jan. 27, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe

[15] H. Ye, T. Liu, A. Zhang, W. Hua, and W. Jia, “Cognitive Mirage: A Review of Hallucinations in Large Language Models.” arXiv, Sep. 13, 2023. doi: 10.48550/arXiv.2309.06794.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples.” arXiv, Mar. 20, 2015. doi: 10.48550/arXiv.1412.6572.

[17] S. M. T. I. Tonmoy *et al.*, "A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models." arXiv, Jan. 08, 2024. doi: 10.48550/arXiv.2401.01313.

[18] Z. Jiang, J. Araki, H. Ding, and G. Neubig, "How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 962–977, Sep. 2021, doi: 10.1162/tacl_a_00407.

[19] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 9459–9474. Accessed: Jan. 23, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[20] Y. Gao *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey." arXiv, Jan. 04, 2024. doi: 10.48550/arXiv.2312.10997.

[21] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A Survey on Retrieval-Augmented Text Generation." arXiv, Feb. 13, 2022. doi: 10.48550/arXiv.2202.01110.

[22] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking Large Language Models in Retrieval-Augmented Generation." arXiv, Dec. 20, 2023. doi: 10.48550/arXiv.2309.01431.

[23] V. K. Chaudhri *et al.*, "Knowledge graphs: Introduction, history, and perspectives," *AI Magazine*, vol. 43, no. 1, pp. 17–29, 2022, doi: 10.1002/aaai.12033.

[24] X. Zou, "A Survey on Application of Knowledge Graph," *J. Phys.: Conf. Ser.*, vol. 1487, no. 1, p. 012016, Mar. 2020, doi: 10.1088/1742-6596/1487/1/012016.

[25] Z. Ji *et al.*, "RHO: Reducing Hallucination in Open-domain Dialogues with Knowledge Grounding," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 4504–4522. doi: 10.18653/v1/2023.findings-acl.275.

[26] D. Su *et al.*, "Generalizing Question Answering System with Pre-trained Language Model Fine-tuning," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, A. Fisch, A. Talmor, R. Jia, M. Seo, E. Choi, and D. Chen, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 203–211. doi: 10.18653/v1/D19-5827.

[27] F. F. Bayat *et al.*, "FLEEK: Factual Error Detection and Correction with Evidence Retrieved from External Knowledge." arXiv, Oct. 25, 2023. doi: 10.48550/arXiv.2310.17119.

[28] M. Namazifar, A. Papangelis, G. Tur, and D. Hakkani-Tür, "Language Model is all You Need: Natural Language Understanding as Question Answering," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 7803–7807. doi: 10.1109/ICASSP39728.2021.9413810.

[29] I. Melnyk, P. Dognin, and P. Das, "Knowledge Graph Generation From Text." arXiv, Nov. 18, 2022. doi: 10.48550/arXiv.2211.10511.

[30] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering." arXiv, Dec. 12, 2022. doi: 10.48550/arXiv.2104.06378.

[31] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge." arXiv, Mar. 15, 2019. doi: 10.48550/arXiv.1811.00937.

[32] D. Chen and W. Yih, "Open-Domain Question Answering," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, A. Savary and Y. Zhang, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 34–37. doi: 10.18653/v1/2020.acl-tutorials.8.

[33] D. Jin, E. Pan, N. Oufattole, W.-H. Weng, H. Fang, and P. Szolovits, “What Disease Does This Patient Have? A Large-Scale Open Domain Question Answering Dataset from Medical Exams,” *Applied Sciences*, vol. 11, no. 14, Art. no. 14, Jan. 2021, doi: 10.3390/app11146421.

[34] R. Speer, J. Chin, and C. Havasi, “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Art. no. 1, Feb. 2017, doi: 10.1609/aaai.v31i1.11164.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks.” arXiv, Feb. 04, 2018. doi: 10.48550/arXiv.1710.10903.

[36] X. Zhang *et al.*, “GreaseLM: Graph REASoning Enhanced Language Models for Question Answering.” arXiv, Jan. 21, 2022. doi: 10.48550/arXiv.2201.08860.

[37] Y. Ding, A. Poudel, Q. Zeng, T. Weninger, B. Veeramani, and S. Bhattacharya, “EntGPT: Linking Generative Large Language Models with Knowledge Bases.” arXiv, Feb. 09, 2024. doi: 10.48550/arXiv.2402.06738.

[38] H. W. Chung *et al.*, “Scaling Instruction-Finetuned Language Models.” arXiv, Dec. 06, 2022. doi: 10.48550/arXiv.2210.11416.

[39] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, “Creating Training Corpora for NLG Micro-Planners,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 179–188. doi: 10.18653/v1/P17-1017.

[40] E. J. Hu *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models.” arXiv, Oct. 16, 2021. doi: 10.48550/arXiv.2106.09685.

[41] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, “Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning.” arXiv, Dec. 22, 2020. doi: 10.48550/arXiv.2012.13255.

[42] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization.” arXiv, Jan. 04, 2019. doi: 10.48550/arXiv.1711.05101.

[43] “PyTorch,” PyTorch. Accessed: Feb. 12, 2024. [Online]. Available: https://pytorch.org/

[44] “PyTorch Lightning.” Accessed: Feb. 12, 2024. [Online]. Available: https://lightning.ai/pytorch-lightning

[45] “Hugging Face – The AI community building the future.” Accessed: Feb. 12, 2024. [Online]. Available: https://huggingface.co/

[46] “IBM/Grapher.” International Business Machines, Feb. 08, 2024. Accessed: Feb. 11, 2024. [Online]. Available: https://github.com/IBM/Grapher

[47] T. Kwiatkowski *et al.*, “Natural Questions: A Benchmark for Question Answering Research,” *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, Aug. 2019, doi: 10.1162/tacl_a_00276.

[48] “LangChain.” Accessed: Feb. 13, 2024. [Online]. Available: https://www.langchain.com/

[49] “CoreferenceResolver · spaCy API Documentation,” CoreferenceResolver. Accessed: Feb. 13, 2024. [Online]. Available: https://spacy.io/api/coref

[50] “sentence-transformers/all-mpnet-base-v2 · Hugging Face.” Accessed: Feb. 12, 2024. [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[51] “ Home | Chroma.” Accessed: Feb. 12, 2024. [Online]. Available: https://www.trychroma.com/

[52] J. Carbonell and J. Goldstein, “The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries,” *SIGIR Forum*, vol. 51, no. 2, pp. 209–210, Aug. 2017, doi: 10.1145/3130348.3130369.

[53] H. Touvron *et al.*, “LLaMA: Open and Efficient Foundation Language Models.” arXiv, Feb. 27, 2023. doi: 10.48550/arXiv.2302.13971.

[54] H. Touvron *et al.*, “Llama 2: Open Foundation and Fine-Tuned Chat Models.” arXiv, Jul. 19, 2023. doi: 10.48550/arXiv.2307.09288.

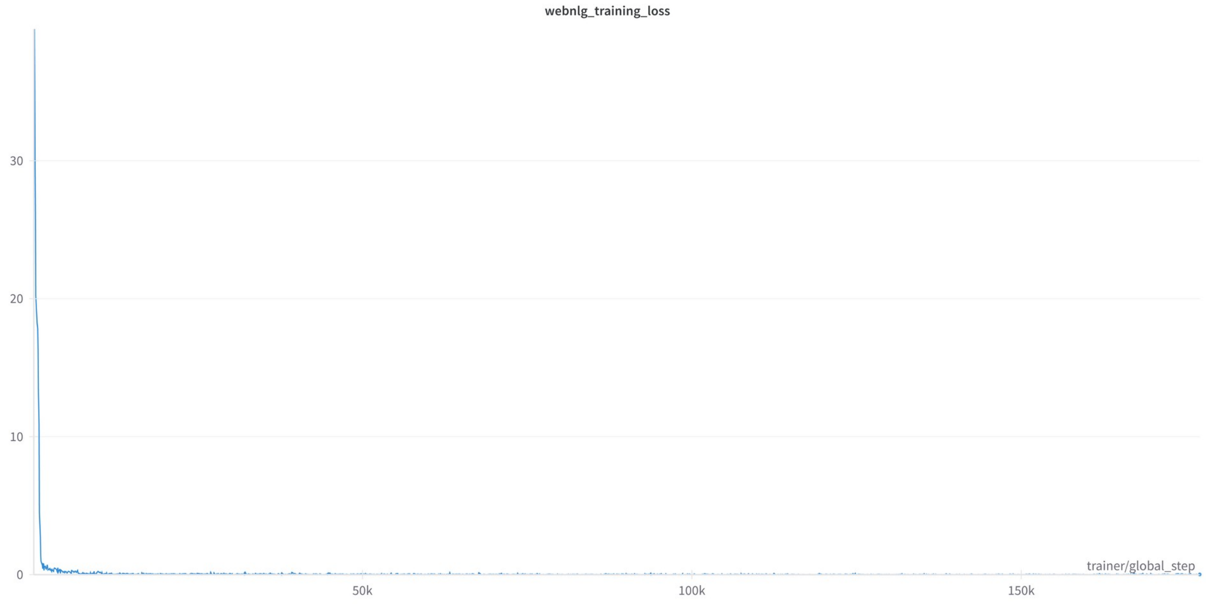
[55] “WebNLG/WebNLG-Text-to-triples.” WebNLG, Sep. 21, 2023. Accessed: Feb. 14, 2024. [Online]. Available: https://github.com/WebNLG/WebNLG-Text-to-triples

[56] “WebNLG2020 Challenge final results.” Accessed: Feb. 14, 2024. [Online]. Available: https://beng.dice-research.org/gerbil/webnlg2020results

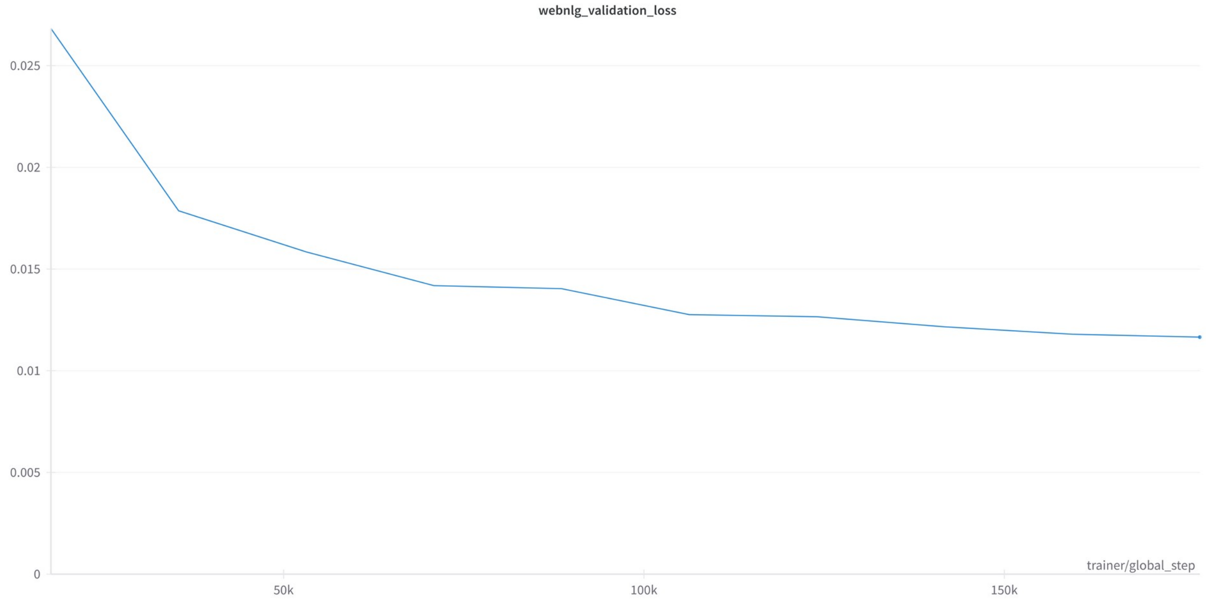
[57] J. Liao, X. Su, L. Jiang, K.-C. Li, T.-H. Weng, and S. Bhalla, “Performance of representation fusion model for entity and relationship extraction within unstructured text,” *J Supercomput*, vol. 79, no. 18, pp. 20826–20844, Dec. 2023, doi: 10.1007/s11227-023-05442-6.

[58] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “RAGAS: Automated Evaluation of Retrieval Augmented Generation.” arXiv, Sep. 26, 2023. doi: 10.48550/arXiv.2309.15217.

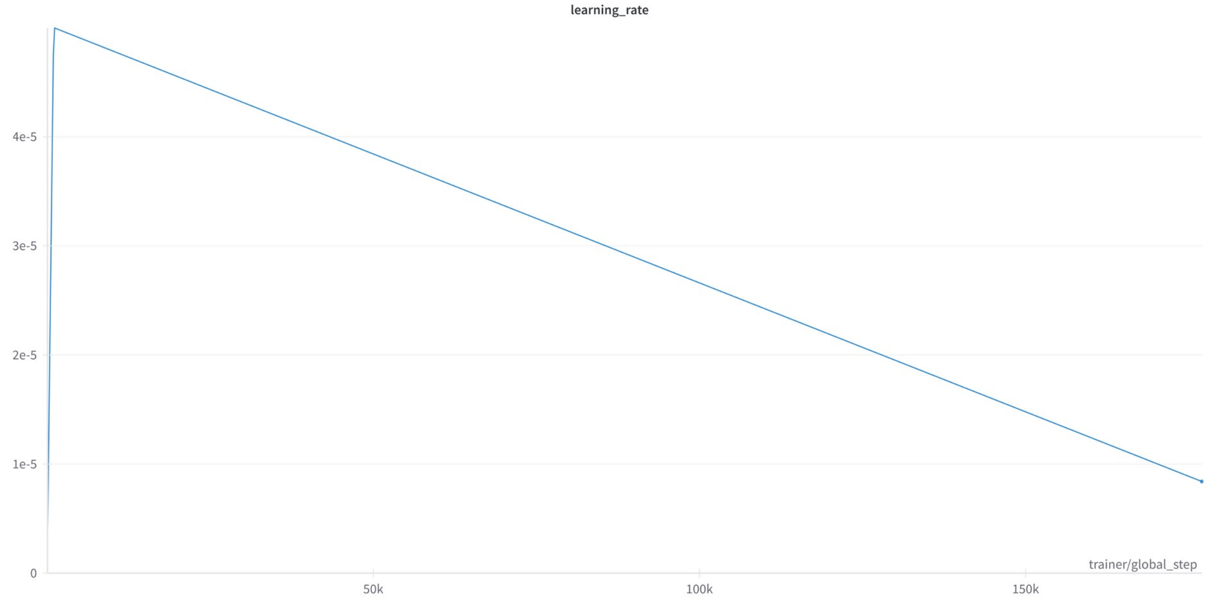
Appendix A: T5 training graphics



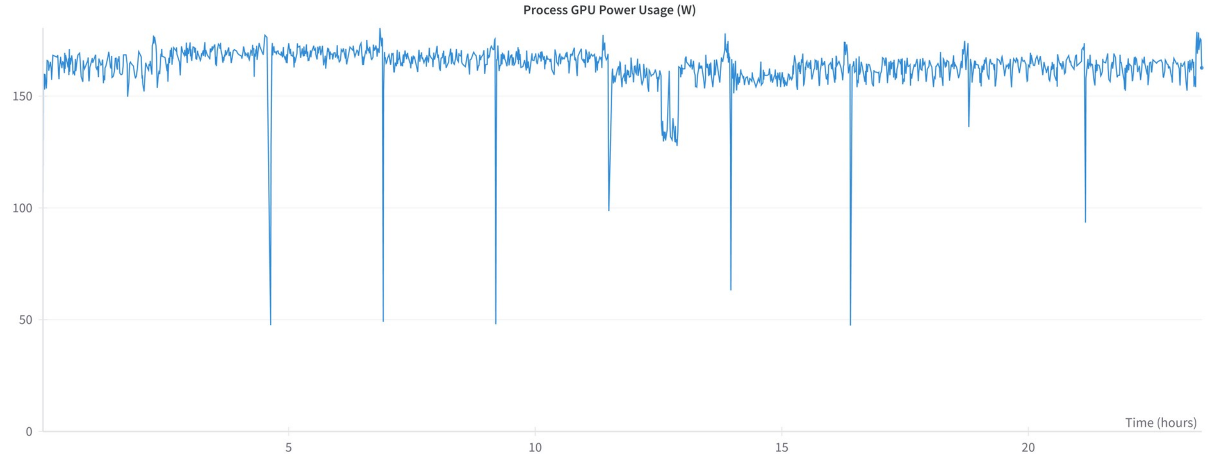
Training loss over number of training steps



Validation loss over number of training steps



Learning rate over number of training steps



GPU power usage in Watts over time

Appendix B: WebNLG2020 evaluation results

```
{
    "Total_scores": {
        "Ent_type": {
            "Correct": 18988,
            "Incorrect": 70,
            "Partial": 0,
            "Missed": 1870,
            "Spurious": 1718,
            "Possible": 20928,
            "Actual": 20776,
            "Precision": 0.8792026136317531,
            "Recall": 0.8945967054813974,
            "F1": 0.8853388958543744
        },
        "Partial": {
            "Correct": 18114,
            "Incorrect": 0,
            "Partial": 944,
            "Missed": 1870,
            "Spurious": 1718,
            "Possible": 20928,
            "Actual": 20776,
            "Precision": 0.8633559598910352,
            "Recall": 0.8761005396194264,
            "F1": 0.8684439061995154
        },
        "Strict": {
            "Correct": 18073,
            "Incorrect": 985,
            "Partial": 0,
            "Missed": 1870,
            "Spurious": 1718,
            "Possible": 20928,
            "Actual": 20776,
            "Precision": 0.8426550980654929,
            "Recall": 0.8524448546814352,
            "F1": 0.846569721420616
        },
        "Exact": {
            "Correct": 18114,
            "Incorrect": 944,
            "Partial": 0,
            "Missed": 1870,
            "Spurious": 1718,
            "Possible": 20928,
            "Actual": 20776,
            "Precision": 0.8444703967495788,
            "Recall": 0.854350089936571,
            "F1": 0.8484230008370878
        }
    },
    "Scores_per_tag": {
        "Subjects": {
            "Ent_type": {
                "Correct": 6450,
                "Incorrect": 29,
                "Partial": 0,
                "Missed": 488,
                "Spurious": 536,
                "Possible": 6967,
                "Actual": 7015,
                "Precision": 0.8973593811281968,
                "Recall": 0.9129508662311844,
                "F1": 0.902108663213463
            },
            "Partial": {
                "Correct": 6225,
                "Incorrect": 0,
                "Partial": 254,
                "Missed": 488,
                "Spurious": 536,
                "Possible": 6967,
                "Actual": 7015,
                "Precision": 0.8910203743525243,
                "Recall": 0.8996733882419767,
                "F1": 0.8936753627892509
            },
            "Strict": {
                "Correct": 6213,
                "Incorrect": 266,
                "Partial": 0,
                "Missed": 488,
                "Spurious": 536,
                "Possible": 6967,
                "Actual": 7015,
                "Precision": 0.8800662014308706,
                "Recall": 0.8807867083214995,
                "F1": 0.8802955509572947
            },
            "Exact": {
                "Correct": 6225,
                "Incorrect": 254,
                "Partial": 0,
                "Missed": 488,
                "Spurious": 536,
                "Possible": 6967,
                "Actual": 7015,
                "Precision": 0.8813442474405269,
                "Recall": 0.8824197671116161,
                "F1": 0.8816919345604378
            }
        },
        "Predicates": {
            "Ent_type": {
                "Correct": 6205,
                "Incorrect": 4,
                "Partial": 0,
                "Missed": 750,
                "Spurious": 608,
                "Possible": 6959,
                "Actual": 6817,
                "Precision": 0.8734320268863013,
                "Recall": 0.8791536495313831,
                "F1": 0.8752705986304395
            },
            "Partial": {
                "Correct": 6095,
                "Incorrect": 0,
                "Partial": 114,
                "Missed": 750,
                "Spurious": 608,
                "Possible": 6959,
                "Actual": 6817,
                "Precision": 0.8689559074126668,
                "Recall": 0.8721243964782732,
                "F1": 0.8699887973744833
            },
            "Strict": {
                "Correct": 6095,
                "Incorrect": 114,
                "Partial": 0,
                "Missed": 750,
                "Spurious": 608,
                "Possible": 6959,
                "Actual": 6817,
                "Precision": 0.8641957777146644,
                "Recall": 0.8645271229764272,
                "F1": 0.8643283158193695
            },
            "Exact": {
                "Correct": 6095,
                "Incorrect": 114,
                "Partial": 0,
                "Missed": 750,
                "Spurious": 608,
                "Possible": 6959,
                "Actual": 6817,
                "Precision": 0.8641957777146644,
                "Recall": 0.8645271229764272,
                "F1": 0.8643283158193695
            }
        },
        "Objects": {
            "Ent_type": {
                "Correct": 6333,
                "Incorrect": 37,
                "Partial": 0,
                "Missed": 632,
                "Spurious": 574,
                "Possible": 7002,
                "Actual": 6944,
                "Precision": 0.8831997809063984,
                "Recall": 0.8913660891792105,
                "F1": 0.8857639875035501
            },
            "Partial": {
                "Correct": 5794,
                "Incorrect": 0,
                "Partial": 576,
                "Missed": 632,
                "Spurious": 574,
                "Possible": 7002,
                "Actual": 6944,
                "Precision": 0.8517639063577718,
                "Recall": 0.8576043737574552,
                "F1": 0.8536649152702831
            },
            "Strict": {
                "Correct": 5765,
                "Incorrect": 605,
                "Partial": 0,
                "Missed": 632,
                "Spurious": 574,
                "Possible": 7002,
                "Actual": 6944,
                "Precision": 0.8132751112373379,
                "Recall": 0.8146833285998296,
                "F1": 0.8138218309192464
            },
            "Exact": {
                "Correct": 5794,
                "Incorrect": 576,
                "Partial": 0,
                "Missed": 632,
                "Spurious": 574,
                "Possible": 7002,
                "Actual": 6944,
                "Precision": 0.8164228912240841,
                "Recall": 0.8187304742970747,
                "F1": 0.8172630881378397
            }
        }
    },
    "Exact_match": {
        "Precision": 0.43361105963818747,
        "Recall": 0.35597241498005566,
        "F1": 0.38072064047307497
    }
}
```