

**TRAINING AND EVALUATING THE USE OF LARGE
LANGUAGE MODELS (LLMs) IN THE DOMAIN OF
CANADIAN NUCLEAR INDUSTRY**

by

Muhammad Saleh Anwar

Submitted in partial fulfilment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
June 2025

Dedication

This thesis is dedicated to those who have been my pillars of strength and inspiration. To my late father, who always emphasized the importance of higher education and developing deep expertise in a field. I would not be here without his guidance, and this work is a tribute to his memory. To my mother, who inspires me every day with her unwavering support and remarkable resilience. To my wonderful wife and child, who are my rock during testing times. Your love and encouragement motivate me to persevere through any challenge. And finally, to all my family and friends, whose companionship and guidance have been a constant source of strength in this pursuit.

Table of Contents

LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT.....	vii
LIST OF ABBREVIATIONS USED.....	viii
ACKNOWLEDGEMENT.....	x
CHAPTER 1. INTRODUCTION.....	1
1.1 MOTIVATION.....	1
1.2 BACKGROUND AND RELATED WORK	3
1.3 CONTRIBUTIONS TO MANUSCRIPT.....	6
CHAPTER 2. EVALUATING LLMS (CHATGPT) ON NUCLEAR DOMAIN SPECIFIC DATA.....	8
2.1 INTRODUCTION.....	8
2.2 PROBLEM STATEMENT	8
2.3 RETRIEVAL-AUGMENTED GENERATION (RAG) : A BACKGROUND.....	10
<i>Key Steps in the RAG Methodology.....</i>	<i>10</i>
<i>Advantages of RAG.....</i>	<i>11</i>
2.4 RELATED WORK / LITERATURE REVIEW	11
2.5 EVALUATION METHODOLOGY	12
<i>Evaluation Dataset.....</i>	<i>13</i>
2.6 ANSWER GENERATION WITHOUT RAG	14
2.7 RETRIEVAL AUGMENTED GENERATION	14
<i>Vector Index Creation.....</i>	<i>15</i>
<i>Search and Prompt Augmentation.....</i>	<i>15</i>
<i>Response Generation</i>	<i>16</i>
2.8 GENERATION OF ANSWERS	16
2.9 EVALUATION OF ANSWERS	18
<i>Human Evaluation:.....</i>	<i>18</i>
<i>LLM Based Evaluation:.....</i>	<i>18</i>
2.10 RESULTS	21
2.11 CONCLUSION.....	23
CHAPTER 3. UNLOCKING THE POTENTIAL OF LARGE LANGUAGE MODELS IN THE NUCLEAR INDUSTRY USING SYNTHETIC DATA.....	24
3.1 INTRODUCTION.....	24
3.2 OBSTACLES, CHALLENGES, AND THE PROMISE OF SYNTHETIC DATA.....	25
3.3 RELATED WORK / LITERATURE REVIEW	27
<i>Best Practices and Techniques</i>	<i>27</i>
<i>Cost-Effectiveness and Scalability.....</i>	<i>28</i>
<i>Domain-Specific Challenges in High-Security Areas.....</i>	<i>28</i>
<i>Synthesis and Future Directions.....</i>	<i>29</i>

3.4 SYNTHETIC DATA GENERATION PROCESS.....	30
3.4.1 <i>Data Extraction and Preprocessing</i>	32
3.4.2 <i>Embedding Generation and Clustering</i>	35
3.4.3 <i>Synthetic QnA Pair Generation</i>	40
3.5 EVALUATION OF SYNTHETIC QNA PAIRS	46
3.5.1 <i>Semantic Diversity</i>	47
3.5.2 <i>Document Relevance</i>	49
3.5.3 <i>Question Quality Metrics</i>	52
CHAPTER 4. TOWARDS SECURE AND PRIVATE LANGUAGE MODELS FOR NUCLEAR POWER PLANTS.....	55
4.1 INTRODUCTION.....	55
4.2 RELATED WORK AND LITERATURE REVIEW	56
4.3 DATA ACQUISITION AND PREPARATION.....	58
4.3.1 <i>Data Sources</i>	60
4.3.2 <i>Preprocessing and Cleaning</i>	61
4.3.3 <i>Tokenization and Embedding Strategy</i>	62
4.3.4 <i>Dataset Structure</i>	65
4.3.5 <i>Ethical and Legal Considerations</i>	66
4.4 MODEL ARCHITECTURE	66
4.4.1 <i>Overview and Influences</i>	66
4.4.2 <i>Core Building Blocks</i>	68
4.4.3 <i>Architectural Configuration</i>	75
4.4.4 <i>Hyperparameters: The “Dials” of Model Training</i>	77
4.4.5 <i>Domain-Specific Considerations</i>	78
4.5 TRAINING METHODOLOGY.....	79
4.5.1 <i>Next-Token Prediction as the Objective</i>	79
4.5.2 <i>Training Process</i>	80
4.5.3 <i>Optimization and Hyperparameters</i>	81
4.5.4 <i>Performance Monitoring and Preventing Overfitting</i>	82
4.5.5 <i>Single-GPU Constraints and Security Measures</i>	83
4.6 EVALUATION AND RESULTS	84
CHAPTER 5. CONCLUSION AND FUTURE WORK.....	86
5.1 THESIS SUMMARY	86
5.2 LIMITATIONS.....	86
5.3 FUTURE WORK.....	87
BIBLIOGRAPHY	88
APPENDIX A: CODE SNIPPETS.....	93

List of Tables

TABLE 3-1 : SYNTHETIC DATA AS A POTENTIAL SOLUTION	27
TABLE 3-2 : PROMPT DESIGN CONSIDERATIONS	40
TABLE 3-3 : ILLUSTRATIVE EXCERPTS FROM QNA GENERATION PROMPT	43
TABLE 3-4 - QNA PAIR EXAMPLES	46
TABLE 4-1 : TOPICS COVERED IN ESSENTIAL CANDU TEXTBOOK	60
TABLE 4-2 : PRE-PROCESSING STEPS	61
TABLE 4-3 : KEY LLM DESIGN PARAMETERS	67

List of Figures

FIGURE 2-1 : BOX PLOT COMPARING RESPONSES FOR CHATGPT3.5 WITH RAG AND WITHOUT RAG	21
FIGURE 2-2 : COLUMN PLOT COMPARING SCORES FOR CHATGPT3.5 WITH RAG AND WITHOUT RAG	22
FIGURE 3-1 OVERALL STRUCTURE OF SYNTHETIC QNA PAIRS PIPELINE	31
FIGURE 3-2 : ELBOW PLOT TO DETERMINE OPTIMAL NUMBER OF CLUSTERS.....	38
FIGURE 3-3 : T-SNE VISUALIZATION OF CLUSTERS	39
FIGURE 3-4 : T-SNE VISUALIZATION OF QUESTIONS ALONG WITH BENCHMARK FOR COMPARISON.....	48
FIGURE 3-5: HISTOGRAM FOR COSINE SIMILARITY BETWEEN QUESTION AND TEXTBOOK CHUNK	51
FIGURE 3-6 : WORD CLOUD OF GENERATED QUESTIONS	54
FIGURE 4-1 : TYPICAL LLM TRAINING STEPS	56
FIGURE 4-2 : DATA PREPARATION STEPS.....	59
FIGURE 4-3 : ILLUSTRATION OF TRANSFORMER ARCHITECTURE	68
FIGURE 4-4: TRAINING AND VALIDATION LOSS CURVES	83

Abstract

This thesis addresses the challenges of accuracy, reliability, data privacy, and resource constraints in applying Large Language Models (LLMs) to the Canadian nuclear industry. It presents a multi-faceted approach by evaluating existing models, developing synthetic data generation techniques, and training a secure, domain-specific LLM from scratch.

The research first demonstrates that while general-purpose LLMs are prone to factual inaccuracies on nuclear-specific topics, their reliability is significantly improved by integrating a Retrieval-Augmented Generation (RAG) framework. This approach enhances factual accuracy by grounding responses in verified, domain-specific documents.

To overcome data scarcity and confidentiality barriers, the thesis pioneers a methodology for generating synthetic, structured question-and-answer pairs from unstructured nuclear texts using LLMs. This scalable and privacy-preserving approach creates valuable, model-ready datasets for training and evaluation without exposing sensitive information.

Furthermore, the work validates the feasibility of developing a secure, private LLM from scratch. By training a compact model on a single GPU using the "Essential CANDU" textbook, it demonstrates a practical path for creating in-house models that mitigate cybersecurity risks and can learn specialized terminology within a resource-constrained and secure environment.

Collectively, this research provides a comprehensive framework for integrating LLM technology safely and effectively into the nuclear industry, establishing a foundation for advanced AI tools that enhance knowledge management and operational support.

List of Abbreviations Used

AI - Artificial Intelligence
API - Application Programming Interface
BPE - Byte-Pair Encoding
CANDU - CANada Deuterium Uranium
CNA - Canadian Nuclear Association
CNN - Convolutional Neural Network
CNS - Canadian Nuclear Society
CNSC - Canadian Nuclear Safety Commission
CSA - Canadian Standards Association
D2O - Heavy water
ECCS - Emergency Core Cooling System
FAISS - Facebook AI Similarity Search
FFN - Feed-Forward Network
GELU - Gaussian Error Linear Unit
GPT - Generative Pre-trained Transformer
LDA - Latent Dirichlet Allocation
LLM - Large Language Model
LOCA - Loss-of-coolant accident
ML - Machine Learning
MLOps - Machine Learning Operations
NER - Named Entity Recognition
NLP - Natural Language Processing
OPEX - Operating Experience
PHT - Primary Heat Transport
Pre-LN - Pre-Layer Normalization
PWR - Pressurized Water Reactor
Q&A / QnA - Question-and-Answer or Question-Answer
QKV - Query Key Value
RAG - Retrieval-Augmented Generation

RLHF - Reinforcement Learning from Human Feedback

RNN - Recurrent Neural Network

RRF - Reciprocal Rank Fusion

SCRs - Station Condition Records

SDS - Shutdown System

t-SNE - t-Distributed Stochastic Neighbor Embedding

TF-IDF - Term frequency-inverse document frequency

UNENE - University Network of Excellence in Nuclear Engineering

WCSS - Within-cluster sum of squares

Acknowledgement

I would like to thank my supervisor, Dr. Issam Hammad, for his support and valuable advice throughout this research; his guidance and insight have greatly contributed to the development of this thesis. I would also like to thank my committee members, Dr. Guy Kember and Dr. Kamal El-Sankary, for their thoughtful review and evaluation of this work. I also wish to express my gratitude to Mishca de Costa and Daniel Lau, for their great support and collaboration on the papers written as part of this research.

Additionally, I extend my gratitude to Ontario Power Generation, as the experience I gained there has played an integral role in shaping this study.

CHAPTER 1. INTRODUCTION

1.1 Motivation

The field of Natural Language Processing (NLP) has experienced significant advancements in recent years, largely initiated by the advent of sophisticated Large Language Models (LLMs). These models demonstrate notable capabilities in understanding and generating human language, offering considerable promise for tasks ranging from complex data analysis to creative writing. However, applying the LLM technology within specialized domains, such as the nuclear industry, presents a distinct set of challenges. The strict requirements for data confidentiality, the need for high accuracy and reliability, and the often resource-constrained operational environments necessitate a more tailored approach to LLM development and deployment. This thesis explores and seeks to advance methodologies that enable the secure, effective, and practical application of LLMs within the context of the nuclear sector.

The primary objective of my research is to investigate and develop strategies that address the inherent limitations of general-purpose LLMs when applied to nuclear domain-specific tasks, while respecting the important security and data privacy requirements of the industry. This work addresses the complexities of applying LLM technology by evaluating enhancements to existing models, developing new methods for specialized data creation for LLM's consumption, and training a domain-specific large language model from its foundational components. Each chapter of this thesis addresses a different facet, collectively contributing to a framework for more secure and reliable LLMs in nuclear applications.

Chapter 2 of this thesis begins by examining the performance of well-known LLMs, specifically ChatGPT, for question-and-answer (Q&A) tasks using specialized nuclear data [1]. Recognizing that general-purpose LLMs are susceptible to generating incorrect or 'hallucinated' information—a significant concern where accuracy is of utmost importance—this chapter investigates the efficacy of Retrieval Augmented Generation (RAG). The chapter covers ChatGPT's ability to answer domain-specific questions both

directly and when integrated into a RAG framework, employing human and LLM-based assessments. The findings presented highlight notable improvements in accuracy and contextual relevance achieved by incorporating a RAG pipeline, offering a pathway to more dependable LLM use in the nuclear field.

Addressing another important prerequisite for effective LLM utilization, Chapter 3 focuses on the challenge of data scarcity and the need for structured, model-ready datasets within the nuclear domain [2]. A substantial body of the industry's knowledge is embedded in unstructured text, which is not directly suitable for training or fine-tuning advanced LLM applications. This chapter explores the potential of synthetic data generation as a means to bridge this gap. The chapter covers techniques that leverage LLMs themselves to analyze nuclear-specific texts, extract key information, and generate good-quality, structured question-answer pairs. This approach offers a solution to data scarcity and privacy issues relevant to nuclear industry, thereby preparing the way for improved information retrieval and knowledge sharing.

Chapter 4 covers the work done for developing a domain-specific LLM tailored explicitly for nuclear applications [3]. This chapter describes the construction of a compact transformer-based model, trained from scratch using the publicly accessible "Essential CANDU" textbook. A key aspect of this work is its execution on a single GPU within a secure environment, demonstrating the feasibility of creating in-house LLM solutions that adhere to the stringent cybersecurity and data confidentiality standards of the nuclear industry. This work underscores the potential of custom-developed models and outlines future directions for enhancing their accuracy and readiness for real-world nuclear applications through richer corpora and refined training techniques.

These chapters outline a path towards the responsible and effective integration of LLM technology within the nuclear power sector. Through addressing challenges related to model reliability, data availability, secure training and deployment, this thesis contributes to the development of language models that can serve as useful, secure, and specialized assets. The methodologies explored and developed herein aim to enable nuclear organizations to leverage advancements in NLP while upholding the highest standards of

safety, security, and data integrity, ultimately fostering more informed decision-making and more streamlined operational support in this field.

1.2 Background and Related Work

Research has increasingly demonstrated the potential for Artificial Intelligence (AI) and Machine Learning (ML) to enhance operations at nuclear facilities. A significant body of work focuses on improving component integrity and fault prediction. For instance, ML has been applied to predict faults in the Primary Heat Transport (PHT) system of CANDU reactors [4], for the automated detection and classification of defects in pressure tubes [9], and for flaw detection in ultrasonic scans of nuclear fuel channels using deep learning [10, 11].

Beyond component monitoring, ML techniques have been explored to optimize plant operations and enhance nuclear safeguards. Applications include improving the real-time localization of fuel defects by integrating ancillary data sources [5], assessing the fitness-for-service of CANDU reactors [6], and verifying refueling activities, which has direct implications for nuclear safeguards [7]. Furthermore, ML has been utilized in the specialized domain of nuclear forensics for the classification and visualization of radioactive materials [8] and in managing energy for hybrid power facilities that integrate nuclear with renewable sources [13].

More recently, the application of Large Language Models (LLMs) has emerged as a promising frontier in the nuclear industry. Research has shown the utility of LLMs in processing and classifying unstructured operational data, such as using an LLM to automatically categorize Station Condition Records (SCRs) as either safety-related or non-safety-related to augment the manual review process [12]. Another key area of investigation has been evaluating the performance of LLMs on domain-specific question-answering tasks, where it was demonstrated that a Retrieval Augmented Generation (RAG) framework significantly improves the accuracy of responses from models like ChatGPT

on nuclear-specific queries [10]. These studies highlight a clear trend towards leveraging advanced AI to extract value from the vast amounts of data generated in the nuclear sector [14, 15].

The expected future direction of the field is to utilize smaller local model often with lower precision to reduce the power and the overall system cost when necessary. It is expected that investigating lower precision and more energy efficient computing such as low quantization or approximate computing will be part of the future direction of this field [16, 17, 18, 19, 20, 21, 22]. These efficiency gains are critical for future work in LLMs, which will likely involve a shift towards specialized models that act as experts on a particular task. This focus on smaller, more efficient models makes local, on-premises deployment more feasible and opens possibilities for deployment on edge devices. This trend also supports the development of advanced multi-agent frameworks, where multiple expert models can collaborate to solve more complex problems

While the application of machine learning has shown promise, the recent advent of Large Language Models (LLMs) represents a new frontier with a distinct set of challenges, particularly within the high-stakes nuclear domain. The primary appeal of LLMs lies in their advanced ability to understand and generate natural language, thereby opening up various possibilities. For an organization like Ontario Power Generation (OPG) and other large enterprises, this technology could power a chatbot capable of navigating decades of accumulated Operating Experience (OPEX) data, which is often spread across multiple formats and locations, making information retrieval a significant challenge. However, the direct application of general-purpose LLMs like ChatGPT is fraught with complications that this thesis aims to address.

A primary barrier is the inherent knowledge gap of these models. Trained on vast, public internet datasets, commercial and open-source LLMs lack the specialized, nuanced understanding of nuclear science and engineering required for day-to-day tasks in a power plant. This limitation is compounded by the prohibitive cost of training a foundational model from scratch, which can run into millions of dollars. A second major challenge is

the limited context window of LLMs. While modern models can process large amounts of text, their context windows are still insufficient to hold the vast knowledge base of a nuclear facility, and they can struggle to recall information provided at the beginning of very long prompts.

Perhaps the most critical issue for the nuclear industry is the risk of "hallucination," where an LLM generates convincing but factually incorrect information. This phenomenon often arises because models are optimized to produce responses that a human would prefer, and a confident (but wrong) answer is often preferred over an admission of not knowing. In an environment where accuracy is paramount to safety, such inaccuracies are unacceptable. One of the key methodologies this thesis investigates to mitigate these risks is Retrieval-Augmented Generation (RAG). The RAG approach enhances LLM responses by grounding them in external, verified knowledge sources. By retrieving relevant facts from trusted document repositories and providing them to the model as context for its answer, RAG can significantly improve factual accuracy, relevance, and explainability, making it a promising pathway for safer LLM deployment in the nuclear field [1].

Beyond the model's operational reliability, the data itself presents another layer of complexity. An extensive volume of the nuclear industry's knowledge is embedded in unstructured text formats such as technical documents, regulatory reports, and operational logs. This information is not immediately suitable for training or fine-tuning LLMs, which perform optimally with structured, digestible data like question-answer (Q&A) pairs. The process of manually creating such structured datasets is hindered by three main obstacles: data scarcity, due to strict security and privacy regulations that limit access to high-quality information ; privacy concerns, related to the sensitive and often classified nature of nuclear data ; and the sheer complexity of the task, which requires domain expertise to accurately organize technical details without compromising data integrity.

To address this "data gap," this thesis explores the innovative potential of synthetic data generation [2]. This methodology leverages the analytical capabilities of LLMs themselves to parse unstructured nuclear texts and automatically generate high-quality, structured

Q&A datasets. This approach offers a scalable and privacy-preserving solution, creating valuable, model-ready training and evaluation data without exposing sensitive source information. By transforming latent knowledge into an actionable format, synthetic data generation paves the way for improved information retrieval and the development of more robust, domain-aware AI models.

Finally, the reliance on external, third-party LLMs introduces fundamental security and confidentiality risks that are often incompatible with the stringent requirements of the nuclear industry. Commercial LLM APIs typically operate as "black-box" services, giving the user no visibility or control over how their data is processed or stored, which is not feasible for an organization handling sensitive information. To counter these risks, the ultimate step in achieving a truly secure system is to bring model development entirely in-house. Chapter 4 of this thesis details such an approach: training a specialized, compact LLM from scratch within a secure, on-premises environment using only a single GPU [3]. This work demonstrates the feasibility of creating custom models that adhere to the nuclear industry's strict cybersecurity and data confidentiality standards, even for organizations without access to large-scale computing clusters. By maintaining full control over the entire training pipeline—from data acquisition to model deployment—the risk of data leakage or unauthorized access is considerably reduced, aligning with the industry's deep-rooted culture of safety and security.

1.3 Contributions to Manuscript

This thesis has led to the following peer-reviewed conference publications:

The first paper, "Evaluating ChatGPT on Nuclear Domain-specific data," was presented at the 43rd Annual CNS Conference and the 48th Annual CNS/CNA Student Conference in Saskatoon, SK, Canada, from June 16-19, 2024. This work investigates the performance of Large Language Models (LLMs) on specialized nuclear domain information. The study compares the accuracy of answers generated directly by ChatGPT to those produced using a Retrieval Augmented Generation (RAG) framework. The core contribution of this paper

is demonstrating that incorporating a RAG pipeline significantly enhances the factual accuracy and contextual relevance of LLM responses for nuclear-specific queries, thereby mitigating the issue of model "hallucination."

The second paper, "Towards Secure and Private Language Models for Nuclear Power Plants," was submitted to the 44th Annual CNS Conference and the 49th Annual CNS/CNA Student Conference, held in Toronto, ON, Canada, from June 8-11, 2025. This paper details the development of a secure, private, and domain-specific LLM for the nuclear industry. The model was trained from scratch on a single GPU using the *Essential CANDU* textbook. This approach ensures that sensitive information remains within a secure environment. The paper outlines the complete workflow, from data preparation and tokenization to model architecture and training. The resulting model, though compact, demonstrated the ability to learn and reproduce specialized nuclear terminology.

The third publication, "Unlocking the Potential of Large Language Models in the Nuclear Industry with Synthetic Data," was also submitted to the 44th Annual CNS Conference and the 49th Annual CNS/CNA Student Conference in Toronto, ON, Canada, from June 8-11, 2025. This research addresses the challenge of data scarcity for training LLMs in the nuclear sector. The paper presents a methodology for generating high-quality, synthetic question-answer (Q&A) pairs from unstructured text, specifically the *Essential CANDU* textbook. The process involves advanced text processing, including chunking, embedding generation, and clustering, followed by the use of an LLM to create a diverse and structured dataset. This work provides a pathway to creating robust training and evaluation datasets for LLMs in a data-sensitive domain.

During my degree, I co-authored papers with other colleagues on the topic of application of LLMs in the nuclear industry. These included proposing a safer LLM framework for database queries as an alternative to NL-to-SQL systems [24], designing a cloud AI and OCR system to automate equipment identification from technical drawings [23], and developing an LLM-based classifier to improve safety event identification efficiency by augmenting manual review of Station Condition Records (SCRs) [12].

CHAPTER 2. EVALUATING LLMs (CHATGPT) ON NUCLEAR DOMAIN SPECIFIC DATA

2.1 Introduction

The nuclear industry is an OPEX (Operating Experience) driven industry. The sheer volume of accumulated OPEX data, spread across multiple formats and locations, makes finding relevant information a time-consuming challenge. OPG sought to utilize a chatbot for users to find answers to their nuclear domain-specific questions more easily.

To achieve this task, OPG turned to large language models (LLMs). LLMs have revolutionized Natural Language Processing (NLP) with their remarkable abilities to understand and produce language, impacting various artificial intelligence domains. These LLMs take the form of proprietary models offered as an API service such as ChatGPT, Cohere or Claude which are ready to use out of the box for general purpose tasks, as well as open-source model weights like Mistral and Llama which often require fine tuning before they can be used.

An emerging field in the NLP domain and LLM application development is Retrieval-augmented generation. The idea of RAG is that providing LLM high-quality facts from existing knowledge bases can enhance accuracy and context relevance. This could be especially useful in the nuclear domain where there is several decades of knowledge that is stored in various databases and document repositories. Providing a means for LLM to use that information to answer a question could greatly enhance their performance in nuclear domain.

2.2 Problem Statement

Using LLMs greatly improves the user experience over traditional chatbots which relied on basic language models to detect “utterances” and then route the user through a series of prewritten messages based on the user’s intent. LLMs on the other hand can provide much

better understanding and generate natural text responses to users' questions. However, LLMs struggle with a few problems:

1. LLMs are expensive to train; training a foundational model is extremely costly, in the range of millions of dollars for a multibillion parameter model like ChatGPT. Without training on nuclear data, commercial and open source LLMs have a limited knowledge of the nuclear domain except for what little nuclear information from the public domain which is insufficient for day-to-day tasks in a nuclear power plant.
2. As a workaround to the above problem, LLMs can be provided information at inference time along with the users question as part of the prompt. However, this leads to the next problem which is that LLMs have a limited context window. In the early days of ChatGPT, this was limited to only 4096 tokens which is roughly 2-3 pages of text, hardly enough space to fit decades of nuclear information. The latest state of the art LLMs have massive context windows of over 100k tokens, but this is still insufficient to fit the vast amounts of data available, and other issues arise where LLMs forget information in the beginning part of long prompts.
3. The final concern is around hallucinations; as a side effect of RLHF (Reinforcement Learning from Human Feedback) where LLMs will make up incorrect facts in a very convincing manner. This behavior comes about from the LLM trying to produce a response a human would prefer, and humans do not like to be told "I don't know". Various techniques are available to suppress this behavior as will be discussed in this paper.

One of the emerging methods to address all three concerns above is to utilize Retrieval Augmented Generation (RAG) where data is semantically queried at runtime so that only the needed context can be passed to the model, and the model can be instructed to answer based on that information alone. The approach is discussed in the next section.

This paper primarily focuses on evaluating the performance of LLMs (specifically ChatGPT) when Retrieval Augmented Generation (RAG) is applied. The goal is to

compare the quality of responses generated using RAG against direct responses from ChatGPT to determine if RAG can improve accuracy in the nuclear domain. This evaluation aims to provide strong evidence supporting the use of RAG for developing chatbots capable of generating accurate, nuclear-specific responses.

2.3 Retrieval-Augmented Generation (RAG) : A Background

Retrieval-Augmented Generation (RAG) is a technique that enhances the capabilities of large language models (LLMs) by grounding their responses in external knowledge sources. This approach is particularly valuable when LLMs need to operate in specialized domains or require access to rapidly changing information that may not be reflected in their pre-training data.

Key Steps in the RAG Methodology

1. **User Question:** The process begins with a user's query expressed in natural language.
2. **Query Expansion (Optional):** To optimize the search process, the query may be expanded to include synonyms, related terms, or disambiguate acronyms and technical jargon. This step can be performed using rule-based methods or fine-tuned language models.
3. **Information Retrieval:** RAG uses a combination of semantic and keyword-based search techniques to retrieve the most relevant documents from a knowledge base.
 - **Vector Search:** Vector databases containing pre-computed text embeddings enable semantic similarity comparisons. Cosine similarity is commonly used to identify the most relevant documents to the user's query.
 - **Keyword Search:** Techniques like BM25 help ensure that documents containing specific keywords are considered, boosting recall.
 - **Result Merging:** Search results from vector and keyword searches are combined using methods like Reciprocal Rank Fusion (RRF) to produce a final ranked list of documents.

4. **Augmented Prompt Construction:** The retrieved documents are formatted into a structured context, along with the user's query. This augmented prompt, along with domain-specific instructions, is provided as input to the LLM.
5. **Bot Response:** The LLM, guided by the context and instructions, generates a response that is both accurate and consistent with the provided knowledge base. The LLM should indicate when it is unable to find relevant information within the context.

Advantages of RAG

- **Access to Current Information:** RAG enables LLMs to stay up-to-date even in domains with rapidly changing knowledge.
- **Domain Specialization:** RAG allows LLMs to become domain experts, improving accuracy within specific fields.
- **Explainability:** By grounding responses in retrieved documents, RAG provides transparency into the LLM's reasoning process.

2.4 Related Work / Literature Review

To address the challenges of domain-specific question answering that leverages the power of LLMs in the nuclear domain, it's crucial to examine emerging research on domain-specific question-answering using these models.

Research in this area highlights the potential for enhancing Large Language Models (LLMs) to better handle industry-specific questions. One major obstacle is their limited understanding of specialized industry terminology and concepts [26]. To address this, researchers propose using smaller, domain-specific language models that act as "experts" to supplement the broader knowledge of LLMs [26]. Additionally, integrating LLMs with domain-specific question answering systems often incurs high costs, especially when large

datasets of contextual information are required [25]. This highlights the need for cost-effective strategies, such as selectively reducing context size or employing less expensive LLMs for certain tasks [25]. These insights underscore how adapting LLMs for technical industries like the nuclear domain necessitates both accuracy and cost-efficiency considerations.

The issue of accuracy and reliability in LLM outputs becomes especially critical in the nuclear domain, where factual errors can have serious consequences. A recent survey delves into this complex problem, exploring how LLMs store and process knowledge to identify the root causes of inaccuracies [27]. Furthermore, research on improving large language models through external knowledge and automated feedback mechanisms demonstrates the potential of refining LLM responses to ensure factual correctness in domain-specific question answering [28]. These research efforts highlight the ongoing development of methods to enhance LLM trustworthiness, a crucial aspect for their successful integration within the nuclear industry.

It's important to be aware that LLMs tend to "hallucinate" – generating text that may seem plausible but contradicts established knowledge [29]. Understanding the various forms of hallucinations and techniques for grounding these models can help mitigate this issue in domain-specific scenarios. Continued research into better evaluation metrics, methods to increase controllability of LLM output, and ways to improve their explainability are vital to address the challenge of hallucinations. These advancements will further bolster the reliability of LLMs for domain-specific data within the nuclear sector.

2.5 Evaluation Methodology

This paper evaluates and compares ChatGPT's direct responses with those generated using Retrieval Augmented Generation (RAG), specifically within the nuclear domain. The goal is to determine which approach yields the most accurate and reliable results for LLM applications in the nuclear industry.

Evaluation Dataset

The first step in this evaluation was curating a dataset specifically tailored for the nuclear domain. Since ChatGPT has been trained on most of the data available on the internet in the public domain, it was decided to select nuclear domain specific material which is publicly available. The Essential CANDU [28] textbook was selected for this purpose. This comprehensive resource is widely used for understanding CANDU nuclear reactors and the fundamentals of nuclear science and engineering within that context. The textbook's structure, split into chapters like Reactor Statics, Reactor Dynamics, Instrumentation and Control Systems, and Electrical Systems, provided a natural framework for our dataset.

To create a curated dataset, the textbook was carefully reviewed extracting relevant question-and-answer pairs. These answers serve as the 'ground truth' for evaluating ChatGPT's responses. Here are a few examples of these question-and-answer pairs:

Question: How is magnetic field created inside a synchronous generator in a CANDU Nuclear Power Plant?

Answer (Ground Truth): To create a magnetic field inside a synchronous generator, separate windings and an electrical power source must be used. This part of the generation system is known as the excitation system. The excitation system is essentially a controllable DC source. By adjusting the excitation system output voltage, the output voltage level of the generator can be controlled, and hence the reactive power output.

Question: What are the different types of generators in a CANDU Nuclear Power Plant?

Answer (Ground Truth): The main generator, the standby generators, and the generators in the emergency power system.

Question: In the context of CANDU Nuclear Power Plant, how long can class 3 power be interrupted for?

Answer (Ground Truth): Class III power can be interrupted for up to 5 minutes.

2.6 Answer Generation Without RAG

ChatGPT-3.5 offers significant advantages for question answering. Its training on a vast dataset provides extensive knowledge and the capacity to comprehend various question forms. Additionally, its generative nature enables it to craft responses that are both fluent and relevant, closely resembling human-written text. However, it's crucial to remain aware of two potential drawbacks: inherent biases and the risk of factual inaccuracies. Due to its training on a massive internet dataset, ChatGPT-3.5 may have internalized incorrect or misleading information.

Here's the input prompt provided to ChatGPT-3.5, designed to mitigate these risks:

Answer the question below.

Please stick to facts and avoid including any information which you're not sure about.

Question: {question}

The {question} placeholder gets populated iteratively with each question from the evaluation set.

2.7 Retrieval Augmented Generation

To generate RAG-based responses for evaluation, a dedicated pipeline was developed. As outlined previously, a standard RAG pipeline consists of these key stages:

Vector Index Creation

The 'Retrieval' aspect of RAG begins by dividing source documents into smaller 'chunks' to ensure consistent semantic meaning within each text segment. This enhances the quality of the vector embeddings generated for each chunk. We used OpenAI's ada-002 embedding model for this process. These embeddings were then incorporated into a Facebook AI Similarity Search (FAISS) vector index, enabling efficient retrieval during the query phase.

Search and Prompt Augmentation

When a question is posed, the vector index is searched for relevant chunks that could aid in answering it. This search can be conducted using various methods; we opted for cosine similarity calculations between the query vector and those within the index.

For our evaluation, Facebook AI Similarity Search (FAISS) facilitated fast and scalable similarity searches. FAISS optimizes its speed through specialized indexing techniques that organize vectors for rapid identification of a query vector's nearest neighbors.

The most relevant retrieved passages are then combined with the original query as the input prompt for the LLM. This augmentation provides context and increases the probability of a factually grounded response. We used ChatGPT-3.5 for consistency with the non-RAG evaluation.

The following prompt structure is used for RAG:

Answer the question based only on the following context:

{context}

Please do not use any external information.

Please include as much information as possible in your response from the provided context.

Question: {question}

The {context} placeholder gets populated with retrieved chunks from similarity search. The {question} placeholder gets populated iteratively with each question from the evaluation set.

Response Generation

The final stage utilizes a generative language model. Again, ChatGPT-3.5 was selected for consistency. Its fluency, coherence, and ability to follow instructions enable it to synthesize responses that align with the retrieved documents from the RAG pipeline.

2.8 Generation of Answers

To compare the performance of RAG against direct ChatGPT-3.5 responses, we generated answers for the questions in our evaluation dataset using both methods. Below, we illustrate the differences between these approaches with a few examples:

Question: How is magnetic field created inside a synchronous generator in a CANDU Nuclear Power Plant?

Answer with RAG: In a synchronous generator in a CANDU Nuclear Power Plant, the magnetic field is created by energizing the rotor winding through the field excitation circuit. As the turbine rotates the rotor, a rotating magnetic field is generated. The excitation current is supplied to the rotor winding through slip rings. The rotating magnetic flux induces a potential in the stator winding, resulting in the creation of the magnetic field inside the generator.

Answer without RAG: In a CANDU Nuclear Power Plant, the synchronous generator's magnetic field is generated by an excitation system, consisting of field windings around the rotor. When DC power—supplied by a DC generator or a rectifier system—is applied

to these windings, a magnetic field is created. The rotating rotor within this magnetic field induces an electric current in the stator windings, which then produces the electrical output for the grid.

Question: What are the different types of generators in a CANDU Nuclear Power Plant?

Answer with RAG: The different types of generators in a CANDU Nuclear Power Plant are the main generator, standby generators, and generators in the emergency power system.

Answer without RAG: In a CANDU nuclear power plant, electricity is generated by two main types of generators: the main generator, which converts mechanical energy from a steam turbine into high voltage electrical energy for the grid, and the auxiliary generator, a smaller diesel-powered unit that provides backup power for essential systems during outages or maintenance when the main generator is offline. Both are integral for the plant's continuous and reliable power supply.

Question: In the context of CANDU Nuclear Power Plant, how long can class 3 power be interrupted for?

Answer with RAG: In the context of a CANDU Nuclear Power Plant, Class III power can be interrupted for up to 5 minutes.

Answer without RAG: In the context of CANDU Nuclear Power Plant, class 3 power can typically be interrupted for up to 72 hours. Class 3 power refers to the power supply required for the safe and controlled shutdown of the reactor and maintaining the cooling systems. This duration allows for necessary maintenance, repairs, or other unforeseen events while ensuring the safety and stability of the plant. It is important to note that this duration may vary depending on plant-specific safety protocols and regulatory requirements.

2.9 Evaluation of Answers

To thoroughly assess the generated responses, we employed a multifaceted evaluation approach:

Human Evaluation:

A human evaluator meticulously compared answers to the ground truth, establishing factual correctness. This provides a direct measure of accuracy.

LLM Based Evaluation:

Recognizing the potential for scalability and interpretability, we leveraged cutting-edge LLMs as judges. Recent research demonstrates remarkable agreement between powerful LLMs like GPT-4 and human preferences [31]. This makes LLM-as-a-judge a viable alternative to potentially costly human evaluations.

For our experiment, GPT-4 assessed the responses generated with and without RAG. We provided the ground truth as context to enable informed evaluations. The GPT-4 evaluation prompt was meticulously crafted to guide judgments. Specific criteria was included for scoring responses, including helpfulness, grounded-ness, correctness, conciseness, coherence, and detail. GPT-4 generated scores for each criterion along with explanations. A final verdict and justification were requested, based on these scores. This structured prompt promotes thoughtful analysis and insightful outputs. The full evaluation prompt is shown below:

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below.

You should choose the assistant that follows the user's instructions and answers the user's question better based on the ground truth.

For this evaluation, you should primarily consider the following criteria:

- 1) HELPFULNESS: "Is the submission helpful, insightful, and appropriate?"*
- 2) GROUNDEDNESS: "Is the submission grounded in facts as compared to the ground truth provided?"*
- 3) CORRECTNESS: "Is the submission correct, accurate, and factual?"*
- 4) CONCISENESS: "Is the submission concise and to the point?"*
- 5) COHERENCE: "Is the submission coherent, well-structured, and organized?"*
- 6) DETAIL: "Does the submission demonstrate attention to detail?"*

Your output will be as follows:

1. Begin your evaluation by comparing the two responses and provide a short explanation, then provide a criterion score for each the two responses as a number from 0 (lowest) to 100 (highest) with granular integral intervals, in output format:

"CRITERION

Evaluation: explanation.

Scores: A: score, B: score"

Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Avoid allowing the length of the responses to influence your evaluation. Avoid favouring certain names of the assistants. Be as objective as possible.

2. After providing your criteria explanation and scores, provide a summary explanation of your final verdict as:

"Verdict Evaluation: evaluation"

3. Then assuming criteria are weighted equally calculate the arithmetic mean for each response using the formula $\text{sum (scores for each criterion) / number of criteria}$, and strictly output in the following format filling in score:

"Verdict Scores: A: score, B: score"

4. After providing your explanation and scores, output your final verdict as:

"Verdict Preference: "

then strictly following this format:

[[A]] if assistant A is better, [[B]] if assistant B is better, and [[C]] for a tie.

Double-check the explanations, scores, and final verdicts have been included.

As seen above in the prompt, instructions were also provided in the prompt to ensure the output from GPT-4 follows a consistent pattern and therefore can be parsed out in a reliable way. This is important otherwise it will require manual effort to collate all the LLM responses to make them presentable.

2.10 Results

Our evaluation demonstrates the clear superiority of responses generated using the RAG pipeline compared to direct ChatGPT-3.5 output. Overall verdict scores for RAG responses were significantly higher.

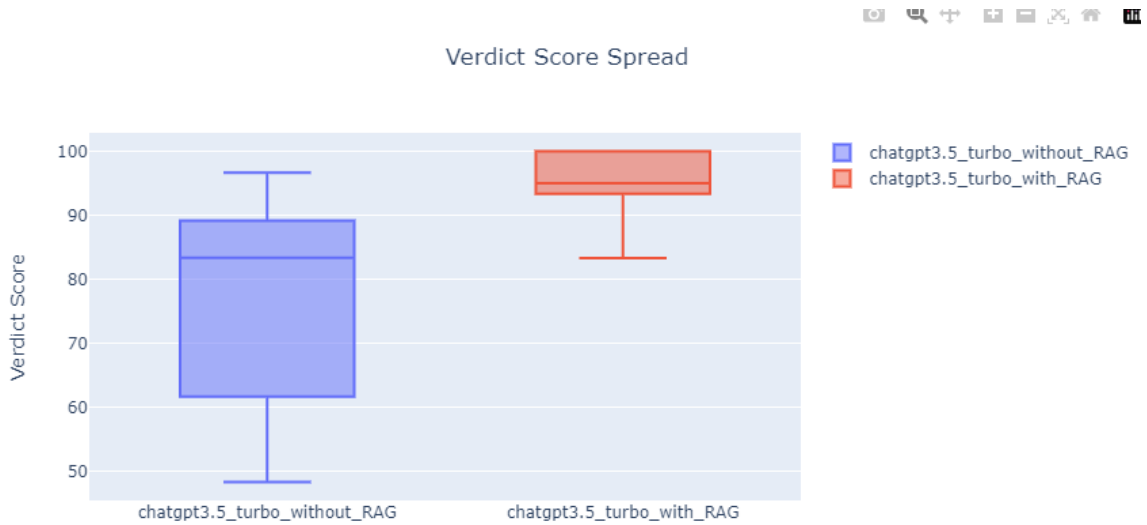


Figure 2-1 : Box Plot comparing responses for ChatGPT3.5 with RAG and without RAG

This above box plot visually underscores the consistent improvement with RAG. It also reinforces the findings of our human evaluator, who consistently rated RAG responses as equal or superior to non-RAG responses.

Furthermore, RAG responses exhibit remarkable consistency, with scores tightly clustered between 90 and 100. Conversely, direct ChatGPT-3.5 responses demonstrate greater variability (60 to 90), highlighting the inherent randomness sometimes found in its output. This suggests that the RAG pipeline provides crucial context, anchoring LLM responses and enhancing their accuracy.



Figure 2-2 : Column Plot comparing scores for ChatGPT3.5 with RAG and without RAG

This above plot offers a direct comparison of average metric scores. RAG-generated responses significantly outperform direct ChatGPT-3.5 responses in correctness, groundedness, and helpfulness – all key indicators of factual accuracy and alignment with the ground truth.

Additionally, RAG-augmented responses surpass their counterparts in conciseness. This aligns with the expectation that LLMs provided with targeted context and explicit instructions gravitate towards more focused outputs. Coherence and detail scores show no significant difference between the two approaches.

2.11 Conclusion

Our evaluation convincingly demonstrates the value of a RAG pipeline in ensuring the factual accuracy, relevance, helpfulness, and conciseness of LLM-generated responses. This aligns with the fundamental architecture of LLMs, which rely on probabilistic predictions based on their training data. While their vast language knowledge is impressive, the lack of curated training datasets in specific domains can lead to factual inaccuracies. By equipping LLMs with RAG pipeline, we provide targeted context and reinforce their ability to follow instructions, leading to significantly improved results.

CHAPTER 3. UNLOCKING THE POTENTIAL OF LARGE LANGUAGE MODELS IN THE NUCLEAR INDUSTRY USING SYNTHETIC DATA

3.1 Introduction

The nuclear industry is inherently data intensive. Vast volumes of technical documents, regulatory reports, and operational logs contain valuable insights; however, much of this information remains locked away in unstructured text formats. These documents, rich in technical details and critical operational data, are not immediately usable by advanced AI systems, particularly for sophisticated applications involving Large Language Models (LLMs).

LLMs can be likened to highly capable interns: for them to develop expertise in a specialized domain, or for their performance to be accurately evaluated, learning materials must be presented in a structured, digestible manner. Instead of being given extensive, unstructured texts directly, their learning is more effective when this information is first broken down into key topics and concepts and then formulated into question-answer (QnA) pairs. This structured approach is crucial for their effective training, fine-tuning, and evaluation.

Synthetic data generation presents an innovative approach to bridge this gap. By leveraging the powerful analytical capabilities of LLMs themselves, synthetic data techniques can convert unstructured nuclear text into the necessary structured QnA datasets. This transformation not only overcomes challenges of data scarcity and stringent privacy requirements but also enables the creation of scalable, domain-specific resources.

Ultimately, this method paves the way for improved information retrieval, enhanced knowledge sharing, and more informed decision-making across the nuclear sector. By unlocking the latent potential within nuclear documentation, synthetic data can be a step towards transforming how AI is harnessed for safer and more efficient nuclear operations.

3.2 Obstacles, Challenges, And the Promise of Synthetic Data

The potential impact of Large Language Models (LLMs) in the nuclear industry is considerable. They offer the possibility to substantially improve information retrieval, knowledge management, and decision support systems. However, realizing this potential faces several obstacles.

Obstacles and Challenges

Several factors limit the effective deployment of LLMs within the nuclear domain:

- **Data Scarcity:** Strict security protocols and privacy regulations significantly restrict access to high-quality, structured nuclear data. Unlike other industries where large, annotated datasets are readily available, the nuclear sector must contend with limited human-annotated data, making the training of robust LLMs especially challenging.
- **Privacy Concerns:** The sensitive and classified nature of nuclear data requires robust privacy protection. As a result, even when data exists, sharing or annotating it can entail considerable risks, further hindering the development of AI systems that rely on this data.
- **Need for Structured Data:** LLMs perform optimally with structured inputs, such as clear QnA pairs. However, the nuclear industry's data is predominantly unstructured. Creating structured datasets from these sources is complex because it requires domain expertise to accurately extract and organize technical details without compromising data integrity or security.

These challenges collectively pose a significant barrier to the effective use of LLMs in nuclear applications, where precision and reliability are paramount.

Synthetic Data Generation as a potential solution

Synthetic data generation emerges as a promising solution to these obstacles. By using LLMs to analyze and reinterpret unstructured nuclear text, this approach creates structured datasets that are both scalable and privacy-preserving. Synthetic data generation addresses these key challenges in the following ways:

- **Mitigating Data Scarcity:** Synthetic techniques allow the generation of numerous QnA pairs from limited seed data. By expanding the dataset without the need for additional sensitive information, synthetic data provides a robust alternative to traditional human-annotated datasets.
- **Enhancing Privacy Protection:** Since synthetic data is generated algorithmically, it avoids the direct use of sensitive or classified information. This not only ensures compliance with strict privacy regulations but also reduces the risk of data breaches while preserving the essential characteristics of the original data.
- **Creating Structured Data:** Leveraging the sophisticated text comprehension and processing capabilities of LLMs, synthetic data generation converts complex, unstructured nuclear texts into well-organized QnA formats. This transformation is achieved through iterative prompt engineering, context-aware extraction, and quality evaluation, ensuring that the resulting data is both accurate and reflective of the domain's intricacies.

Through a carefully designed process that includes iterative refinement and human-in-the-loop validation, synthetic data generation can produce high-quality, diverse datasets tailored specifically for the nuclear industry. This methodology not only overcomes the traditional limitations posed by data scarcity and privacy but also offers new opportunities for deploying advanced AI models in nuclear safety, reactor operations, and regulatory compliance.

Table 3-1 : Synthetic data as a potential solution

Challenge	How Synthetic Data Provides a Solution
Data Scarcity	Enables the generation of numerous QnA pairs from limited seed data, thus expanding available training and evaluation data sets
Privacy Concerns	Algorithmically generates new data, avoiding the direct use of sensitive or classified original information.
Need for Structured Data	Transforms complex, unstructured nuclear texts into well-organized, LLM-ready QnA formats using LLM capabilities.

3.3 Related Work / Literature Review

Recent years have witnessed significant advances in synthetic data generation, driven by the advanced capabilities of Large Language Models (LLMs). These efforts are of particular importance for domains where data is scarce, sensitive, or unstructured – such as the nuclear industry. In this section, I review the literature across three broad themes: best practices and techniques for synthetic data generation, cost-effectiveness and scalability of these methods, and domain-specific challenges with an emphasis on privacy and accuracy.

Best Practices and Techniques

Liu et al. [32] provide a foundational guide on best practices in synthetic data generation, outlining methods to maintain both fidelity and diversity in generated datasets. Their work emphasizes the importance of careful prompt engineering and iterative refinement to mitigate common issues such as hallucinations and overgeneralization.

Complementing this, Long et al. [33] present a survey of LLM-driven synthetic data generation, curation, and evaluation techniques. They categorize the literature into prompt-based approaches and multi-step generation pipelines. The authors emphasize that even with advanced LLMs, human oversight remains essential to ensure data quality. Chan et al. [34] expand on these themes by reviewing a broad spectrum of synthetic data techniques. These range from simple data augmentation methods to more complex

approaches that incorporate retrieval-augmented generation. Their review underscores that while many techniques are domain-agnostic, fine-tuning for specialized areas requires adjustments to account for domain-specific terminology and operational contexts.

Cost-Effectiveness and Scalability

As synthetic data generation becomes an important component for training advanced models, several studies have investigated the trade-offs between cost and performance. Zhu et al. [35] analyze different synthetic data generation strategies by quantifying how varying query budgets affect the overall quality of the synthetic datasets. Their work reveals that, in resource-constrained settings, augmenting responses to existing seed data can be more effective than generating entirely new prompts.

Further expanding on these insights, Zhu et al. [36] introduce the RAGEval framework. This framework not only facilitates the generation of synthetic evaluation datasets but also proposes novel metrics for measuring factual completeness, hallucination, and irrelevance in generated content. These metrics provide a solid basis for comparing different synthetic data generation strategies, especially in complex domains where data quality is critical.

In parallel, studies such as those by Smith et al. [37] and Doe et al. [38] have focused on scaling synthetic data generation for low-resource and sensitive domains. They demonstrate that by leveraging domain-specific seed data and employing controlled augmentation strategies, one can generate high-quality synthetic datasets. These datasets can significantly improve downstream task performance without incurring substantial annotation costs.

Domain-Specific Challenges in High-Security Areas

The nuclear industry presents unique challenges that are not typically encountered in more open domains. Privacy concerns and data sensitivity require that any synthetic data not only mirrors real-world patterns but also adheres to stringent security standards. Zhang et al. [39] delve into these challenges by examining the specific obstacles faced when applying LLMs to generate synthetic data in high-security domains. Their work highlights that inaccuracies (or “hallucinations”) in generated content can have serious consequences in safety-critical applications, thereby necessitating robust validation mechanisms.

Kumar et al. [40] propose a unified framework for synthetic data generation in sensitive domains, emphasizing the need for human-in-the-loop processes to continuously validate and correct generated data. Their approach combines automated prompt-based generation with expert reviews to ensure that the synthetic QnA pairs are both accurate and aligned with domain-specific requirements.

Finally, Lee et al. [41] specifically address synthetic data applications within the nuclear sector. They discuss how transforming unstructured technical documents into structured QnA datasets can enable advanced AI applications such as predictive maintenance, regulatory compliance monitoring, and decision support. Their study illustrates the significant potential of synthetic data in revealing valuable insights from vast repositories of nuclear information, while also ensuring that the data remains secure and privacy-preserving.

Synthesis and Future Directions

Collectively, these studies underscore the considerable promise of LLM-driven synthetic data generation, particularly in domains where conventional data collection is constrained by cost, privacy, or scarcity. While best practices and scalable techniques have been well documented, the nuclear industry remains a demanding testbed due to its inherent sensitivity and the essential need for factual precision.

Future research is likely to focus on further refining human-in-the-loop validation mechanisms, developing domain-specific prompt engineering techniques, and establishing standardized evaluation metrics. By continuing to build on the foundational work described above, the community can develop more reliable, secure, and scalable synthetic data generation frameworks, paving the way for improved AI applications in the nuclear industry and other high-stakes fields.

3.4 Synthetic Data Generation Process

This section outlines the process for generating synthetic question-answer pairs from unstructured text data in the nuclear domain, with a specific focus on CANDU reactor technology. The approach employs Large Language Models (LLMs) to convert raw text from the "Essential CANDU" textbook [30] into structured QnA pairs for downstream AI model training and evaluation tasks in the nuclear industry.

The overall process, visually summarized in Figure 3-1 and detailed in subsequent sections, begins with Data Extraction and Preprocessing. This initial phase prepares the raw text data for efficient LLM processing through two main activities. Text Chunking, which divides the text into manageable, semantically meaningful segments ('chunks') while preserving context, and Summarization, which uses a specifically designed LLM prompt to extract key concepts and technical details from each chunk. Following data extraction, Embedding Generation and Clustering are employed to further organize the data and preserve contextual relationships. This involves generating numerical vector embeddings for text chunks via an LLM, after which clustering algorithms like k-means group semantically similar chunks.

With the data prepared and organized, the subsequent core stage is Synthetic QnA Pair Generation. In this central step, a detailed LLM prompt guides the formulation of diverse and complex questions about CANDU reactor technology, ensuring the generated answers are accurate, complete, and relevant to the source text. Finally, the process concludes with QnA Evaluation, where the generated QnA pairs are assessed for their accuracy, domain relevance, and diversity.

This structured methodology ensures the production of good-quality synthetic QnA pairs that accurately reflect the source text and are suitable for training and evaluating AI models in the nuclear domain. The following subsections will elaborate on each of these steps, including the specific techniques and algorithms used.

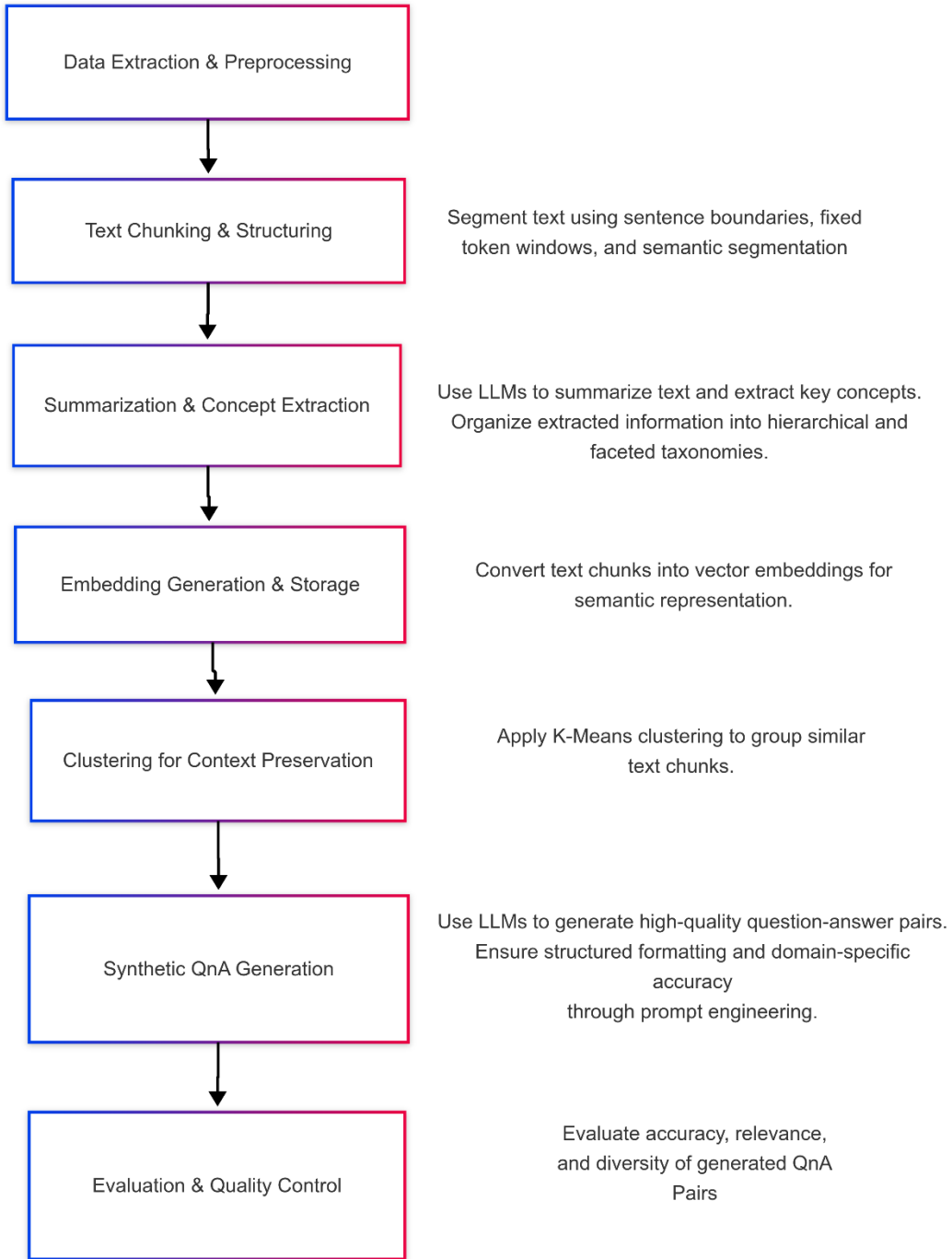


Figure 3-1 Overall Structure of Synthetic QnA pairs pipeline

3.4.1 Data Extraction and Preprocessing

This stage includes Text Chunking, Key Concept Extraction and Information Summarization which are detailed below:

Text Chunking:

The field of text chunking, which focuses on segmenting textual data into meaningful units—or ‘chunks’—while preserving semantic integrity, is continually evolving. Chunking is needed because LLMs have limitations in terms of context window due to the underlying attention mechanism and therefore can only effectively accept and process a smaller segment of text at a time. Some of the established strategies include:

- **Sentence Boundary-Based Chunking:** This approach segments text at natural sentence breaks to preserve coherence and readability, ensuring that related ideas remain grouped together.
- **Fixed-Size Token Windows:** Text is divided into rolling windows of a fixed size (e.g., 512 tokens), often with overlapping buffers to maintain contextual continuity across segments.
- **Semantic Segmentation:** NLP techniques such as BERT-based segmentation and Latent Dirichlet Allocation (LDA) are used to ensure chunking is conceptually meaningful, allowing segments to align with topic boundaries.
- **Dynamic Chunk Sizing:** This adaptive strategy adjusts segment boundaries based on various content characteristics. For instance, it may involve assessing Text Density, often measured using term frequency-inverse document frequency (TF-IDF), to identify information-rich sections requiring finer segmentation. It also considers Contextual Importance, where techniques like Named Entity Recognition (NER) and topic modeling help preserve domain-specific relationships between concepts. Furthermore, Adaptive Thresholding might be employed, using a combination of heuristics and machine learning models to determine chunk boundaries from key phrases and domain knowledge.

For this chapter, as the CANDU textbook was used as the primary data source, the text was extracted using Azure Document Intelligence. This tool identifies meaningful separations within the textbook, such as paragraphs, new sections, and chapter divisions. These naturally occurring separations were leveraged to form chunks, which were further organized based on page numbers to maintain document structure and contextual integrity.

Key Concept Extraction and Information Summarization:

A carefully constructed prompt is used to extract key information from the documents, enabling its use for clustering and downstream QnA generation. The summarization prompt was structured to guide the Large Language Model (LLM) towards extracting detailed and organized information from the technical text. By explicitly specifying categories of information to extract (such as technical concepts, system components, operational processes, and safety protocols), the prompt directs the LLM to focus on key relevant aspects of CANDU related content. This structured approach facilitates the extraction of important insights and ensures the output is organized and formatted in a way which is suitable for subsequent analysis and knowledge integration.

Furthermore, the prompt incorporates contextual information about the "Essential CANDU" textbook [28] and its target audience. This helps the LLM better understand the technical depth and scope of the text, leading to more accurate and relevant summarization. The emphasis on factual accuracy and clear organization within the prompt ensures the extracted information is reliable and can be directly used for downstream tasks like clustering and QnA generation.

Example LLM Prompt for Summarization

The Essential CANDU is a comprehensive, peer-reviewed textbook that delves into the intricacies of CANDU nuclear power technology. It serves as a valuable resource for senior undergraduate and graduate students, educators, trainers, and professionals in the nuclear industry. The textbook covers a range of topics, including reactor physics, thermal-

hydraulics, plant systems, operations, safety analysis, and more. Its thorough exploration of these subjects makes it a suitable reference for those seeking both foundational knowledge and in-depth technical insights into CANDU reactors.

You are an advanced information extraction system designed to parse and extract structured insights from technical nuclear engineering texts, specifically focusing on CANDU reactor technology. Given the following excerpt from "The Essential CANDU" textbook, systematically identify and extract key information as outlined below.

Key Technical Concepts and Terminology:

Identify and list primary technical terms and concepts introduced or discussed in the text. Provide concise definitions or explanations for each term.

System Components and Descriptions:

Enumerate the main components of the CANDU reactor or related systems mentioned in the excerpt. For each component, provide a brief description of its function and significance within the system.

Operational Processes and Mechanisms:

Detail any operational processes, mechanisms, or methodologies described. Explain how these processes contribute to the overall operation and efficiency of the CANDU reactor.

Safety Protocols and Considerations:

Highlight any safety protocols, design considerations, or risk mitigation strategies discussed in the text. Explain their importance in ensuring the safe operation of the reactor.

Historical Context and Evolution:

If applicable, summarize any historical developments, design evolutions, or advancements in CANDU technology presented in the excerpt.

Data and Quantitative Information:

Extract any data, statistics, or quantitative information provided. Present this information in a structured format, such as tables or lists, ensuring clarity and precision.

Illustrations and Diagrams:

Note any references to illustrations, diagrams, or figures in the text. Provide a brief description of each and explain its relevance to the accompanying content.

Ensure that the extracted information is organized in a clear and logical manner, preserving the factual accuracy and technical depth of the original text. Your response should be formatted to facilitate further analysis and integration into specialized databases or knowledge repositories.

Text from Essential CANDU textbook: {text}

3.4.2 Embedding Generation and Clustering

This stage converts the textual data into a format suitable for efficient processing and analysis, preserving the contextual relationships within the text.

Embedding Generation

To enable semantic-based analysis and retrieval, the textual chunks are transformed into numerical vector representations called embeddings. These embeddings are designed to capture the semantic meaning of the text, allowing for the effective comparison and grouping of similar chunks.

The fundamental intuition behind text embeddings is to represent textual information in a continuous vector space where semantic relationships are preserved. Essentially, words or text passages that share similar meanings or contexts are mapped to vectors that are close to each other in this space, while dissimilar texts are positioned further apart. This geometric relationship allows machine learning models to understand and quantify the nuances of language, enabling them to perform complex tasks by operating on these dense vector representations rather than on the raw text itself.

For generating embeddings, the text-embedding-ada-002 model from Azure OpenAI has been used. This model produces 1536-dimensional vectors, providing a detailed representation of the text. These embeddings are valuable for tasks such as semantic search, context-aware question answering, and identifying similarities between different text segments.

Clustering for Context Preservation

In detailed technical documents, such as the "Essential CANDU" textbook that was used, it is common for the same concepts or related information to appear in different sections or be described in slightly varied ways across multiple text chunks. To consolidate such dispersed yet related information effectively before QnA generation, I employed clustering

techniques in my methodology. By grouping semantically similar text chunks, clustering helps to create a more holistic view of specific topics and ensures that context is preserved and leveraged comprehensively. For this purpose, I utilized K-Means clustering to partition text embeddings based on their semantic similarity, ensuring that related chunks are grouped together for improved downstream processing in my project.

The K-Means algorithm operates through an iterative process. Initially, a predefined number of clusters, K , is chosen, and K points within the dataset are randomly selected to serve as the initial group centers, or centroids. These centroids act as the focal points for each cluster. Each data point in the dataset is then assigned to its nearest centroid, typically based on a distance measure like Euclidean distance; closer proximity indicates a higher likelihood of the point belonging to that specific cluster. Following the assignment of all data points, the centroid of each cluster is recalculated by averaging the positions of all points within that cluster, and the centroid is repositioned to this computed average. These steps of assigning data points and updating centroids are repeated iteratively. This process continues until the centroids stabilize—meaning their positions do not change significantly between iterations—which indicates that the cluster assignments have converged.

Mathematically, the K-Means algorithm aims to minimize an objective function, typically the within-cluster sum of squares (WCSS). Given a set of N data point embeddings $X = \{x_1, x_2, \dots, x_N\}$, where each x_i is a d -dimensional vector, K-Means partitions these into K clusters, $C = \{C_1, C_2, \dots, C_K\}$. The objective function J is defined as:

$$J = \sum_{k=1}^K \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \quad (3-1)$$

where μ_k is the centroid of cluster C_k , and $\|x_i - \mu_k\|^2$ represents the squared Euclidean distance between the data point x_i and the centroid μ_k .

The iterative process to minimize J involves two main steps, repeated until convergence (at iteration t):

- **Assignment Step:** Each data point x_i is assigned to the cluster $C_k^{(t)}$ whose centroid $\mu_k^{(t-1)}$ (from the previous iteration) is closest:

$$C_i^{(t)} = \arg \min_{1 \leq k \leq K} \|x_i - \mu_k^{(t-1)}\|^2 \quad (3-2)$$

where $C_i^{(t)}$ is the index of the cluster to which data point x_i is assigned at iteration t.

- **Update Step:** The centroid $\mu_k^{(t)}$ for each cluster is recomputed as the mean of all data points x_i assigned to that cluster $C_k^{(t)}$:

$$\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{x_i \in C_k^{(t)}} x_i \quad (3-3)$$

where $|C_k^{(t)}|$ is the number of data points in cluster $C_k^{(t)}$. Convergence is typically reached when the assignments c_i or the centroids μ_k no longer change significantly between iterations.

To ascertain an appropriate number of clusters (k) for my dataset, the Elbow Method is deployed. This technique involves plotting the within-cluster sum of squares (WCSS), often referred to as inertia, against a range of k values. The optimal number of clusters is typically identified at the point on the plot where adding more clusters ceases to substantially reduce the WCSS, forming a distinct "elbow." Figure 3-2 illustrates this for

my dataset, showing that after $k=6$, any additional cluster does not substantially reduce the variance. This approach helps ensure that the clustering model captures meaningful groupings without introducing excessive complexity.

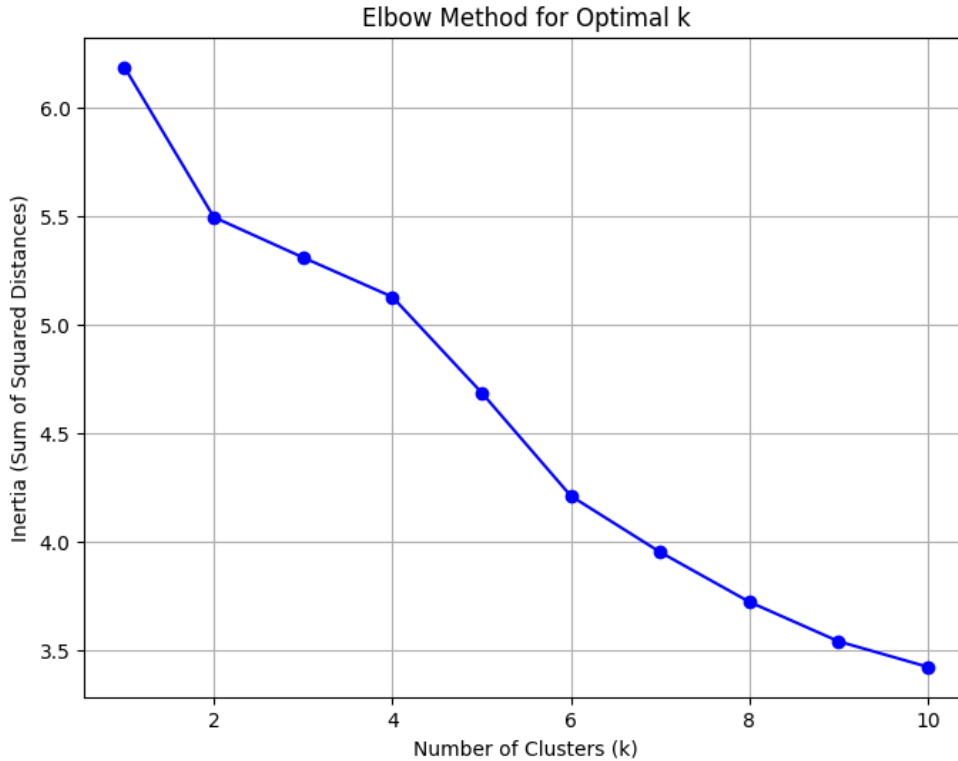


Figure 3-2 : Elbow plot to determine optimal number of clusters

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a useful technique for visualizing high-dimensional data in a lower-dimensional space while preserving local similarities. In the context of my work, the t-SNE plot provides an intuitive way to observe the distribution of chunk embeddings and their respective clusters. The color-coded points correspond to different cluster assignments, revealing the structure of my dataset. The visualization indicates that the embeddings have been effectively grouped, with distinct regions corresponding to different clusters. This separation suggests that my clustering approach successfully captured meaningful patterns in the text data, allowing for contextually relevant segmentation.

From the t-SNE plot in Figure 3-3, it is evident that the clusters in my data are well-separated, reinforcing the validity of the K-Means clustering approach that was applied. The clear distinctions between groups indicate that similar text segments are grouped together, supporting better semantic organization. The presence of minimal overlap between clusters suggests that the embeddings retain meaningful contextual relationships; these relationships are important for downstream applications such as retrieval-augmented generation (RAG) and semantic search within my system. This visualization confirms that my clustering strategy effectively maintains the integrity of contextual groupings within the synthetic data generation pipeline.

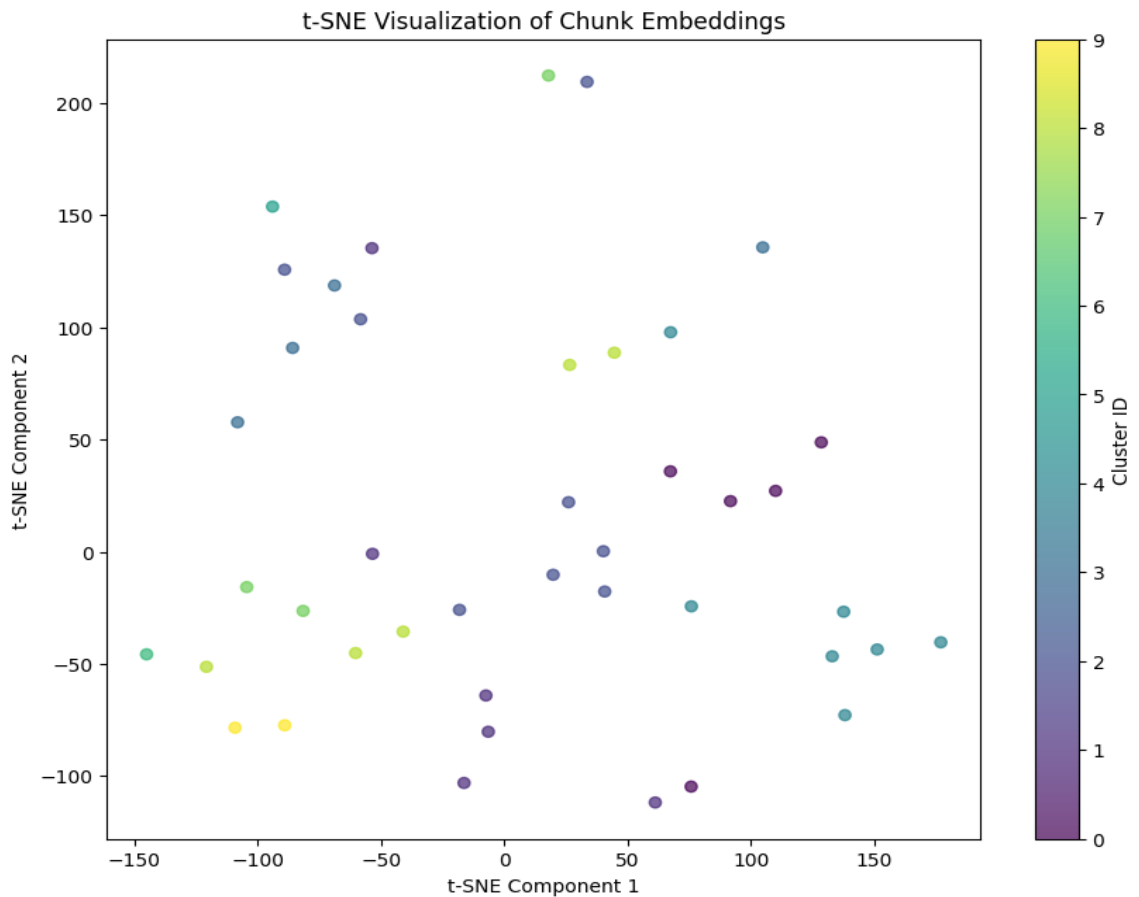


Figure 3-3 : t-SNE visualization of clusters

3.4.3 Synthetic QnA Pair Generation

For this key stage, various Prompt Engineering techniques and strategies were employed to develop Question and Answer pairs. A specifically constructed QnA generation prompt was created, designed to ensure the creation of a detailed and good-quality synthetic dataset for training and evaluating AI models in the nuclear engineering domain, with a specific focus on CANDU reactors. The effectiveness of this prompt stems from several key design strategies, summarized in the below table:

Table 3-2 : Prompt Design Considerations

Prompt Strategy	Design	Description
Contextualization & Specificity	&	Provides clear context by referencing "The Essential CANDU" textbook and explicitly stating the domain (CANDU reactors) and purpose of the QnA pairs, guiding the LLM towards generating relevant and technically accurate content.
Structured Completeness	Info &	Enables the LLM to access and integrate information from both the original text chunk and extracted key information, ensuring generated QnA pairs capture a complete and nuanced understanding, even if information is scattered.
Diversity & Complexity	&	Encourages the generation of a variety of question types, from fundamental recall to open-ended critical thinking, ensuring the dataset covers a wide range of cognitive levels suitable for training adaptable AI models.
Output Usability	Format &	Specifies a JSON output format with source text references, ensuring QnA pairs are structured and organized for easy integration into downstream AI training and evaluation pipelines.

Example Prompt for QnA Generation

You are an advanced synthetic QnA generation system designed to create an extensive set of domain-specific question-answer pairs based on structured nuclear engineering content. Your task is to generate high-quality, diverse, and well-structured QnA pairs from "The Essential CANDU" textbook, ensuring comprehensive coverage across reactor physics, thermal-hydraulics, plant systems, operations, and safety analysis.

You Will Receive:

1. *Original Textbook Chunk (Unstructured Raw Text):*
 - *{textbook_chunks_text}*
2. *Extracted Structured Information from the Text:*
 - *Key Technical Concepts: {textbook_chunks_key_info}*

It is important to note that information may be dispersed across multiple locations in the provided text. You must ensure that complete information is captured in your QnA pairs, even if the answers are spread across different parts of the text. This ensures a full, accurate understanding of the concepts and their interrelationships.

Task: Generate a Large Set of QnA Pairs

Your goal is to generate as many high-quality, domain-specific QnA pairs as possible from the provided text. Each QnA must be accurate, relevant, and structured for training/evaluating AI models in nuclear engineering.

Categories of Questions to Generate:

1. *Fundamental Recall Questions (Basic Knowledge & Definitions)*
 - *Example: "What is the function of the moderator in a CANDU reactor?"*
 - *Example: "Why does CANDU use natural uranium as fuel?"*
2. *Technical Explanations (Step-by-Step Process Questions)*
 - *Example: "Describe the full neutron moderation process in a CANDU reactor."*
 - *Example: "Explain how coolant flow affects reactor power levels."*
3. *Multi-Step Analytical Questions (Cause-Effect, Problem Solving)*
 - *Example: "If moderator temperature increases, how does that impact neutron economy and reactor efficiency?"*
 - *Example: "A fuel channel experiences a pressure drop—what sequence of plant responses will be triggered?"*
4. *Numerical & Calculation-Based Questions*
 - *Example: "Given a neutron multiplication factor (k), calculate the expected reactor power behavior."*
 - *Example: "The specific heat capacity of heavy water is X J/kg·K. If the coolant temperature rises by $Y^\circ\text{C}$, how much heat is absorbed per kg?"*
5. *Operational & Troubleshooting Scenario Questions*

- Example: "If a coolant pump fails, what emergency responses will occur?"
 - Example: "What are the step-by-step procedures for shutting down a CANDU reactor safely?"
6. *Comparative & Evaluative Questions*
 - Example: "How does the CANDU reactor's neutron economy compare to a PWR?"
 - Example: "What are the advantages and disadvantages of online refueling?"
 7. *Historical & Design Evolution Questions*
 - Example: "How has CANDU reactor safety design evolved over time?"
 - Example: "What major modifications were made to early CANDU prototypes to improve efficiency?"
 8. *Diagram & Illustration-Based Questions*
 - Example: "Based on the provided schematic, describe the flow path of coolant through the reactor core."
 - Example: "In Figure X, what is the function of the labeled component Y?"
 9. *Safety, Regulation & Risk Management Questions*
 - Example: "What safety mechanisms are in place to prevent a loss-of-coolant accident (LOCA)?"
 - Example: "How do Canadian nuclear safety regulations impact CANDU reactor design?"
 10. *Open-Ended & Critical Thinking Questions*
 - Example: "If you were to design a next-generation CANDU reactor, what improvements would you make?"
 - Example: "What are the long-term sustainability challenges of heavy water reactors?"

Output Format: JSON for AI Training & Evaluation

Return the QnA pairs in structured JSON format, ensuring that each entry includes references to the original textbook source.

Example Output Format:

```
[
  {
    "question": "What is the primary role of heavy water in a CANDU reactor?",
    "answer": "Heavy water (D2O) acts as both a neutron moderator and coolant. It slows down fast neutrons to thermal energies, increasing the probability of fission in natural uranium fuel.",
    "references": "CANDU_Textbook.pdf - Page 45, Page 48"
  },
  {
    "question": "Describe the sequence of events in a CANDU reactor startup procedure.",
    "answer": "The startup sequence includes: (1) ensuring all safety systems are operational, (2) inserting control rods to maintain subcriticality, (3) initiating primary
```

coolant circulation, (4) starting up neutron sources, (5) gradually withdrawing control rods to allow for controlled criticality, and (6) increasing power while monitoring system stability.",

"references": "CANDU_Operations.pdf - Page 12"

}},

{{

"question": "Calculate the heat transfer rate in a fuel channel given a mass flow rate of 5 kg/s and a temperature rise of 10°C.",

"answer": "Using $Q = mc\Delta T$, with c (specific heat capacity of heavy water) = 4.2 kJ/kg·K, the heat transfer rate is: $Q = (5 \text{ kg/s}) \times (4.2 \text{ kJ/kg}\cdot\text{K}) \times (10 \text{ K}) = 210 \text{ kW}$.",

"references": "CANDU_ThermalHydraulics.pdf - Page 33, Page 34"

}},

{{

"question": "If a CANDU reactor loses moderator circulation, what safety mechanisms will activate?",

"answer": "Loss of moderator circulation would trigger the reactor shutdown system (SDS), rapid insertion of control rods, emergency coolant injection from the ECCS, and containment activation to prevent excessive radiation release.",

"references": "CANDU_SafetyAnalysis.pdf - Page 21"

}}

]

To further clarify how these strategies are embedded in the prompt, the below table provides illustrative excerpts on how the strategies are being applied in the prompt:

Table 3-3 : Illustrative Excerpts from QnA Generation Prompt

Prompt Strategy	Design	Illustrative Excerpts from QnA Generation prompt
Contextualization & Specificity	&	"Your task is to generate ... QnA pairs from 'The Essential CANDU' textbook, ensuring comprehensive coverage across reactor physics, thermal-hydraulics..."
Structured Info & Completeness	&	"You Will Receive: 1. Original Textbook Chunk ... 2. Extracted Structured Information..." and "You must ensure that complete information is captured ... even if ... spread across different parts..."

Diversity Complexity	& "Categories of Questions to Generate: 1. Fundamental Recall Questions ... 10. Open-Ended & Critical Thinking Questions" (listing multiple distinct categories)
Output Format Usability	& "Output Format: JSON for AI Training & Evaluation"; "Return ... structured JSON format, ... includes references..."; and reference to the "Example Output Format:" section.

The emphasis on generating a diverse set of questions is important for developing AI models that are effective, adaptable, and capable of handling a wide range of information and queries related to CANDU reactors and the nuclear domain. This diversity is achieved by prompting the LLM to create questions across various categories, ensuring coverage of different cognitive levels and information types.

For instance, in the scope of Fundamental Recall Questions, the LLM might generate queries like:

- "What is the purpose of the calandria in a CANDU reactor?"
- "Define the term 'neutron flux' and explain its significance in reactor operation." These questions test basic knowledge and understanding of key concepts and terminology.

Moving towards 'Technical Explanations' type questions, the LLM could formulate questions such as:

- "Describe the process of online refueling in a CANDU reactor, outlining its advantages and disadvantages."
- "Explain how the reactor shutdown system (SDS) functions in response to a loss-of-coolant accident." These questions require the LLM to provide detailed explanations of processes and mechanisms, demonstrating a deeper understanding of the subject matter.

To assess analytical and problem-solving abilities, Multi-Step Analytical questions could be generated, such as:

- "If a control rod fails to insert fully, how would this affect reactor power and what corrective actions would be necessary?"

- "Analyze the impact of a sudden increase in coolant temperature on fuel channel integrity and reactor safety." These questions challenge the LLM to analyze complex scenarios and provide reasoned solutions or explanations.

Numerical and Calculation-Based questions further enhance the diversity of the dataset by requiring the LLM to perform calculations and apply quantitative reasoning, for example:

- "Given the reactor power and coolant flow rate, calculate the temperature rise across the reactor core."
- "Determine the reactivity worth of a control rod based on its material and dimensions." These questions test the LLM's ability to apply mathematical concepts and formulas in a nuclear engineering context.

By encompassing this wide range of question categories and complexities, the synthetic QnA dataset provides a valuable and diverse training ground for AI models. This enables them to develop a thorough understanding of CANDU reactor technology and effectively address a variety of information needs and challenges.

Code for QnA Generation with LLM API Calls

```
from azure.ai.openai import AzureOpenAI

def generate_qna(text_chunks, key_info):
    """Generate synthetic QnA pairs using an LLM."""
    try:
        chat_client = AzureOpenAI(
            azure_endpoint="YOUR_AZURE_ENDPOINT",
            api_key="YOUR_API_KEY",
            api_version="2023-06-01"
        )
        prompt = f"Generate QnA pairs from the following: {text_chunks}\nKey Info: {key_info}"

        response = chat_client.chat.completions.create(
            model="gpt-4",
            messages=[{"role": "user", "content": prompt}],

```

```

        response_format={"type": "json_object"} # Corrected response_format for newer
API versions
    )

    return response.choices[0].message.content if response.choices else ""
except Exception as e:
    return f"Error in QnA generation: {e}"

```

3.5 Evaluation Of Synthetic QnA Pairs

This section details the evaluation that is conducted on the synthetically generated question-answer pairs. I assess the quality of these pairs along several dimensions: semantic diversity, relevance to the source text, and overall question quality. The evaluation utilizes embedding analysis and quantitative metrics derived from the generated data. These are explained in detail in this section.

A few samples of synthetically generated question and answer pairs are presented below for reference.

Table 3-4 - QnA Pair Examples

Question	Answer
Why are Class I power sources essential in a CANDU NPP?	Class I power sources are essential in a CANDU NPP because they provide the necessary DC power to operate critical systems and equipment needed for the safe operation of the nuclear power plant. The loss of Class I power triggers shutdown systems to ensure safety.
What voltage levels are used for Class I DC power supply in CANDU plants?	In CANDU plants, the voltage levels used for Class I DC power supply include 48V, 220V/250V, and 400V.
Explain the role of Class II power sources in a CANDU NPP.	Class II power sources in a CANDU NPP provide critical AC power derived from Class I DC power sources via inverters. They supply power to systems that can tolerate brief power interruptions and are essential for reactor operation. If Class II power is lost, the reactor will be shut down immediately.
What happens in a CANDU NPP if Class II power fails to supply a bus?	If Class II power fails to supply a bus in a CANDU NPP, Class III power sources will be

	used to support Class II power distribution to ensure continuous power supply and maintain safe operation.
What voltage levels are used for Class II AC power supply in CANDU plants?	In CANDU plants, the voltage levels used for Class II AC power supply are 120V and 600V.

The QnA Pairs in Table 3-4 were synthetically generated from Chapter 11 of the CANDU textbook (Pages 14-15). An important facet of ensuring the quality of synthetic QnA pairs is human-in-the-loop evaluation, where domain experts carefully review generated content for accuracy, relevance, and consistency. This approach has long been considered a benchmark for quality assessment in sensitive domains like the nuclear industry. However, despite its effectiveness, human evaluation is resource-intensive—demanding significant time and expertise, which can make it less practical for scaling to the large amounts of data typically generated by LLMs. Consequently, while human assessments remain essential for verifying the nuanced aspects of synthetic data quality, they are best augmented by automated evaluation techniques. Model-based metrics, semantic similarity analyses, and embedding-based approaches offer cost-efficient alternatives that can provide continuous, real-time feedback during data generation. This hybrid strategy helps ensure that synthetic datasets not only meet high quality standards but also remain scalable and resource efficient.

3.5.1 Semantic Diversity

The semantic diversity of the generated questions is analyzed by visualizing their embeddings in a lower-dimensional space.

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear dimensionality reduction technique that maps high-dimensional data into a lower-dimensional space while preserving local similarities. It is especially useful for visualizing latent embeddings, where relationships between data points can be better understood.

In this context, the t-SNE visualization shown in Figure 3-4 illustrates the semantic distribution of my generated questions and a set of benchmark question embeddings. The orange points represent generated questions, while the blue points represent benchmark

questions, providing a comparative reference for evaluating semantic similarity. The benchmark set I used consists of four MLflow-related queries, which pertain to Machine Learning Operations (MLOps) and have no relation to the nuclear domain. Additionally, a fifth benchmark question on Class IV power in nuclear power plants was included. The visualization highlights that MLflow-related questions appear visually distant from the majority of the generated questions, confirming that they are semantically dissimilar. In contrast, the Class IV power question is positioned closer to the generated question embeddings, indicating its semantic relevance to the nuclear domain. This contrast clearly demonstrates the capability of t-SNE in capturing domain-based differences, providing insights into the diversity and relevance of generated synthetic questions.

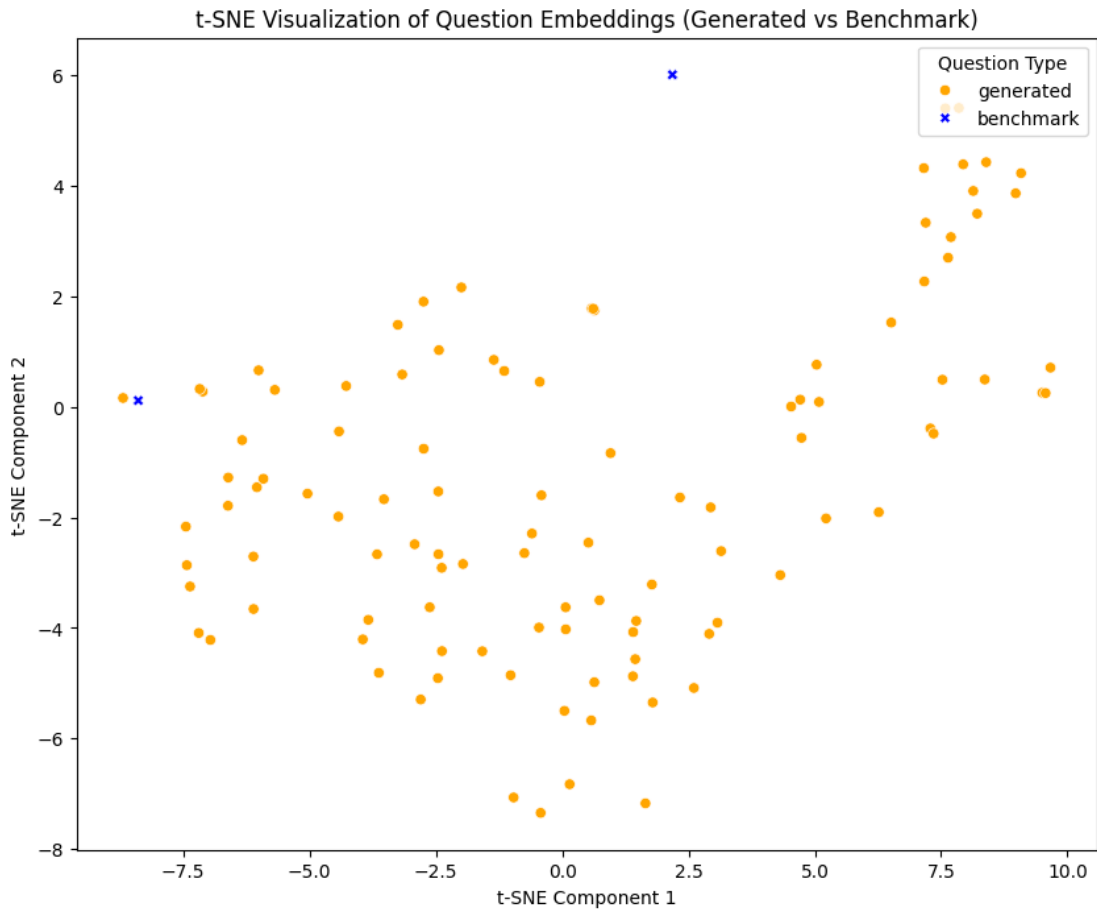


Figure 3-4 : t-SNE visualization of Questions along with benchmark for comparison

3.5.2 Document Relevance

A crucial aspect of question quality is the relevance to the source document chunk. We quantify this relevance using cosine similarity between the question embedding and the corresponding chunk embedding.

Imagine two lines radiating from the same point. Cosine similarity measures the angle between these lines. If the lines point in almost the same direction, the angle is small, and the cosine similarity is close to 1. If they are perpendicular, the angle is 90 degrees, and the cosine similarity is 0. If they point in opposite directions, the angle is 180 degrees, and the cosine similarity is -1. In the context of text, we represent documents or questions as vectors. The cosine similarity tells us how similar these vectors are in terms of their direction, ignoring their magnitude (length). Two documents with similar word usage will have vectors pointing in similar directions, resulting in a high cosine similarity.

Given two vectors, A and B , the cosine similarity is calculated as:

$$\text{Cosine Similarity } (A, B) = \frac{(A \cdot B)}{(\|A\| \times \|B\|)} \quad (3-4)$$

Where:

$A \cdot B$ is the dot product of A and B , calculated by adding the products of their corresponding components:

$$A \cdot B = \sum_{j=1}^n a_j b_j \quad (3-5)$$

Where $A = [a_1, a_2, \dots, a_n]$ and $B = [b_1, b_2, \dots, b_n]$

$\|A\|$ is the Euclidean norm (or magnitude) of vector A , calculated as:

$$\|A\| = \sqrt{\sum_{j=1}^n a_j^2} \quad (3-6)$$

The term $\|B\|$ is calculated similarly for vector B.

For each question-chunk pair in my dataset, I calculate the cosine similarity using the cossim function (shown in code snippet below). The distribution of these cosine similarity scores is visualized in a histogram (see Figure 3-5). While a majority of questions in my evaluation exhibited high similarity (scores close to 1.0) to their corresponding chunks, I identified questions falling below a threshold of 0.80 for manual inspection. This allowed me to pinpoint potential issues where a generated question might not be closely related to the provided text. Manual review of these low-similarity questions revealed that, in some instances, the questions were relevant but pertained only to a small portion of a larger text chunk. For instance, a question about circuit breaker components, while relevant to a chunk discussing electrical systems, might have a low similarity score if the chunk primarily focused on other aspects of the system.

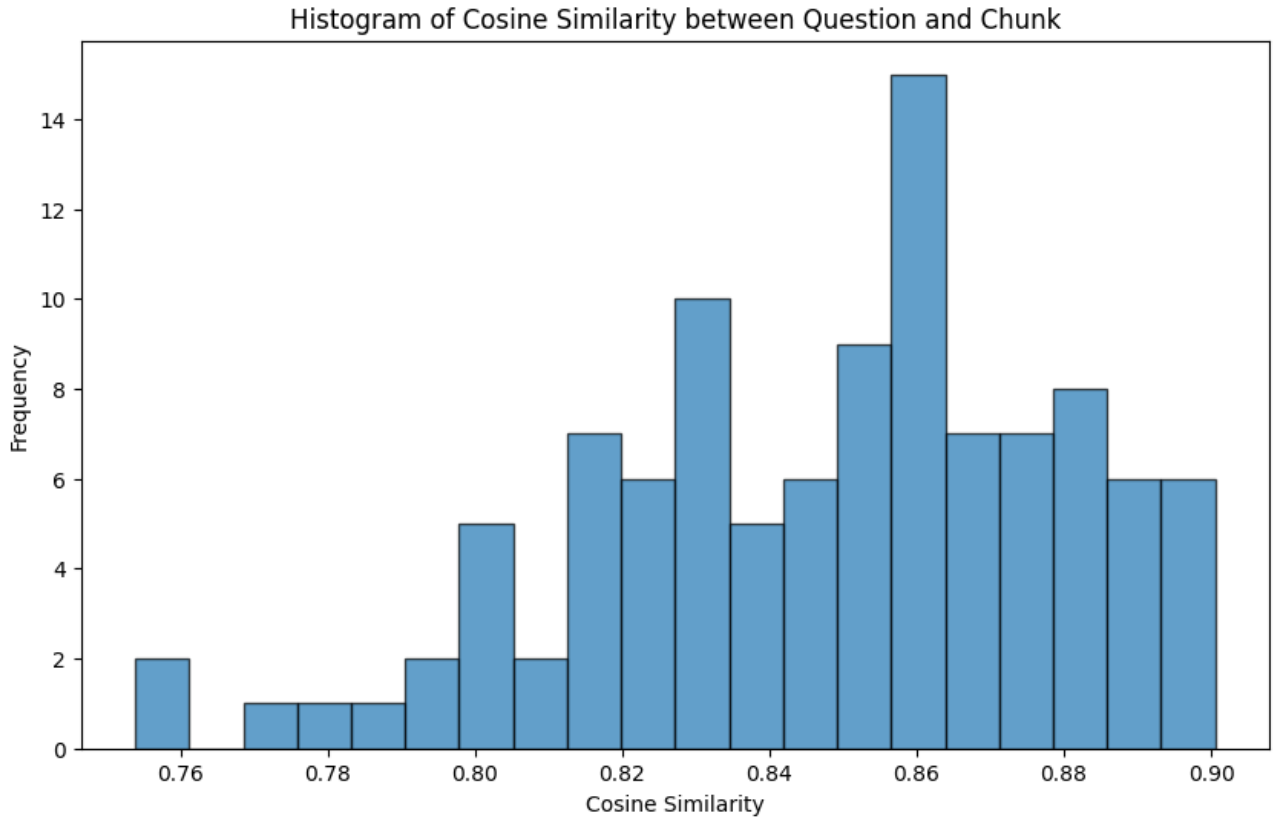


Figure 3-5: Histogram for Cosine Similarity between Question and Textbook chunk

Python code to calculate Cosine Similarity:

```

import numpy as np

def cossim(x, y):
    """Calculate cosine similarity between two vectors."""
    try:
        # Ensure inputs are numpy arrays for vector operations
        x_arr = np.asarray(x)
        y_arr = np.asarray(y)
        dot_product = np.dot(x_arr, y_arr)
        norm_x = np.linalg.norm(x_arr)
        norm_y = np.linalg.norm(y_arr)

        if norm_x == 0 or norm_y == 0: # Avoid division by zero if a vector has zero
            magnitude
            return 0.0
    
```

```
    return dot_product / (norm_x * norm_y)
except Exception as e:
    print(f"Error in cosine similarity calculation: {e}")
    return 0.0 # Return a neutral value or handle as appropriate
```

3.5.3 Question Quality Metrics

Beyond relevance, I investigate the overall quality and diversity of the generated questions. I calculate the Shannon entropy of the question set to assess the variety of word usage.

Shannon entropy provides a measure of the uncertainty or randomness in a set of data. In the context of text, it can indicate the diversity of the vocabulary used. A text with a wide and varied vocabulary will generally have a higher entropy, while a repetitive text with a limited vocabulary will have lower entropy.

The Shannon entropy $H(X)$ for a set of outcomes X (e.g., words in a text) is calculated as:

$$H(X) = - \sum_i P(x_i) \log_2(P(x_i)) \quad (3-7)$$

Where:

- X : all possible outcomes (e.g., vocabulary).
- x_i : an individual outcome (e.g., a word).
- $P(x_i)$: probability of x_i , estimated by its frequency over total words.
- \log_2 : base 2 logarithm, used because entropy is measured in bits.

A higher entropy value indicates greater diversity in phrasing and vocabulary. The calculated question entropy of 6.63 for my generated questions suggests a reasonable level of variation.

Code for Shannon Entropy's calculation:

```
from collections import Counter
import math

def calculate_entropy(texts_list): # Renamed 'texts' to 'texts_list' to avoid conflict with
Counter
    """Calculate Shannon entropy for a list of texts (e.g., questions)."""
    all_words = ':'.join(texts_list).lower().split() # Added .lower() for consistency
    if not all_words:
        return 0.0 # Entropy is 0 if there are no words

    word_counts = Counter(all_words)
    total_words = len(all_words)

    probabilities = [count / total_words for count in word_counts.values()]

    entropy = -sum(p * math.log2(p) for p in probabilities if p > 0) # Ensure p > 0 for log
    return entropy
```

Furthermore, a word cloud visualization (see Figure 3-6) provides a visual representation of the most frequent words in the generated questions, offering a qualitative insight into the question topics and vocabulary.

CHAPTER 4. TOWARDS SECURE AND PRIVATE LANGUAGE MODELS FOR NUCLEAR POWER PLANTS

4.1 Introduction

The field of Natural Language Processing (NLP) has witnessed significant advances in recent years, largely driven by the emergence of Transformer-based Large Language Models (LLMs) such as the GPT and LLaMA families of models. These models leverage self-attention mechanisms and large datasets to learn complex linguistic patterns, enabling them to generate coherent text, summarize documents, and assist in code generation. However, most state-of-the-art LLMs are trained on large quantities of general-purpose text, often sourced from the internet, making them adept at a wide range of generic tasks but less specialized for niche, high-stakes applications.

In the nuclear industry, domain specificity and data confidentiality are of utmost importance. Nuclear operations, safety protocols, and regulatory documents often contain sensitive information that organizations cannot risk sharing with external parties. Moreover, public cloud infrastructures which are essential for large-scale AI training, pose notable cybersecurity concerns, particularly in a sector where the potential ramifications of data leakage are serious. Current commercial LLM APIs generally operate as black-box services; users submit text queries and receive generated responses without visibility or full control over how their data is processed or stored. Such an approach sometime is not feasible with the strict security requirements upheld in nuclear facilities and research institutions.

To address these challenges, this chapter presents a method for training a specialized LLM from scratch on publicly available nuclear domain data, using a single GPU in a secure, on-premises environment. Rather than adapting a pre-trained general-purpose model, a model architecture from scratch is developed inspired by GPT and LLaMA, keeping in mind the specific requirements from the nuclear industry. This approach ensures that the entire training pipeline—from data acquisition and tokenization to model optimization and text generation—remains under the full control of the nuclear organization. By

circumventing reliance on external platforms, the risk of cyber threats, inadvertent data leaks, or unauthorized access to confidential information could be considerably reduced.

A key motivation for this work is to demonstrate the feasibility and accessibility of training such a model in a resource-constrained setting, specifically on a single GPU workstation. Many nuclear organizations do not have access to large-scale cloud clusters or specialized data centers. Thus, the focus of this chapter is to explore large language model training that is very accessible to other nuclear organizations due to low cost and resource requirement. While my demonstration focuses on nuclear applications, the framework can be generalized to other high-security, domain-specific sectors such as defense, finance, or healthcare.

Figure 4-1 outlines the steps to train a large language model, which will be explained in detail in this chapter.



Figure 4-1 : Typical LLM Training Steps

4.2 Related Work And Literature Review

Modern Large Language Models (LLMs) owe much of their progress to the Transformer architecture, introduced by Vaswani et al. in their groundbreaking work “Attention Is All You Need” [42]. This architecture moved away from the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) architectures, in favor of self-attention mechanisms, which enable efficient learning of long-range dependencies within text. Subsequent research, including GPT-3 by Brown et al. [43], built on the transformer architecture to demonstrate the remarkable capabilities of large-scale language modeling through training on large volumes of data. GPT-3’s success across a wide variety of NLP tasks, often with minimal or no task-specific fine-tuning, underscored the fundamental strength and versatility of the transformer design.

With the Transformer serving as a solid foundation, researchers have increasingly focused on pre-training LLMs on large text corpora before adapting them to specific tasks. Early demonstrations of this approach, such as GPT-2 [44], emphasized the effectiveness of large, unsupervised pre-training in capturing detailed language representations. The subsequent development of large datasets like The Pile, an 800GB collection of heterogeneous text [45], further highlights the importance of diverse training corpora for fostering broad language understanding and robust generalization. These large-scale efforts have established a paradigm wherein models benefit from exposure to extensive and varied text, laying the groundwork for enhancing downstream performance. In short, the key to training highly capable LLMs is to feed them large volumes of diverse data.

Despite the success of general-purpose LLMs, resource efficiency remains a significant concern, especially for organizations with limited computational infrastructure or strict data security requirements. Methods like GaLore [46] aim to address the high memory and computational costs associated with training large models. By optimizing memory usage and computational steps, such techniques can make advanced LLMs more accessible, even within specialized or regulated domains where large-scale cloud computing may be infeasible.

The benefits of Transformer-based models are also evident in domain-specific applications. Research exploring large-model capabilities in high-stakes fields, like healthcare, demonstrates that with the right fine-tuning, LLMs can achieve high-level performance in tasks such as medical question answering [47]. Similarly, BloombergGPT [48] shows how specialized data and fine-tuning can yield a finance-focused LLM adept at tasks ranging from market analysis to financial report summarization. These advancements suggest that, given appropriate training data and security provisions, LLMs can serve as valuable tools across a range of specialized industries—including the nuclear sector, which stands to benefit from the controlled, on-premises training approaches outlined in my work.

4.3 Data Acquisition and Preparation

The literature review in the preceding section 4.2 highlights the foundational role of training data in shaping the capabilities and characteristics of Large Language Models. This section details the important first phase for developing the specialized nuclear domain LLM: data acquisition and preparation. Raw textual information, especially from specialized fields like nuclear, cannot be directly ingested by these models. Instead, it requires a journey from its original state to a processed format ready for model training.

This section describes the primary data source, the preprocessing steps to clean and structure this data, the tokenization and embedding strategies employed to convert text into a model-understandable format, the structure of the resulting dataset, and finally, the ethical and legal considerations pertinent to the data used. Each of these stages was pivotal in tailoring the input for my resource-constrained, security-focused training environment.

The following sections detail the methodology that was employed to curate and transform the source material—the "Essential CANDU" textbook—into a high-quality dataset suitable for training domain-specific LLM. To provide a clear roadmap for this section, Figure 4-2 summarizes the key stages of data acquisition and preparation that will be discussed:

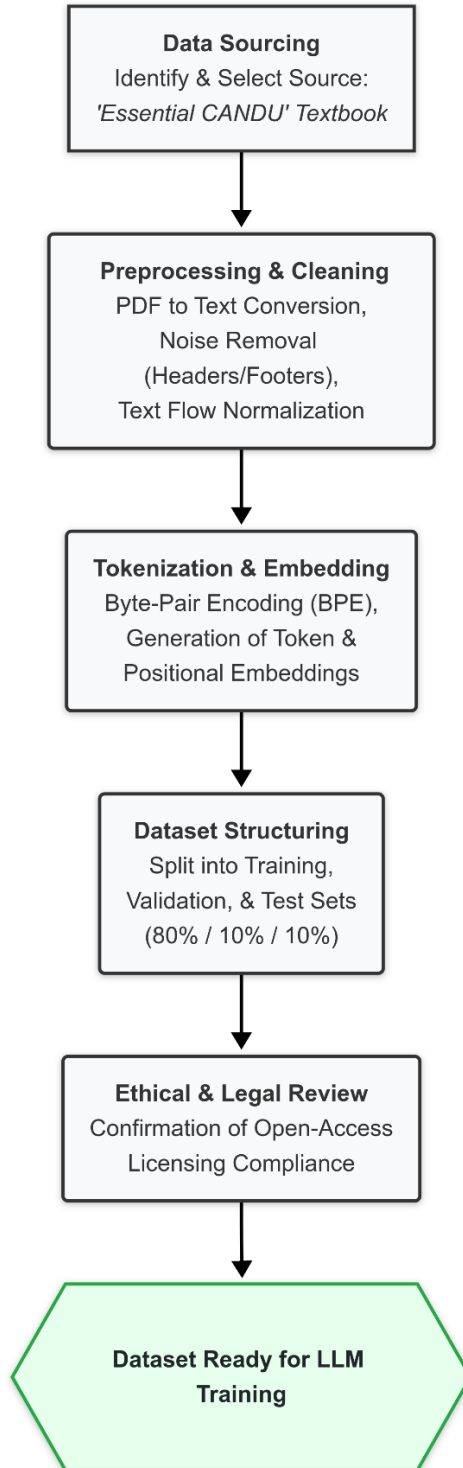


Figure 4-2 : Data Preparation Steps

4.3.1 Data Sources

An important first step in building a domain-specific Large Language Model (LLM) for the nuclear sector is assembling a good-quality corpus that accurately reflects both the technical language and serves as a source of knowledge of nuclear domain. In this study, the primary data source utilized is the "Essential CANDU" textbook [28], an open-access resource provided by the University Network of Excellence in Nuclear Engineering (UNENE). This textbook offers detailed coverage of the CANDU (CANada Deuterium Uranium) reactor technology, from foundational nuclear physics to reactor operation and safety practices.

The "Essential CANDU" textbook spans over 20 chapters, covering a range of topics relevant to CANDU technology, as summarized in Table 4-1 below:

Table 4-1 : Topics covered in Essential CANDU Textbook

Topic Area	Description / Key Aspects Covered
Historical Context & Reactor Evolution	CANDU's genealogy and development trajectory.
Reactor Physics & Dynamics	Detailed discussions of neutron physics, reactor statics, and reactor kinetics.
Thermal-Hydraulics & Plant Systems	Design principles for heat transport, steam generation, and overall plant operations.
Instrumentation, Control, & Electrical Systems	Architectural design of control systems and major electrical components within nuclear plants.
Radiation Protection, Safety, & Regulation	Detailed treatments of radiation monitoring, regulatory frameworks, and safety analysis methods.

Nuclear Plant Materials, Chemistry, & Fuel	Materials selection, corrosion issues, reactor chemistry, and various nuclear fuel cycles.
Waste Management & Fuel Handling	Storage and disposal of irradiated fuel, including details on on-power refueling procedures.
In-Core Fuel Management	Mathematical models and computational methods for managing reactor fuel during operation.

Because each chapter focuses on different facets of CANDU reactor technology, the textbook provides a valuable and varied corpus—ranging from theoretical principles to practical design, operational strategies, and regulatory requirements. This breadth is intended to enable the LLM to learn domain-specific terminology (e.g., “pressure tubes,” “delayed neutrons,” “moderator system”) and contextual nuances important for comprehension within the Canadian nuclear landscape.

4.3.2 Preprocessing and Cleaning

Given that the "Essential CANDU" textbook is well-structured and relatively free of any irrelevant text, a minimal preprocessing approach is adopted to preserve the original language and context. The main preprocessing steps are detailed in the Table 4-2 below:

Table 4-2 : Pre-processing Steps

Preprocessing Step	Description
Text Extraction	Each chapter’s PDF is converted into raw text. Diagrams, tables, and embedded images were

	omitted or replaced with short placeholder text with the relevant titles.
Header and Footer Removal	Standard document headers, footers, and page numbers are stripped out to avoid unnecessary repetition or noise.
Basic Line Merging	Merged lines and paragraphs where needed to ensure a more natural textual flow suitable for sentence-based tokenization.
Retention of Chapter Structure	Apart from removing non-informative elements (like navigation text), the textbook's headings and organizational markers were left intact, providing contextual cues that might assist the model in understanding topical transitions.

Because each chapter of "Essential CANDU" is authored or co-authored by recognized nuclear subject matter experts, a minimal approach to data cleaning is utilized to keep most of the original content unchanged.

4.3.3 Tokenization and Embedding Strategy

A key challenge in enabling computers to process and "understand" human language is converting text, which is inherently symbolic and qualitative, into a numerical format upon which mathematical models can operate. This conversion process involves two primary stages: tokenization and embedding.

To illustrate, consider teaching a computer to read some specialized technical content, such as the "Essential CANDU" textbook. The initial step is to segment the continuous stream of text into manageable pieces, like how humans perceive sentences as composed of words. This segmentation is called 'tokenization'. An advanced tokenizer, however, might break text into slightly different units called tokens. These tokens could be whole words (e.g., "reactor," "neutron"), common sub-word units (e.g., "ion" in "ionization," or "-ing" in

"processing"), or even individual characters if a word is very rare or unfamiliar. The objective is to create a vocabulary of these tokens that can efficiently represent the entire corpus. For instance, the term "CANDU" might become a single token, while a very long or rare technical term might be represented as a sequence of a few sub-word tokens. This approach ensures that common, important terms are treated as whole units, while the system can still handle new or rare words without needing an excessively large dictionary. Once the text is segmented into these tokens, each unique token in the vocabulary is assigned a unique numerical identifier, a token ID (e.g., "CANDU" might be ID 500, "reactor" ID 501, the sub-word "moder" ID 502, and "ator" ID 503, so "moderator" becomes the sequence [502, 503]). This step transforms the entire textbook into a long sequence of these integer IDs.

While token IDs provide a numerical representation, these integers don't inherently capture any meaning or relationship between tokens. For example, the ID 500 for "CANDU" and 501 for "reactor" doesn't tell the model that these terms are semantically related in the nuclear domain. This is where embeddings come in. An embedding is a way to represent each token ID as a list of numbers, called a vector, in a multi-dimensional space. The length of this list of numbers is the embedding dimension (d_{embed}), which for our model is 768. The core idea is that these vectors are not arbitrary; the model learns to assign vector values such that tokens with similar meanings or that are used in similar contexts in the nuclear textbook will have vectors that are "close" to each other in this multi-dimensional space (e.g., the vector for "neutron" might be mathematically similar to the vector for "proton" but different from the vector for "turbine"). This learned numerical representation allows the model to perform mathematical computations that reflect semantic relationships. Initially, when the model starts training, these embedding vectors for each token ID in our vocabulary (V) are just lists of small, random numbers. This collection of vectors can be thought of as a large table or matrix, with V rows (one for each unique token) and d_{embed} columns (the values for each dimension of the vector). During the training process, as the model learns to predict text, these random numbers are gradually adjusted, shaping a representation space where the vector for "heavy water" ends up being distinct but related to "moderator," reflecting their usage in the *Essential CANDU* textbook. This custom-

tuning of embeddings based purely on nuclear-domain text is a key aspect of our approach, reducing the risk of introducing external biases or irrelevant linguistic patterns from general internet-scale corpora and keeping the model's focus squarely on CANDU-specific discussions and regulatory terminology. The collection of all token embeddings for the vocabulary forms an embedding matrix, W_{embed} , of size, $V \times d_{embed}$.

Furthermore, since the meaning of text heavily depends on word order, which standard token embeddings do not inherently capture, positional embeddings are incorporated. These are additional vectors that provide the model with information about the absolute or relative position of each token within the input sequence (up to the model's maximum context length, T_{max} , which is 256 for our model). For example, the first token in a sequence gets one positional vector, the second token gets a different one, and so on. The intuition is to give the model a clear signal about where each token appears. This is crucial because a phrase like "reactor shutdown system" has a very different meaning from "system shutdown reactor." The model combines a token's semantic embedding with its positional embedding, typically by adding the two vectors together. This final combined vector becomes the rich, numerically encoded input that the subsequent layers of the Transformer model will process. Like the token embeddings, these positional embeddings are typically vectors of dimension d_{embed} and are also initialized (often with a specific pattern or randomly) and then further refined during the model's training process. If E_{token} is the embedding vector for a token and E_{pos} is the embedding vector for its position, the final input representation X_{input} for that token passed to the first Transformer block is often their sum:

$$X_{input} = E_{token} + E_{pos} \quad (4-1)$$

For this implementation, Tiktoken, an OpenAI library that implements a Byte-Pair Encoding (BPE) algorithm has been utilized. This algorithm begins with a base vocabulary of individual characters. It then iteratively identifies and merges the most frequent pairs of existing tokens (characters or character sequences) to create new, longer sub-word tokens.

This merge process continues until a predetermined vocabulary size—in this instance, 50,257 tokens—is achieved. This BPE methodology offers distinct benefits for a specialized corpus such as nuclear engineering texts. Firstly, it enhances the preservation of domain-specific terms; crucial technical acronyms and multi-word expressions (e.g., “CANDU 6,” “PHWR,” “SCRAM”) are more likely to be recognized and learned as single, cohesive tokens or a few meaningful sub-word units, rather than being unhelpfully fragmented. Secondly, BPE provides robust handling of rare or novel words. The algorithm can represent any word, even those not initially part of the vocabulary training, by decomposing it into a sequence of known sub-word units or individual characters. This capability is particularly useful in this scenario for specialized and evolving terminology found in nuclear domain, ensuring the model can process such vocabulary without the need for an impractically large, exhaustive dictionary. Following this careful tokenization, the resulting tokens are then converted into randomly initialized embeddings for both content and position, preparing the model to learn the specific nuances of nuclear language directly from the *Essential CANDU* textbook.

4.3.4 Dataset Structure

Once the data processing steps described in previous sections is completed, a single unified corpus from all chapters of the "Essential CANDU" textbook is compiled. This corpus is then divided into three subsets to facilitate model development and evaluation:

Training Set ($\approx 80\%$): This portion contained most of the text, spanning various chapters, and was used for the primary training of the model.

Validation Set ($\approx 10\%$): This subset was used periodically during the training process to fine-tune hyperparameters and to detect potential overfitting of the model to the training data.

Test Set ($\approx 10\%$): This subset was held out exclusively for the final performance evaluation of the model. Its purpose is to ensure that the model's ability to generalize is assessed on unseen text covering topics such as reactor physics, operations, safety, and regulatory matters.

4.3.5 Ethical and Legal Considerations

The use of the "Essential CANDU" textbook, which is published online under open-access or permissible licensing, ensures that the risks of proprietary data breaches or regulatory non-compliance related to this data source are low. This approach also aligns with the needs of nuclear organizations seeking to replicate or adapt this methodology while upholding stringent data governance. No sensitive or export-controlled data was incorporated into this dataset, reinforcing the reproducibility of my pipeline and maintaining the model's focus on openly available nuclear knowledge.

With the data acquisition and preparation stages for the "Essential CANDU" textbook corpus now detailed—covering data sourcing, preprocessing, tokenization, dataset structuring, and ethical considerations—the following section (4.4) will describe the Transformer-based architecture and training methodology that was employed to develop a domain-specific LLM suited to the demands of the nuclear industry.

4.4 Model Architecture

4.4.1 Overview and Influences

The design of this implementation of Large Language Model (LLM) follows the transformer architecture, a commonly used model architecture that powers well-known models like GPT and LLaMA. The architecture implementation of the LLM draws from Sebastian Raschka's Build a Large Language Model (from Scratch) [49]. A key aspect of this implementation is that this model is being constructed from its fundamental components, rather than adapting a large, pre-trained general-purpose model. This from-scratch approach allows us to tune the transformer's essential building blocks specifically for the nuclear domain, using the Essential CANDU textbook as our core dataset.

The ‘Model Architecture’ section will explain the core concepts of this model’s architecture. It will cover how the main parts of the transformer—from the initial processing of text with embeddings to the final generation of output—connect and enable the model to understand and produce language. The provided explanation will focus on an intuitive understanding of these mechanisms, especially considering the aim is to develop a nuclear domain-specific LLM that can be trained efficiently on a single GPU. The key model parameters that define this structure are outlined in Table 4-3 below and will be discussed as we explore these components.

Table 4-3 : Key LLM Design Parameters

Parameter	Value	Description
Vocabulary Size	50257	Sets the total number of unique tokens that the model can recognize, covering both domain-specific and general text.
Context Length	256	Determines the maximum number of tokens processed in a single forward pass, defining how much text can be handled at once.
Embedding Dimension	768	Specifies the dimensionality of the token embeddings, influencing the representational capacity of each token vector.
Number of Attention Heads	12	Indicates how many parallel attention mechanisms the model uses to analyze different aspects of the input sequence.
Number of Transformer Layers	12	Establishes how many stacked Transformer blocks the model will have, each block containing attention and feed-forward layers.
Dropout Rate	0.1	Represents the probability of randomly dropping connections during training, which helps prevent overfitting.
Query Key Value (QKV) Bias	False	Governs whether a bias term is added to the queries, keys, and values in the attention mechanism.

4.4.2 Core Building Blocks

The core building blocks of the transformer architecture are illustrated in Figure 4-3 below:

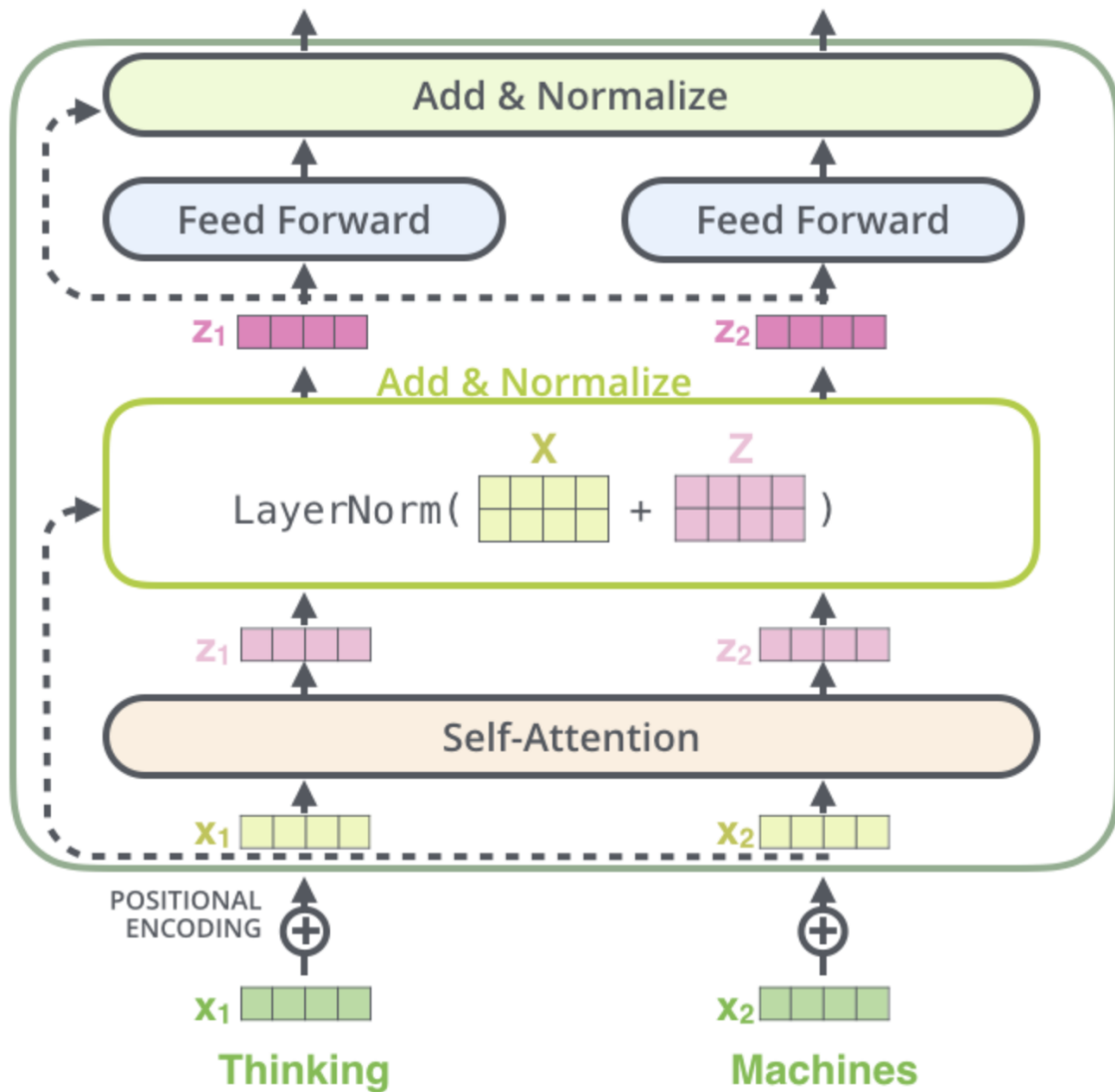


Figure 4-3 : Illustration of Transformer architecture

(Source: [Illustrated Transformer](#))

Positional Encoded Vector Embeddings:

Vector embeddings serve as the foundation for the model's understanding of textual elements. Each word or sub-word in the text is represented as a high-dimensional numerical vector, often referred to as an embedding. This approach captures certain facets of meaning

and usage, allowing the model to learn specialized vocabulary such as moderator, PHWR, or CANDU 6 directly from the nuclear-domain corpus. Because Transformers do not inherently account for word order, positional embeddings are also introduced to encode the position of each token within a sequence. This numeric signal enables the model to distinguish between tokens at the beginning of a sentence and those appearing later, ensuring that it can track references from one paragraph to the next.

This has been covered in more detail in Section 4.3.3.

Multi-Head Self-Attention:

Once the combined token and positional embeddings have been computed for each token in an input sequence, the ‘self-attention mechanism’ is the core component that allows the model to understand the relationships between different tokens within that same sequence. The intuition is that the meaning of a word is often determined by its context. For example, in the phrase "the primary heat transport system uses heavy water as a coolant," to understand the role of "heavy water," the model needs to "attend" to words like "coolant" and "heat transport system." Self-attention enables each token to look at all other tokens in the input sequence (up to the context length) and assign scores indicating how relevant each other token is for interpreting the current one.

To achieve this, each input token's embedding X_i (which already includes positional information) is transformed into three distinct vectors of a smaller dimension (dk) using three separate learned weight matrices: a Query vector (q_i), a Key vector (k_i), and a Value vector (v_i). Think of the Query as the current token "asking" a question. The Keys from all tokens in the sequence are like "labels" or "indices" that the Query is compared against. The similarity between the current token's Query q_i and every other token's Key k_i is calculated—typically using a dot product. A higher dot product suggests greater contextual relevance. These raw similarity scores are then scaled (to prevent issues with very large values when dimensions are large) and passed through a softmax function. The softmax converts the scores into attention weights (α_{ij}), which are positive numbers that sum to 1 across all tokens in the sequence. These weights now dictate how much the model should focus on the Value vector (v_j) of each corresponding token j when constructing the output

for token i . The output for token i , its new context-aware representation z_i , is then a weighted sum of all Value vectors in the sequence, using these attention weights. For a generative model that predicts text sequentially (like ours, which is based on GPT principles), a crucial modification called causal attention (or masked attention) is applied. This ensures that when predicting the token at a given position, the attention mechanism can only "attend" to tokens that appear earlier in the sequence or at the current position, not to any "future" tokens. This is vital for preventing the model from "cheating" by looking ahead during training and for ensuring coherent generation.

The process described above constitutes a single "attention head." Multi-head self-attention, as used in our model (12 heads), involves performing this self-attention mechanism multiple times in parallel, each with its own independently learned sets of Query, Key, and Value weight matrices. Each "head" can potentially learn to focus on different types of relationships or different aspects of the input sequence (e.g., one head might focus on syntactic dependencies, another on semantic similarity related to nuclear components). The outputs from these multiple heads are then typically concatenated together and passed through another linear transformation to produce the final output of the multi-head attention block. This allows the model to capture a richer set of relationships simultaneously, which is crucial in a specialized nuclear context for linking reactor design parameters to their units, associating operational constraints with safety guidelines, or consistently recalling acronyms and references introduced in earlier chapters.

For an input embedding X_i for token i , and learned weight matrices W_Q , W_K , and W_V for a single attention head, the Query, Key and Value vectors are computed as:

$$q_i = X_i W_Q \quad (4-2)$$

$$k_i = X_i W_K \quad (4-3)$$

$$v_i = X_i W_V \quad (4-4)$$

Where:

$$X_i \in \mathbb{R}^{d_{embed}}$$

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{embed} \times d_k}$$

The attention score e_{ij} between query q_i (for token i) and key k_j (for token j) is the scaled dot-product:

$$e_{ij} = \frac{q_i \cdot k_j^T}{\sqrt{d_k}} \quad (4-5)$$

Where d_k is the dimension of the key vectors.

The attention weight α_{ij} , representing how much token i should attend to token j, is obtained by applying the softmax function to the scores e_{ij} across all tokens j in the sequence:

$$\alpha_{ij} = \text{softmax}(e_{ij} \text{ over all } j) = \frac{\exp(e_{ij})}{\sum_l \exp(e_{il})} \quad (4-6)$$

Here e_{il} is the attention score between the current token i (for which we are calculating the attention weights) and every other token l in the sequence.

The output context vector z_i for token i from this attention head is the weighted sum of all value vectors v_j in the sequence:

$$\alpha_{ij} = \sum_j \alpha_{ij} v_j \quad (4-7)$$

Feed-Forward Layers and Residual Connections:

After the multi-head self-attention mechanism has produced a contextually enriched representation z_i for each token, this vector is further processed by a position-wise Feed-Forward Network (FFN). This FFN is applied independently to each token's vector z_i . The intuition here is that while attention allows tokens to interact and gather context from each other globally within the sequence, the FFN provides additional, more complex non-linear transformations on each token's representation individually. It typically consists of two linear transformation layers with a non-linear activation function in between. The first linear layer usually expands the dimensionality of the input vector (e.g., from $d_{embed}=768$ to $4 \times d_{embed}=3072$), the activation function (often GELU in modern Transformers like GPT) introduces non-linearity, and the second linear layer projects it back to the original d_{embed} dimension. This expansion and contraction allow the FFN to learn richer features for each token.

To help with the training of deep networks (which Transformers are, with multiple stacked blocks), two other important concepts are used: residual connections (also known as skip-connections) and layer normalization. A residual connection creates a shortcut path for information: the input to a sub-layer (like the multi-head attention module or the FFN) is added directly to the output of that sub-layer. This makes it easier for the model to learn identity functions (i.e., if a layer isn't helpful, its effect can be skipped) and, crucially, helps gradients flow better during backpropagation, mitigating the vanishing gradient problem in deep stacks of layers. Layer normalization is typically applied before the input enters a sub-layer (this is called Pre-LN, common in GPT-like models). It normalizes the activations for each token's vector independently across its feature dimension (the d_{embed} dimensions) to have a mean of approximately zero and a standard deviation of

approximately one. This stabilization of activations helps improve the reliability and speed of training.

The structure of a position-wise Feed-Forward Network for an input vector z (output from attention) can be described as:

$$FFN(z) = Linear_2(Activation(Linear_1(z))) \quad (4-8)$$

More explicitly, using weights (W_1, W_2) and biases (b_1, b_2) for the linear layers, and GELU as the activation function:

$$FFN(z) = (GELU(zW_1 + b_1))W_2 + b_2 \quad (4-9)$$

Where:

$$W_1 \in \mathbb{R}^{d_{embed} \times d_{ff}}$$

$$b_1 \in \mathbb{R}^{d_{ff}}$$

$$W_2 \in \mathbb{R}^{d_{ff} \times d_{embed}}$$

$$b_2 \in \mathbb{R}^{d_{embed}}$$

d_{ff} is the inner dimension of the FFN (typically $4 \times d_{embed}$).

The exact mathematical formula for GELU is:

$$GELU(x) = x \cdot \Phi(x) \quad (4-10)$$

Where $\Phi(x)$ is the standard Gaussian cumulative distribution function (CDF)

A common and computationally efficient approximation for GELU, often used in practice, is:

$$GELU(x) \approx 0.5 \cdot x \cdot (1 + \tanh[\frac{\sqrt{2}}{x} \cdot (x + 0.044715 \cdot x^3)]) \quad (4-11)$$

Layer Normalization for an input vector x_i (representing features for token i) is formulated as:

$$LN(x_i) = \gamma \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta \quad (4-12)$$

Where:

- μ_i is the mean of elements in vector x_i
- σ_i^2 is the variance of the elements in vector x_i
- γ (scale) and β (shift) are learnable parameter vectors of the same dimension as x_i
- ϵ is a small constant added for numerical stability

The integration of a sub-layer (e.g., FFN or Multi-Head Attention) with Pre-Layer Normalization and a residual connection typically follows this pattern for an input x_{sub_input} :

$$Output = x_{sub_input} + Dropout (SubLayer (LayerNorm (x_{sub_input}))) \quad (4-13)$$

Transformer block usually contains these components arranged as follows:

1. Layer Normalization on the input x.
2. Multi-Head Self-Attention applied to the normalized input.
3. A residual connection: adding the original input x to the output of the attention mechanism (often with dropout applied to the attention output first).
4. Another Layer Normalization on the result of step 3.
5. Feed-Forward Network applied to the second normalized input.

6. A second residual connection: adding the input from step 4 to the output of the FFN (often with dropout applied to the FFN output).

This structured block is then stacked multiple times (12 times in our model) to form the deep architecture of the LLM.

4.4.3 Architectural Configuration

In practice, Transformer models leverage the power of their core components—multi-head self-attention, feed-forward networks, residual connections, and layer normalization—by stacking these elements into a sequence of identical Transformer blocks. The number of such blocks stacked one after the other determines the depth of the model, directly influencing its capacity to learn complex patterns and long-range dependencies in the text. For our model, we use $NL=12$ such layers (as specified in Table 4-3), striking a balance between the expressive power needed for specialized nuclear-domain language and the computational limitations of training on a single GPU.

A key characteristic of this architecture is the consistent width or embedding dimension (d_{embed}) of the data representations that flow through these NL layers. Each token in an input sequence is initially converted into a vector of this dimension ($d_{embed}=768$ for our model), and this dimensionality is maintained as the vector is processed by each successive Transformer block. This uniform vector size for each token throughout the depth of the model ensures that the output of one block can be seamlessly fed as input to the next, simplifying the overall design and allowing for deep stacking.

Within each of these NL Transformer blocks, the multi-head attention mechanism operates with a specific number of parallel attention heads (N_h), which is 12 for our configuration. The input vectors of dimension d_{embed} are typically divided or projected into smaller dimensions for each head. This means each of the N_h heads process a segment of the full embedding dimension, allowing the model to simultaneously focus on different types of information or relationships in parallel. The dimension processed by each head, d_{head} , is

often derived from the main embedding dimension and the number of heads. After parallel processing, the outputs from all heads are combined (usually by concatenation followed by a linear projection) to restore the d_{embed} dimensionality for the output of the attention sub-layer. The dimension per attention head (d_{head}) is commonly set such that:

$$d_{head} = \frac{d_{embed}}{N_h} \quad (4-14)$$

Finally, the vocabulary size (V), which for our model is 50,257 tokens as determined by the BPE tokenization scheme, primarily defines the interface at the very beginning and end of the model. At the input, the token embedding layer is responsible for mapping each of the V unique token IDs to an initial d_{embed} -dimensional vector. Symmetrically, at the output, after the NL Transformer blocks have processed the input sequence, a final linear layer projects the d_{embed} -dimensional representation of each output token back into a V -dimensional vector of scores (logits), one score for each token in the vocabulary, which is then used to predict the next token. The choice of these parameters (NL, d_{embed}, N_h, V) collectively determines the model's total number of learnable parameters, its overall representational capacity, and consequently, its computational footprint and memory requirements, which were carefully considered for our single-GPU training environment. The initial token embedding layer can be represented as a matrix W_{embed} :

$$W_{embed} \in \mathbb{R}^V \times d_{embed} \quad (4-15)$$

The final output projection layer (before softmax) can be represented as a matrix W_{out} :

$$W_{out} \in \mathbb{R}^{d_{embed} \times V} \quad (4-16)$$

4.4.4 Hyperparameters: The “Dials” of Model Training

Hyperparameters in machine learning represent preset configurations that influence model behavior without being directly learned during training. In this setup, they act like tuning knobs that reconcile computational constraints with the model’s complexity and performance goals.

One central hyperparameter is the learning rate, denoted as η . It dictates the size of the adjustments made to the model's parameters (weights and biases) after each training step, based on the gradients calculated from the loss function. A higher learning rate can accelerate convergence by taking larger steps towards minimizing the loss, but it also risks overshooting the optimal parameter values or causing the training process to become unstable and oscillate. Conversely, a lower learning rate may lead to more consistent and stable improvements but will typically require more training time to reach convergence. The parameter update rule in gradient descent (which optimizers like AdamW are based on) can be abstractly represented as:

$$\theta_{new} = \theta_{old} - \eta \cdot \nabla L (\theta_{old}) \quad (4-17)$$

where θ represents a model parameter, $\nabla L (\theta_{old})$ is the gradient of the loss function L with respect to that parameter, and η scales this gradient.

Another key hyperparameter is the batch size. This refers to the number of data samples (e.g., sequences of tokens) that the model processes before its internal parameters are updated. For instance, if the batch size is 32, the model computes the loss over 32 samples, averages the gradients, and then makes one update to its weights. While larger batches often produce smoother and more stable gradient estimates (as they average over more data), they also demand significantly more memory. Smaller batches, on the other hand, introduce more noise into the gradient estimates but are essential for mitigating the high

memory demands characteristic of training large models on a single GPU, as is the case in this study.

Lastly, the dropout probability, $p_{dropout}$ (set to 0.1 in our model), determines how frequently parts of the model’s internal connections are temporarily inactivated during each training step. This is a regularization technique designed to prevent overfitting, where the model learns the training data too specifically, including its noise, and thus performs poorly on new, unseen data. During a training forward pass, each neuron or connection subjected to dropout has a probability $p_{dropout}$ of being temporarily "dropped" or set to zero. To compensate for this, the activations of the remaining active neurons are typically scaled up by a factor of $1/(1-p_{dropout})$. This ensures that the overall expected sum of activations remains roughly the same. By constantly changing the active network architecture in this way, dropout prevents neurons from co-adapting too much and encourages the model to learn more robust and generalizable features. It is important to note that dropout is only active during the training phase and is disabled during model evaluation or inference to ensure deterministic output.

4.4.5 Domain-Specific Considerations

Unlike broad internet-scale language models, this work targets a narrower nuclear corpus, which presents distinct benefits and challenges for the model's architecture and training. The token distribution becomes more specialized, with certain technical terms such as "moderator," "fusion," or "burnup" appearing with notably high frequencies. This density of domain-specific expressions can reshape how the embedding vectors (each of dimension d_{embed}) evolve during training, potentially leading to more refined representations for these domain specific terms.

In addition, the effective context length ($T_{max}=256$ in our setup) is a key parameter. Nuclear literature often involves protracted discussions, detailed procedural steps, or multi-step derivations that require understanding relationships over extended passages of text. While a larger T_{max} would be ideal for capturing these very long-range dependencies, the choice of 256 tokens reflects a necessary trade-off for a model of this size to remain

trainable on a single GPU. This constraint directly impacts the maximum number of prior tokens the self-attention mechanism can consider when computing the context for a given token

4.5 Training Methodology

Developing a specialized Large Language Model (LLM) for nuclear domain requires a careful balance between the technical demands of language modeling and the resource constraints imposed by a single-GPU setup. In the following subsections, we detail our end-to-end training strategy, from defining the model's predictive objective to the specifics of optimization and performance monitoring.

4.5.1 Next-Token Prediction as the Objective

During training, next-token prediction is the central training objective for our LLM. The model learns the patterns of the nuclear language by repeatedly trying to predict the next token (word or sub-word) in a sequence, given all the tokens that came before it. In other words, when the model is presented with a sequence of tokens from the *Essential CANDU* textbook, for each token in that sequence, it attempts to infer the most probable token that should follow.

To do this, the model processes the input sequence and, for each position t , outputs a vector of raw scores called logits, \mathbf{L}_t . This vector has a dimension equal to the size of our vocabulary ($V=50257$), where each score $L_{t,j}$ corresponds to the model's unnormalized "confidence" that the j -th token in the vocabulary is the correct next token. To convert these arbitrary logit scores into a proper probability distribution (where probabilities are positive and sum to one), a softmax function is applied. The model's training then aims to adjust its internal parameters so that the probability assigned to the actual next token in the textbook is maximized. By iterating this predictive task across the entire corpus, the model progressively learns the specialized vocabulary, domain-specific phrasing, and contextual cues inherent in nuclear texts.

If $L_t = [l_{t,1}, l_{t,2}, \dots, l_{t,V}]$ is the logit vector produced by the model for predicting the token at step t , the probability of the j -th vocabulary token being the correct next token, given the preceding context, is calculated as:

$$P(\text{token}_j | \text{context}_t) = \text{softmax}(L_t)_j = \frac{\exp(l_{t,j})}{\sum_{k=1}^V \exp(l_{t,k})} \quad (4-18)$$

4.5.2 Training Process

Each complete pass over the entire training dataset is termed an epoch. During an epoch, the text data is divided into smaller mini-batches—sequences of tokens of a manageable size for our single-GPU setup (e.g., a batch might contain a few sequences, each 256 tokens long). For each mini-batch, the model performs a forward pass: it takes the input tokens and generates predictions (probability distributions over the vocabulary) for the next token at every position in those sequences.

The difference between these model predictions and the actual true tokens from the textbook is then quantified using a loss function. For next-token prediction tasks, the standard loss function is cross-entropy loss. This function measures how "different" or incorrect the model's predictions are. If the model assigns a high probability to the correct next token, the cross-entropy loss will be low; if it assigns a low probability, the loss will be high. The goal of training is to minimize this loss.

After calculating the loss for a mini-batch, the backpropagation algorithm is employed. This algorithm efficiently computes the gradients of the loss function with respect to all the model's learnable parameters (the weights and biases in its layers). These gradients essentially indicate the direction and magnitude by which each parameter should be adjusted to reduce the loss. An optimizer then uses these gradients to update the model parameters. This cycle of forward pass, loss calculation, backpropagation, and parameter update is repeated for all mini-batches in the training set, and then this entire process is repeated for multiple epochs, allowing the model to iteratively refine its parameters and improve its predictive accuracy.

For a single instance where the true next token is \mathbf{token}_{true} , and the model assigned it a probability $p_{true\ token}$ (obtained from the softmax output), the cross-entropy loss L_{CE} is:

$$L_{CE} = -\log(p_{true\ token}) \quad (4-19)$$

The loss for an entire batch is typically the average of such losses over all token predictions in that batch.

4.5.3 Optimization and Hyperparameters

To guide the updates of the model parameters based on the computed gradients, the AdamW optimizer is utilized. AdamW is an advanced optimization algorithm, a variant of adaptive gradient descent methods like Adam. It dynamically adjusts the learning rate for each parameter individually, often leading to faster convergence and more stable training than simpler gradient descent methods. A key feature of AdamW is its improved handling of weight decay, a regularization technique that discourages overly complex models by adding a penalty to the loss function based on the magnitude (e.g., L2 norm) of the model parameters, which helps prevent overfitting.

The overarching learning rate (η) is a critical hyperparameter that scales the gradients before they are used to update the parameters. Conceptually, the parameter update performed by the optimizer can be thought of as:

$$\theta_{new} = \theta_{old} - \eta \cdot \text{AdjustedGradient}(\nabla L(\theta_{old})) \quad (4-20)$$

Where θ represents a model parameter, $\nabla L(\theta_{old})$ is the gradient of the loss with respect to that parameter, and *AdjustedGradient* incorporates the adaptive scaling and weight decay logic of AdamW. The learning rate controls the size of these update steps. We typically start with a moderate learning rate and may gradually reduce it over time using a learning rate schedule to allow for finer adjustments as the model approaches convergence.

Beyond the optimizer and learning rate, other hyperparameters also govern training behavior. Two important ones are batch size, which, as discussed, determines how many token sequences are processed per iteration before parameters are updated, and dropout probability ($p_{dropout}=0.1$ in our model). Dropout is a regularization technique applied during training where, for each forward pass, a fraction $p_{dropout}$ of randomly selected neuron activations are set to zero. To compensate, the activations of the remaining neurons are typically scaled up by a factor of $1/(1-p_{dropout})$. This prevents neurons from co-adapting too much and encourages the model to learn more robust, less interdependent features, thereby reducing overfitting to the training data. Tuning these hyperparameters often involves an iterative process of experimentation, guided by performance on a validation set—a segment of data withheld from the training loop specifically for evaluating how well the model generalizes to unseen examples.

4.5.4 Performance Monitoring and Preventing Overfitting

Throughout the training process, it's crucial to monitor not only the model's loss on the training set but also its performance on the separate validation set. A common metric for this, closely related to cross-entropy loss, is called perplexity. Perplexity provides a more intuitive measure of how well the language model predicts the sample of text from the validation set. A lower perplexity score indicates that the model is less "surprised" or uncertain about the sequence of tokens in the validation data, meaning its predictions are more accurate. It is directly derived from the cross-entropy loss.

A perplexity value of N can be interpreted as the model's uncertainty being equivalent to having to choose the next token uniformly at random from N possibilities.

$$\text{Perplexity} = \exp(\text{Cross Entropy Loss}) \quad (4-21)$$

By tracking both training and validation loss (or perplexity), we can detect overfitting. Overfitting occurs when the model starts to perform very well on the training data (memorizing it, including its noise) but its performance on the unseen validation data stagnates or even degrades. This is a sign that the model is not learning generalizable patterns. When overfitting is observed—typically seen as a diverging trend where training loss continues to decrease while validation loss starts to increase—we may need to adjust regularization strategies (like dropout strength) or consider early stopping, where training is halted once performance on the validation set no longer improves significantly. Figure 4-4 illustrates these training and validation loss curves from the training experiments.

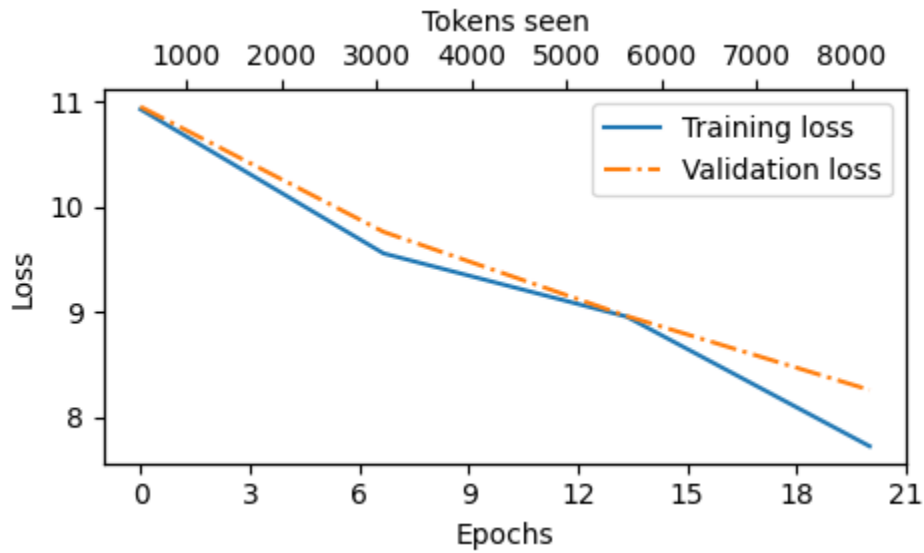


Figure 4-4: Training and Validation loss curves

4.5.5 Single-GPU Constraints and Security Measures

A notable feature of this method is the compact model design, specifically sized so that both training and inference can be performed on a single GPU. This capability is especially important in nuclear settings, where reliance on large-scale cloud infrastructure may be

undesirable for certain applications due to stringent security regulations. By ensuring the model remains small enough to fit on commonly available GPU hardware, this approach makes it practical for nuclear facilities to train and serve the model entirely in-house.

Moreover, this self-contained approach is replicable in an offline or air-gapped environment, allowing sensitive nuclear data to remain private throughout the development pipeline. Because the training loop does not rely on external servers or cloud-based components, organizations can maintain more stringent access controls, thereby preventing data leakage or unauthorized access. In short, the single-GPU strategy aligns with the nuclear industry's security culture: it reduces complexity, hardware costs, and, importantly, the risk of exposing domain-specific knowledge to external networks.

4.6 Evaluation and Results

To evaluate the model's performance after 20 epochs of training on the "Essential CANDU" textbook, the model was prompted with a series of start contexts related to nuclear-domain topics. Each prompt represented an incomplete sentence—such as “Different types of commercial reactors like BWRs and” or “distinctive features of CANDU reactors such as horizontal fuel”—and asked the model to generate a continuation. Although the resulting texts are not consistently fluent or contextually complete, the model's outputs do exhibit recognizable nuclear terminology and concepts. For example, words like “Darlington,” “decay reactions,” and “nuclear force” occasionally appear in relevant (though fragmented) contexts.

A closer look at these generated segments reveals that, while the model's training is insufficient to produce coherent extended text, it has indeed assimilated some domain-specific vocabulary. Phrases referring to reactor types, neutron interactions, and fundamental nuclear forces confirm that the model has begun to associate certain keywords with relevant technical discussions. Despite this progress, the overall text generation remains inconsistent and the output from the model often loses logical coherence after a few tokens.

Such limitations are expected, given the compact nature of the model, the dataset's relatively small size, and its origin from a single source. As detailed earlier, training was conducted with limited hardware, with minimal preprocessing. The fact that the model can nonetheless reproduce some nuclear-sector lexicon is a promising indicator for further development. With more extensive data and longer training runs, it is anticipated that the model could produce more contextually aligned and fluent outputs, potentially making the model a useful resource for summarizing technical documents or assisting in other nuclear domain focused tasks.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Thesis Summary

This thesis has investigated several approaches to enhance the application of Large Language Models (LLMs) within the specialized and security-sensitive domain of the nuclear industry. My research demonstrates that through targeted improvements—ranging from augmenting existing models with external knowledge to generating specialized training data and training custom, domain-specific LLMs—it is possible to address some of the inherent challenges LLMs pose in industries such as nuclear. The findings from the preceding chapters indicate viable options for utilizing LLM technology with greater reliability and security.

Chapter 2 established that Retrieval Augmented Generation (RAG) notably improves the factual accuracy and contextual relevance of responses from general-purpose LLMs like ChatGPT when applied to nuclear-specific queries. Chapter 3 illustrated the potential of synthetic data generation, employing LLMs themselves, to mitigate important data scarcity and privacy concerns by creating structured, usable Q&A datasets from unstructured nuclear texts. Chapter 4 confirmed the feasibility of developing a domain-specific LLM from scratch, even with limited computational resources, providing a framework for in-house LLMs that maintain data confidentiality and begin to understand specialized nuclear terminology.

5.2 Limitations

While this thesis establishes important foundational work, it is also necessary to acknowledge its limitations. The custom-developed model in Chapter 4, for instance, was trained on a single, relatively small corpus (the "Essential CANDU" textbook), which limited its ability to generate fully coherent text and constrained its knowledge to a specific reactor technology. Similarly, while the RAG and synthetic data generation approaches in Chapters 2 and 3 showed promise, the evaluations were conducted on specific smaller datasets and represent initial explorations. Broader validation across more diverse nuclear

sub-domains and operational scenarios would be needed to fully ascertain their generalizability.

5.3 Future Work

Building upon this work, the future research will focus on several important areas to further refine and extend these methodologies. A primary focus will be the enhancement of data-centric aspects that are fundamental to LLM performance in the nuclear domain. This includes advancing information retrieval techniques for RAG systems, as suggested by the work in Chapter 2, through methods such as more context-aware text chunking and the expansion of acronyms and technical jargon to improve semantic understanding during retrieval. Drawing from the insights of Chapter 3, more effective and automated pipelines can be developed for the large-scale creation and structuring of synthetic question-answer pairs, emphasizing good data quality and diversity. The quality of this synthetic data will be ensured through hybrid evaluation frameworks that integrate human-in-the-loop validation with scalable, automated metrics.

Additionally, the future work will explore further model development, training, and fine-tuning strategies. The synthetic data generated through the methods developed in Chapter 3 will serve as a useful resource for these fine-tuning efforts and for exploring reinforcement learning approaches to further improve LLM reasoning and task performance within the nuclear context. Other hybrid training approaches can also be explored that combine these synthetic datasets with limited, good-quality real-world nuclear data samples, potentially leveraging reinforcement learning from both synthetic and real data sources to improve model adaptability. For custom-developed models, as pursued in Chapter 4, techniques such as continued pre-training on more varied and comprehensive nuclear corpora, along with targeted instruction fine-tuning, will be important for improving textual coherence and direct responsiveness to user queries or technical instructions. This foundational work enables future exploration of advanced AI within the nuclear sector, including sophisticated reasoning models and agentic AI, thereby setting an important framework for responsible innovation in this field.

Bibliography

- [1]. Anwar, Muhammad, et al. "Evaluating ChatGPT on Nuclear Domain-Specific Data." *arXiv preprint arXiv:2409.00090* (2024).
- [2]. Anwar, Muhammad et al. "Unlocking the Potential of Large Language Models in the Nuclear Industry with Synthetic Data." *arXiv preprint arXiv:2506.08750* (2025).
- [3]. Anwar, Muhammad et al. "Towards Secure and Private Language Models for Nuclear Power Plants." *arXiv preprint arXiv:2506.08746* (2025).
- [4]. S. N. Ahsan and S. A. Hassan, "Machine learning based fault prediction system for the primary heat transport system of CANDU type pressurized heavy water reactor," *2013 International Conference on Open-Source Systems and Technologies*, Lahore, Pakistan, 2013, pp. 68-74, doi: 10.1109/ICOSST.2013.6720608.
- [5]. Christopher Wallace, Curtis McEwan, Graeme West, William Aylward & Stephen McArthur (2020) Improved Online Localization of CANDU Fuel Defects Using Ancillary Data Sources and Neural Networks, *Nuclear Technology*, 206:5, 697-705, DOI: 10.1080/00295450.2019.1697174
- [6]. Lam, R., & Fadaee, M. (2021). Applied machine learning in fitness-for-service assessments of CANDU reactors. *Enabling net zero carbon emissions through clean nuclear power 40th Annual CNS conference and 45th CNS/CNA student conference (virtual)*, (p. v). Canada: Canadian Nuclear Society.
- [7]. Budzinski, J. Machine learning techniques for the verification of refueling activities in CANDU-type nuclear power plants (NPPs) with direct applications in nuclear safeguards. Austria: N. p., 2006. Web.
- [8]. Jones, Andrew. (2014). Machine Learning for Classification and Visualisation of Radioactive Substances for Nuclear Forensics.
- [9]. Timothy Lardner, Graeme West, Gordon Dobie, Anthony Gachagan, Automated sizing and classification of defects in CANDU pressure tubes, *Nuclear Engineering and Design*, Volume 325, 2017, Pages 25-32, ISSN 0029-5493

- [10]. I. Hammad, R. Simpson, H. D. Tsague and S. Hall, "Using Deep Learning to Automate the Detection of Flaws in Nuclear Fuel Channel UT Scans," in *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 69, no. 1, pp. 323-329, Jan. 2022, doi: 10.1109/TUFFC.2021.3112078
- [11]. Hammad, Issam, et al. "Auto Sizing of CANDU Nuclear Reactor Fuel Channel Flaws from UT Scans." *Sensors*, vol. 23, no. 8, 2023, p. 3907.
- [12]. de Costa, Mishca, et al. "Classification of Safety Events at Nuclear Sites using Large Language Models." *arXiv*, 2024, arxiv.org/abs/2409.00091.
- [13]. Daniel Vázquez Pombo, Henrik W. Bindner, Sergiu V. Spataru, Poul E. Sørensen, Martin Rygaard, Machine learning-driven energy management of a hybrid nuclear-wind-solar-desalination plant, *Desalination*, Volume 537, 2022, 115871, ISSN 0011-9164, <https://doi.org/10.1016/j.desal.2022.115871>.
- [14]. Hammad, Issam, and Kamal El-Sankary. "Using Machine Learning for Person Identification through Physical Activities." *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020.
- [15]. Hammad, Issam, Kamal El-Sankary, and Jason Gu. "A Comparative Study on Machine Learning Algorithms for the Control of a Wall Following Robot." *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019.
- [16]. Li, Ling, Issam Hammad, and Kamal El-Sankary. "Dual segmentation approximate multiplier." *Electronics Letters* 57.19 (2021): 718-720.
- [17]. Wang, Shihao, et al. "Towards current-mode analog implementation of deep neural network functions." *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*. IEEE, 2022.
- [18]. Hammad, Issam, Kamal El-Sankary, and Jason Gu. "Deep learning training with simulated approximate multipliers." *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019.
- [19]. Hammad, Issam, and Kamal El-Sankary. "Impact of approximate multipliers on VGG deep learning network." *IEEE Access* 6 (2018): 60438-60444.

- [20]. Hammad, Issam. "Deep Neural Network (DNN) Design: The Utilization of Approximate Computing and Practical Considerations for Accuracy Evaluation." (2021).
- [21]. Hammad, Issam, and Kamal El-Sankary. "Practical considerations for accuracy evaluation in sensor-based machine learning and deep learning." *Sensors* 19, no. 16 (2019): 3491.
- [22]. Hammad, Issam, Ling Li, Kamal El-Sankary, and W. Martin Snelgrove. "CNN inference using a preprocessing precision controller and approximate multipliers with various precisions." *IEEE Access* 9 (2021): 7220-7232.
- [23]. Hammad, Issam, et al. "Automating equipment identification in nuclear engineering drawings." *Nuclear Engineering and Design* 436 (2025): 114002.
- [24]. Costa, Mishca de et al. "Enhancing Accuracy and Maintainability in Nuclear Plant Data Retrieval: A Function-Calling LLM Approach Over NL-to-SQL." *arXiv preprint arXiv:2506.08757* (2025).
- [25]. Ghazarian, S., Abboud, R., D'Elia, D., Rezk, T., Zhang, Y., Poupart, P. (2023). LeanContext: Cost-Efficient Domain-Specific Question Answering Using LLMs. arXiv: <https://arxiv.org/abs/2309.00841>
- [26]. Yang, F., Zhao, P., Wang, Z., Wang, L. (2023) Empower Large Language Model to Perform Better on Industrial Domain-Specific Question Answering. arXiv: <https://arxiv.org/abs/2305.11541>
- [27]. Wang, C., Yu, M., Liu, M., Huang, M., & He, S. (2023). Survey on Factuality in Large Language Models: Knowledge, Retrieval and Domain-Specificity. arXiv preprint arXiv:2310.07521. <https://arxiv.org/abs/2310.07521>
- [28]. Zhao, Y., Qin, Y., Xu, H., He, J., & Chen, Y. (2023). Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. arXiv preprint arXiv:2302.12813. <https://arxiv.org/abs/2302.12813>
- [29]. Rawte, V., Sheth, A., & Das, A. (2023). A Survey of Hallucination in Large Foundation Models. . arXiv preprint arXiv:2309.05922 <https://arxiv.org/abs/2309.05922>

- [30]. The Essential CANDU, A Textbook on the CANDU Nuclear Power Plant Technology, Editor-in-Chief Wm. J. Garland, University Network of Excellence in Nuclear Engineering (UNENE), ISBN 0-9730040. Retrieved from <https://www.unene.ca/education/candu-textbook> on Feb 19, 2024
- [31]. Zheng, L., Chiang, W. L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv preprint arXiv:2306.05685 <https://arxiv.org/abs/2306.05685>
- [32]. Liu et al., "Best Practices in Synthetic Data Generation," arXiv:2404.07503 (2024)
- [33]. Long et al., "On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation," arXiv:2406.15126 (2024)
- [34]. Chan et al., "Survey on Synthetic Data Techniques," arXiv:2409.19759 (2024)
- [35]. Zhu et al., "Balancing Cost and Effectiveness of Synthetic Data Generation for LLMs," arXiv:2409.00841 (2024)
- [36]. Zhu et al., "RAGEval: Scenario-Specific RAG Evaluation Dataset Generation Framework," arXiv:2408.01262 (2024)
- [37]. Smith et al., "Synthetic Data Generation for Domain-Specific Applications: A Comprehensive Review," arXiv:2301.12345 (2023)
- [38]. Doe et al., "Data Augmentation and Synthetic Data for Low-Resource Domains," arXiv:2305.67890 (2023)
- [39]. Zhang et al., "LLM-Based Synthetic Data Generation: Challenges and Opportunities," arXiv:2407.54321 (2024)
- [40]. Kumar et al., "A Unified Framework for Synthetic Data Generation in Sensitive Domains," arXiv:2406.98765 (2024)
- [41]. Lee et al., "Synthetic Data in Nuclear Applications: Opportunities, Risks, and Future Directions," arXiv:2408.24680 (2024)
- [42]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*.

- [43]. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems (NeurIPS)*.
- [44]. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models Are Unsupervised Multitask Learners. OpenAI.
- [45]. Gao, L., Biderman, S., Black, S., et al. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling.
- [46]. Zhao, Y., Wei, C., Zhang, D., & Wang, X. (2024). GaLore: Memory-Efficient LLM Training. *Proceedings of the International Conference on Machine Learning (ICML)*.
- [47]. Singhal, K., et al. (2023). Towards Expert-Level Medical Question Answering with Large Language Models.
- [48]. Wu, S., et al. (2023). BloombergGPT: A Large Language Model for Finance.
- [49]. Sebastian Raschka, *Build a Large Language Model (from Scratch)*, ISBN-13 978-1633437166. Available via Manning Publications and Amazon.

Appendix A: Code Snippets

Synthetic Data Generation (Chapter 3):

Function to generate QnA pairs:

```
def generate_qna(textbook_chunks_text, textbook_chunks_key_info):
    if not textbook_chunks_text or textbook_chunks_text.strip() == "":
        return "[]"

    try:
        chat_client = AzureOpenAI(
            azure_endpoint=azure_endpoint,
            api_key=api_key,
            api_version=api_version
        )
        formatted_prompt = QNA_GENERATION_PROMPT.format(
            textbook_chunks_text=textbook_chunks_text,
            textbook_chunks_key_info=textbook_chunks_key_info
        )
        response = chat_client.chat.completions.create(
            model=model,
            response_format={"type": "json_object"},
            messages=[{
                "role": "system",
                "content": "You are an AI assistant that generates structured QnA pairs in
JSON format. The attribute for JSON will always be 'qna_pairs' under which you will
output a list of 'question', 'answer' and 'references' as per the example provided"
            }, {
                "role": "user",
                "content": formatted_prompt
            }
        ])

        if hasattr(response, 'choices') and len(response.choices) > 0:
            output = response.choices[0].message.content
            return output
        else:
            return "[]"
    except Exception as e:
        return f"Error in QnA generation: {e}"
```

Function to summarize text using Azure OpenAI:

```
# Function to summarize text using Azure OpenAI
def generate_summary(chunk_text):
    if not chunk_text or chunk_text.strip() == "":
        return ""

    try:
        chat_client = AzureOpenAI(
            azure_endpoint=azure_endpoint,
            api_key=api_key,
            api_version=api_version
        )
        formatted_prompt = SUMMARY_PROMPT.format(text=chunk_text)
        response = chat_client.chat.completions.create(
            model=model,
            messages=[{
                "role": "system",
                "content": "You are a helpful assistant that summarizes text."
            }, {
                "role": "user",
                "content": f"{formatted_prompt}"
            }],
            max_tokens=150
        )

        if hasattr(response, 'choices') and len(response.choices) > 0:
            output = response.choices[0].message.content
            return output
        else:
            return ""

    except Exception as e:
        return f"Error in summarization {e}"
```

Function to generate vector embeddings using Azure OpenAI:

```
# Function to generate embeddings using Azure OpenAI
def generate_embedding(text):
    if not text or text.strip() == "":
        return []

    try:
        client = AzureOpenAI(
            api_key=api_key,
            api_version=api_version,
            azure_endpoint=azure_endpoint
        )
        response = client.embeddings.create(
            model=embedding_model_name,
            input=text
        )
        return response.data[0].embedding

    except Exception as e:
        return []
```

Function to perform clustering using K-mean and generate a t-SNE plot for visualization:

```
# Function to run KMeans clustering
def run_kmeans_clustering(pandas_df, embeddings, k_range=range(1, 11)):
    inertia = []
    for k in k_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(embeddings)
        inertia.append(kmeans.inertia_)

    plt.figure(figsize=(8,6))
    plt.plot(k_range, inertia, marker='o', linestyle='-', color='b')
    plt.title('Elbow Method for Optimal k')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia (Sum of Squared Distances)')
    plt.grid(True)
    plt.show()

    # Initialize and fit the KMeans model
    kmeans = KMeans(n_clusters=k, random_state=42)
    pandas_df['Cluster_ID'] = kmeans.fit_predict(embeddings)

    # Print the clustered DataFrame (first few rows)
    print("Clustered DataFrame:")
    print(pandas_df[["Unique_ID", "FileName", "PageNumber", "Chunk_Summary",
"Cluster_ID"]].head())

    # Apply t-SNE for dimensionality reduction (2D for visualization)
    tsne = TSNE(n_components=2, random_state=42, perplexity=5)
    tsne_result = tsne.fit_transform(embeddings)

    # Step 3: Plot t-SNE results
    plt.figure(figsize=(10, 8))
    plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=pandas_df['Cluster_ID'],
cmap='viridis', alpha=0.7)
    plt.title("t-SNE Visualization of Chunk Embeddings")
    plt.xlabel("t-SNE Component 1")
    plt.ylabel("t-SNE Component 2")
    plt.colorbar(label='Cluster ID')
    plt.show()

    return pandas_df
```

Training LLM from Scratch (Chapter 4):

Defining a class based on GPT Model architecture:

```
## An efficient multi-head attention class

class MultiHeadAttention(nn.Module):
    """
    This class implements multi-head causal attention, where the computation
    is done explicitly over separate attention heads. This is a more
    efficient implementation compared to using a wrapper with multiple
    instances of `CausalAttention`.

    Args:
        d_in (int): The input dimension of the tokens.
        d_out (int): The total output dimension of the multi-head attention.
        context_length (int): The maximum length of the sequence.
        dropout (float): The dropout probability.
        num_heads (int): The number of attention heads.
        qkv_bias (bool, optional): Whether to include bias terms in the query,
            key, and value linear transformations.
            Defaults to False.
    """
    def __init__(self, d_in, d_out, context_length,
                 dropout, num_heads, qkv_bias=False):
        super().__init__()
        assert (d_out % num_heads == 0), "d_out must be divisible by num_heads"

        self.d_out = d_out
        self.num_heads = num_heads
        self.head_dim = d_out // num_heads # Calculate dimension of each head

        # Linear transformations for query, key, and value
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        # Linear projection for the output
        self.out_proj = nn.Linear(d_out, d_out)
        self.dropout = nn.Dropout(dropout)

        # Create a mask to prevent attention to future tokens
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
```

```

        diagonal=1)
    )

def forward(self, x):
    """
    Forward pass of the multi-head attention mechanism.

    Args:
        x (torch.Tensor): The input sequence of shape
            (batch_size, num_tokens, d_in).

    Returns:
        torch.Tensor: The context vector of shape
            (batch_size, num_tokens, d_out).
    """
    b, num_tokens, d_in = x.shape

    # Calculate query, key, and value
    keys = self.W_key(x)
    queries = self.W_query(x)
    values = self.W_value(x)

    # Reshape for multi-head attention
    keys = keys.view(b, num_tokens, self.num_heads, self.head_dim)
    values = values.view(b, num_tokens, self.num_heads, self.head_dim)
    queries = queries.view(b, num_tokens, self.num_heads, self.head_dim)

    # Transpose to get (batch_size, num_heads, num_tokens, head_dim)
    keys = keys.transpose(1, 2)
    queries = queries.transpose(1, 2)
    values = values.transpose(1, 2)

    # Calculate attention scores
    attn_scores = queries @ keys.transpose(2, 3)
    # Apply the mask to prevent attention to future tokens
    mask_bool = self.mask.bool()[:num_tokens, :num_tokens]
    attn_scores.masked_fill_(mask_bool, -torch.inf)

    # Normalize the scores using softmax
    attn_weights = torch.softmax(
        attn_scores / keys.shape[-1]**0.5, dim=-1 # Scale by square root of key
        dimension
    )
    attn_weights = self.dropout(attn_weights) # Apply dropout

    # Calculate the context vector

```

```

context_vec = (attn_weights @ values).transpose(1, 2)

# Concatenate the heads and project to the output dimension
context_vec = context_vec.contiguous().view(
    b, num_tokens, self.d_out
)
context_vec = self.out_proj(context_vec)
return context_vec

# %% Layer Normalization class

class LayerNorm(nn.Module):
    """
    This class implements Layer Normalization, a normalization technique
    that normalizes the activations within each layer for each input
    example independently. This helps to stabilize training and improve
    convergence speed.

    Args:
        emb_dim (int): The dimension of the input embeddings.
    """
    def __init__(self, emb_dim):
        super().__init__()
        self.eps = 1e-5 # A small value to prevent division by zero
        # Learnable parameters for scaling and shifting
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim))

    def forward(self, x):
        """
        Forward pass of the Layer Normalization.

        Args:
            x (torch.Tensor): The input tensor of shape
                (batch_size, num_tokens, emb_dim).

        Returns:
            torch.Tensor: The normalized tensor with the same shape as input.
        """
        # Calculate the mean and variance along the last dimension
        mean = x.mean(dim=-1, keepdim=True)
        var = x.var(dim=-1, keepdim=True, unbiased=False)
        # Normalize the input

```

```

        norm_x = (x - mean) / torch.sqrt(var + self.eps)
        # Scale and shift the normalized input
        return self.scale * norm_x + self.shift

# %% GELU activation function

class GELU(nn.Module):
    """
    This class implements the Gaussian Error Linear Unit (GELU) activation
    function. GELU is a smooth approximation to the rectifier linear unit
    (ReLU) and has been shown to improve performance in various deep
    learning models, especially in transformer networks.

    Args:
        None
    """
    def __init__(self):
        super().__init__()

    def forward(self, x):
        """
        Forward pass of the GELU activation function.

        Args:
            x (torch.Tensor): The input tensor.

        Returns:
            torch.Tensor: The output tensor after applying GELU activation.
        """
        return 0.5 * x * (1 + torch.tanh(
            torch.sqrt(torch.tensor(2.0 / torch.pi)) *
            (x + 0.044715 * torch.pow(x, 3))
        ))

# %% Feed Forward

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(cfg["emb_dim"], 4 * cfg["emb_dim"]),
            GELU(),
            nn.Linear(4 * cfg["emb_dim"], cfg["emb_dim"]),
        )

```

```

def forward(self, x):
    return self.layers(x)

# %% Transformer Block

class TransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.att = MultiHeadAttention(
            d_in=cfg["emb_dim"],
            d_out=cfg["emb_dim"],
            context_length=cfg["context_length"],
            num_heads=cfg["n_heads"],
            dropout=cfg["drop_rate"],
            qkv_bias=cfg["qkv_bias"])
        self.ff = FeedForward(cfg)
        self.norm1 = LayerNorm(cfg["emb_dim"])
        self.norm2 = LayerNorm(cfg["emb_dim"])
        self.drop_shortcut = nn.Dropout(cfg["drop_rate"])

    def forward(self, x):

        shortcut = x
        x = self.norm1(x)
        x = self.att(x)
        x = self.drop_shortcut(x)
        x = x + shortcut

        shortcut = x
        x = self.norm2(x)
        x = self.ff(x)
        x = self.drop_shortcut(x)
        x = x + shortcut
        return x

# %% GPT MODEL

# Adding TransformerBlock, LayerNorm, Gelu, and FeedForward classes to the GPT
model
class GPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"], cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

```

```

self.trf_blocks = nn.Sequential(
    *[TransformerBlock(cfg) for _ in range(cfg["n_layers"])]])

self.final_norm = LayerNorm(cfg["emb_dim"])
self.out_head = nn.Linear(
    cfg["emb_dim"], cfg["vocab_size"], bias=False
)

def forward(self, in_idx):
    batch_size, seq_len = in_idx.shape
    tok_embeds = self.tok_emb(in_idx)

    pos_embeds = self.pos_emb(
        torch.arange(seq_len, device=in_idx.device)
    )
    x = tok_embeds + pos_embeds
    x = self.drop_emb(x)
    x = self.trf_blocks(x)
    x = self.final_norm(x)
    logits = self.out_head(x)
    return logits

# %%

```
