

UTRECHT UNIVERSITY

Department of Information and Computing Science

---

Applied Data Science master thesis

**Investigating Open Source Transformer Techniques for  
Question Answering Systems on Cloud Domain: A  
Comparison with ChatGPT3.5-Turbo**

**First examiner:**

Yannis Velegarakis

**Candidate:**

José Enrique Leal Castillo

**Second examiner:**

Enas Khwaileh

**In cooperation with:**

SUE B.V.

June 30, 2023

## Abstract

The advent of transformer models in natural language processing (NLP), particularly extensive transformer models with billions of parameters often referred to as large language models (LLMs), have made significant strides in question-answering (QA) tasks leading to the production of more convincing natural language responses. Despite these advancements, assessing the reliability of QA systems remains challenging due to the intricate nature of language and the diverse array of question types used. This complexity is further compounded by data privacy concerns derived from the handling of training data and user information in proprietary LLMs like ChatGPT, sparking discussions about the need for decentralized and open-source solutions. This research seeks to address these concerns by investigating and comparing the efficiency of various open-source transformer techniques within a practical QA context. The performance of the models is assessed against ChatGPT3.5-turbo using a unique approach. This involves the creation and labeling of a custom evaluation dataset related to the cloud computing domain, specifically Kubernetes technology and the implementation of novel Machine-Trained Evaluation Score (MTES), called Estimated Human Label (EHL). The findings show that among the open-source models studied, the GPT4All model, when combined with optimal input, competes closely with ChatGPT3.5 in code commands and situational type of questions, exceeding a 2.5 EHL score in both cases. Furthermore, it is discovered that merely increasing the input context size proved beneficial only for Flan-T5 and did not consistently enhance the performance of other models. These insights underscore the effectiveness of open-source models in QA tasks, significantly contributing to the advancement and understanding of such systems.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Literature review . . . . .	6
1.2	Theoretical framework . . . . .	7
<b>2</b>	<b>Data</b>	<b>17</b>
2.1	Source of knowledge data . . . . .	17
2.2	Evaluation dataset . . . . .	17
<b>3</b>	<b>Models</b>	<b>18</b>
3.1	Deep Set: RoBERTa-base-Squad2 . . . . .	18
3.2	Google: Flan-t5-xl . . . . .	18
3.3	Nomic AI: GPT4All-LoRa (7B) . . . . .	18
3.4	Open AI: ChatGPT 3.5-turbo . . . . .	19
3.5	Sentence transformer: all-MiniLM-L6-v2 . . . . .	19
<b>4</b>	<b>Methodology</b>	<b>20</b>
4.1	Suggested QA system . . . . .	20
4.2	System evaluation: . . . . .	24
<b>5</b>	<b>Results</b>	<b>28</b>
<b>6</b>	<b>Discussion</b>	<b>32</b>
6.1	Best OS model in the experiment . . . . .	32
6.2	Comparison vs ChatGPT 3.5-turbo . . . . .	32
6.3	Impact of context size on performance . . . . .	32
6.4	Limitations . . . . .	33
<b>7</b>	<b>Conclusion</b>	<b>35</b>
	<b>Appendix A</b>	<b>42</b>
	<b>Appendix B</b>	<b>43</b>

<b>Appendix C</b>	<b>44</b>
<b>Bibliography</b>	<b>62</b>

# 1. Introduction

In recent times, there has been considerable interest in Question Answering (QA) systems, leading to their emergence as a prominent area of study in the field of Natural Language Processing (NLP). These systems have become essential components in applications involving human interaction, such as customer service chatbots and virtual assistants like SIRI or Alexa [1], [2].

The main objective of a QA system is to generate articulate responses in natural language that satisfactorily address the queries presented [3]. However, this objective extends beyond mere provision of answers explicitly stated in the text. As posited by Saeidi *et al.* [4] and Mensio *et al.* [5], numerous practical QA problems necessitate comprehending the text not only for the direct answer but also for deriving the response using the system's background knowledge and guidelines provided within the text. This often requires the system possessing reasoning capabilities and comprehension of the deeper meanings and nuances within a query. This view is further endorsed by Mohnish *et al.* [6] as they note that the raw response from a QA system is often inadequate. Indeed, Strzalkowski *et al.* [7] observe that users value extensive information over a mere direct answer, even when the query is unambiguous.

Recently, modern NLP approaches such as transformer models have been utilized to tackle the Question Answering task [8]–[10]. Unlike traditional sequential models that process input elements one-by-one, transformer's self-attention enables the model to compute the importance of all input sequence elements relative to each other. By doing so, it can capture meaningful relationships between distant elements, which is crucial for handling complex language tasks [11], [12]. Moreover, Research by Izacard *et al.* [13] demonstrates that incorporating supplementary knowledge through retrieval in the input of transformer models can lead to significant QA performance improvements. While there have been attempts to explore the impact of input passage characteristics on the effectiveness of responses generated by modern QA systems [8], this field of research is still in the process of development.

The progress in computational capabilities has ushered in an era of extensive transformer models with millions or even billions of parameters [14]–[16]. Frequently known as Large Language Models (LLMs) [17], these models are trained and fine-tuned on massive datasets, enabling them to produce more convincing and fluid natural language due to their expanded scope [18]–[20]. Proprietary LLMs have become robust solutions for NLP tasks, delivering remarkable results [10], [13], [16], [21]. Specifically, since 2022, OPENAI's ChatGPT 3 and its successors have garnered considerable

---

interest, witnessing one of the fastest growth rates in user bases in consumer applications, with over 100 million active users [22]. However, issues around data privacy have been raised, particularly regarding the training data [23] and how user information is handled [22]. Based on ChatGPT's privacy policy [24], it gathers diverse data including user interactions, browser configurations, IP addresses, and preferred content. It also monitors users' online activities across multiple websites over time. The policy also indicates that in certain circumstances, personal data may be divulged to third parties without prior warning, except where legally required. Additional privacy problems with ChatGPT have been underlined by experts, such as glitches revealing users' conversation histories [25]. The CEO of OpenAI has acknowledged these issues and assured to address them, yet vulnerabilities to cyber and differential privacy attacks have been spotted [26]–[28], endangering users' private information [22]. In light of these issues, it is vital to focus on investigating decentralized methods and building open-source (OS) solutions for NLP tasks that provide superior privacy safeguards.

Assessing the effectiveness of QA systems is a challenging task due to the intricacy of language, the wide range of question types employed for testing, and the possibility of varying acceptable responses [1], [29]. Past studies have mainly concentrated on evaluating QA systems by measuring N-grams metrics between their answers and a designated "gold answer" considered to be the correct response [9]. However, the assessment of QA systems' reliability remains uncertain due to their inherent complexity. Consequently, determining the optimal approach for evaluating QA systems is still unclear [1].

The aim of this research is to investigate and contrast the efficiency of various OS transformer techniques, specifically the Extractive Pre-trained Transformer (EPT), Generative Pre-trained Transformer (GPT), and Text-to-Text Transfer Transformer (T5) within the context of a practical QA task. These models will be evaluated against the most recent publicly available LLM by OpenAI: ChatGPT3.5-turbo. The goal is to determine how well OS models can compete against proprietary models. Additionally, this research is interested in exploring the impact of modifying the input context size on the quality of outputs generated by each model. The guiding research questions (RQ) for this study are:

**RQ1:** Which OS transformer method, EPT, GPT, or T5, demonstrates superior performance when answering the proposed domain questions in overall and by each type of question?

**RQ2:** Can any of the studied OS transformer techniques compete in terms of performance with ChatGPT3.5-Turbo when answering the proposed domain questions in general and by each kind of question?

**RQ3:** Is increasing the size of the input context always beneficial for each method in terms of performance?

The selected domain for simulating a real-world QA task is cloud computing, focusing on questions related to Kubernetes technology. The QA system uses the entire Kubernetes public documentation, enriched by real-time searches on Google, as its primary source of knowledge. To evaluate the performance of each approach, a self-proposed Machine-trained evaluation score (MTES) called estimated human label (EHL) is computed using a Machine Learning (ML) classification model. This model is trained using N-gram-based metrics, which analyze continuous sequences of  $n$  words. Additionally, human experts labeled a self-proposed, carefully balanced dataset that encompasses diverse question categories, including close-ended, open-ended, conceptual, situational, command-based, comparative, procedural, and multiple-answer questions. This research endeavors to contribute to the advancement of QA systems powered by OS models and furnish valuable insights into the factors influencing their performance. By combining human expertise and a MTES, it strives to offer meaningful findings and improve our comprehension of QA systems.

### 1.1 Literature review

A QA system in the domain of information retrieval represents a task within NLP that is designed to offer responses to queries posed by humans in natural language. It achieves this by either leveraging a well-structured database or using a compilation of documents with unstructured data [30]–[33].

The quest for discovering appropriate responses in natural language to provided questions has gained considerable focus in recent times [34]. Nonetheless, this concern is not a recent one, the field of NLP emerged in the 1950s as the intersection of Artificial Intelligence and Linguistics [35]. The concept of QA originated during that era with Alan Turing's "Imitation Game" or "Turing Test," which aimed to facilitate human-machine interaction through a natural language interface [36].

In the 1960s and 1970s, Early systems like BASEBALL [37] and LUNAR [38] allowed users to retrieve information from databases using natural language queries. However, they were limited by a lack of domain-specific information. As research progressed, linguistic analysis and statistical methods were incorporated to better understand question requirements [39], [40].

With the appearance of the web as a vast source of information, QA systems started using web data for answering questions in the 2000s [41], [42]. In 2016 Mishra *et al.* [43] expanded the knowledge base of QA systems using linguistic and rule-based approaches to retrieve answers and emphasized the utilization of the semantic web in QA tasks.

The attention mechanism, first proposed in 2015 by Bahdanau *et al.* [11], has had a significant impact on computerized natural language processing. This approach has been widely adopted across various NLP tasks, and its effectiveness was further improved in 2017 with Vaswani *et al.* [12] introduc-

tion of self-attention in transformer networks. Attention mechanism have made possible to introduce a new generation of NLP neural algorithms such as EPT [44], GPT [14], or T5 [15]. This breakthrough has revolutionized the NLP field, providing a powerful new framework for natural language understanding tasks.

## 1.2 Theoretical framework

### 1.2.1 Question answering framework

The configuration of QA methods can differ across different systems, tailored to the specific demands of each QA task. Nevertheless, a fundamental operative framework for a QA system can be organized into two main components: Knowledge Source and Question Answering Algorithms [1]

### 1.2.2 Knowledge source

A knowledge source in general is a repository that stores data related to a QA system in a structured (databases), unstructured (free texts) or semi-structured (such as XML-annotated texts) format [1], [45]. In terms of their knowledge source, modern QA systems can be divided into two categories: Open-domain question answering (ODQA) and closed-book question answering (CBQA). ODQA systems, also known as Information Retrieval models, have the ability to access external texts or sources in order to retrieve relevant context and generate answers based on the extracted information[46]. Indeed, Modern ODQA systems leverage the internet, specifically search engines like Google Search, to consider the entire web as an extensive information repository [47]. On the other hand, CBQA, also referred to as Knowledge Based Question answering, aims to answer open-domain questions directly without relying on external knowledge sources. Instead, CBQA utilizes the stored parametric knowledge within pretrained language models (LMs) [16], [18], [48].

Recently, there has been a rise in a novel concept known as Generative Question Answering [13], [49]. This approach involves enhancing large generative models like GPT and T5 models, which were initially utilized without incorporating extra knowledge [14], [15], with a given source of knowledge as input. This approach enables generative models to leverage their strong ability in reading comprehension tasks [15].

### 1.2.3 Question answering algorithms

QA systems heavily rely on QA algorithms, which are crucial for effectively processing the provided context (such as a passage, document, or parametric knowledge) and query, and subsequently generating the anticipated answers [1]. In general, a QA algorithm consists of two essential components: a knowledge retriever and an answering model. These components



work together to retrieve relevant information and formulate appropriate responses in QA systems [50].

### 1.2.3.1 Retriever

The main objective of a document retriever is to process a query and fetch the most relevant documents. These retrieved documents are then ranked according to their similarity score to the given query [1]. Several methods have been utilized in recent years to compute the cosine similarity between the query and a document corpus. These methods include TF-IDF [51], Okapi Best Matching (BM25) [52], and dense embedding vectors generated by neural networks (e.g., Dense Passage Retrieval and sentence embeddings) [50], [53], [54]. Furthermore, when dealing with a large set of knowledge documents, it has been beneficial to divide the documents into shorter passages with some overlap to improve response performance [8].

### 1.2.3.2 Answering model

The primary purpose of the Answering model is to respond to a given question by either extracting relevant information or generating a response in the form of free text [50]. This model can take both the question and a given knowledge source as input in order to provide an answer. The primary objective of these models is to encode the relationship between the question and the given context, handle long-range dependencies during the answer processing modules, and capture the concepts and hidden patterns of hierarchical features in a comprehensive way, in order to produce accurate responses [1].

#### 1.2.3.2.1 Language models and large language models

Language Models (LMs), which are statistical models that assign probabilities to word sequences, play a crucial role in various natural language processing tasks [17]. Recent advancements in neural-network based LMs have led to the development of large model architectures with millions or even billions of parameters [14]–[16]. These models are trained on extensive datasets, including nearly a terabyte of English text, and are commonly referred to as Large Language Models (LLMs) [17]. The increased scale of these models enhances their capacity to generate coherent and fluent natural language [18]–[20].

LLMs are usually assessed based on the quantity of parameters they possess, with larger quantities often considered superior. The parameter count serves as a reflection of the model’s size and complexity, enabling it to handle, learn from, and create more data. Nonetheless, a higher parameter count demands greater computational and memory resources, and introduces a higher likelihood of either overfitting or underfitting the data. During training, these parameters are fine-tuned or revised through an optimization process which seeks to reduce the discrepancies between the pre-

dicted and actual outputs. By modifying these parameters, the model's efficiency and precision in performing a specific task or in a particular domain can be enhanced [55].

#### 1.2.3.2.2 Transformer models

Transformer model uses a neural network architecture that relies on a self-attention mechanism [12]. This allows the model to discern the relevance of different text elements for a given task, letting it focus on specific aspects of the text. In essence, attention determines the relative importance of an input sequence for an output by applying a dynamically learned score to each element of the input sequence.[56]

#### 1.2.3.2.3 Extractive pre-trained transformer

Along with the architectural changes, pre-training has also played an essential role in performance improvement [56]. While pre-training is extremely resource-intensive, the resulting model, which obtains a general language representation, is often made publicly available to the NLP community. The same model can thus be fine-tuned on different downstream tasks, which is cheap in training time and data size, therefore affordable even in low-resource scenarios.

Bidirectional Encoder Representations from Transformers (BERT), is a model based on the original transformer architecture that is pre-trained using two strategies: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM involves random masking of some input tokens and then predicting these tokens based on their surrounding context. NSP, on the other hand, is a binary classification task where the model determines whether two sentences are consecutive or not. [44], [56]. This model differs from its predecessors by employing the Transformer's architecture [12], to interpret text both forwards and backwards, leading to a more nuanced understanding of context [8]. BERT has influenced the development of subsequent models, such as RoBERTa [57] and ALBERT [58], which have further exploited BERT's features. Pretrained transformer models have been effectively fine-tuned and deployed for QA tasks in the past few years.[8], [9].

Robustly optimized BERT approach (RoBERTa), is an EPT model strategy that uses dynamic masking, full sentences without NSP (Next Sentence Prediction) loss, larger mini-batches, and an expanded vocabulary for training. It builds upon the BERT model, modifying key hyperparameters, removing the next-sentence pretraining objective, and using much larger mini-batches and learning rates. This implementation is similar to Bert-Model, but with a minor tweak in embeddings, and a configuration suitable for RoBERTa pretrained models [57]. Among the various BERT models, RoBERTa is recognized as one of the most stable, delivering the highest F1 and EM scores for extractive QA tasks [56].

### 1.2.3.2.4 Text to text generation

Text-to-text generation is an encoder-decoder type of ML framework used widely in diverse natural language processing tasks. They are constructed on transformer structures that utilize attention mechanisms [12]. A prime example of these models is Google's T5, which revolutionizes all NLP tasks by morphing them into text-to-text tasks, hence offering a standardized solution for a variety of tasks [15]. T5 simplifies its usage across diverse tasks by redefining all NLP tasks in a text-to-text manner. T5 undergoes training on a vast collection of text, aiming to forecast the ensuing word in a sentence during its pre-training stage. This phase is succeeded by a fine-tuning stage tailored for particular tasks, which permits the model to assimilate from a broad spectrum of data before focusing on a distinct task, thus achieving superior outcomes across numerous benchmarks.

FLAN-T5, a development of the T5 model, is a LLM that's been specially refined through instruction tuning [59]. Its enhancements deliver robust zero-shot, few-shot, and 'Chain of Thoughts' capabilities. Leveraging the strengths inherent in generative models, FLAN-T5 has been effectively utilized in question answering tasks, demonstrating satisfactory performance [13], [34].

### 1.2.3.2.5 Generative pretrained transformers

The use of generative pretrained transformers for text generation has been widely recognized, particularly with models such as GPT [14], GPT-2 [60], and GPT-3 [16]. These models utilize a transformer framework and operate using a dual-phase approach: first, pretraining, then fine-tuning. During pre-training, they are taught on massive volumes of untagged text to foresee the subsequent word in a string, allowing them to comprehend contextual correlations. After this comes fine-tuning, where the models are educated on specified tagged data for further tasks [14]. GPT models have demonstrated exceptional performance in question-answering exercises [48], with GPT-3 outperforming its predecessors in terms of contextual enhancement [13], [21].

Currently, numerous recognized LLMs are in operation, with the biggest ones having more than a hundred times the parameters of GPT-2 (1.5B). Nonetheless, these models are typically the guarded intellectual assets of big corporations, and are restricted behind a commercial API, accessible only upon application, or completely unavailable for external use [61]. Lately, there has been significant progress in making OS GPT models such as PT-Neo (2.7B parameters) [62], GPT-J-6B [63], Pangu--13B [64], GPT-NeoX-20B [61], LLama (7-65B parameters) [65] and the most recent GPT4All (6,7,13B) [66]–[68] publicly and freely accessible.

### 1.2.3.2.6 Prompt engineering

Prompt engineering is growing in significance as a necessary compe-

tence for proficient interactions with LLMs [69]. Prompts serve as commands to LLMs, guiding them to follow certain rules, automate functions, and deliver specific quality and quantity of output. Moreover, prompts represent a unique kind of programming that facilitates the personalization of the LLM's responses and interactions. They can further optimize or fine-tune an LLM's performance in designated tasks [9].

#### 1.2.3.2.7 Parameter-efficient fine-tuning

As LLMs continue to expand in size, complete fine-tuning becomes too demanding to be executed on consumer-grade hardware. Moreover, the costs associated with storing and deploying separately fine-tuned models for each specific task become exceedingly high, given that these fine-tuned models are identical in size to the original pretrained model. The purpose of Parameter-Efficient Fine-tuning (PEFT) techniques is to mitigate these two issues. [70]. PEFT is a strategy designed to facilitate the execution of a new task by a model, through minimal adjustments. In this paradigm, a pre-trained model undergoes fine-tuning through updates to a limited number of added or chosen parameters. In simpler terms, PEFT strategies only adjust a small amount of additional model parameters while preserving most of the pretrained LLMs' parameters, significantly reducing both computational and storage demands. [71] PEFT techniques allow for performance levels comparable to full fine-tuning but require only a fraction of the parameters to be trainable [70].

Among the most commonly employed PEFT methods is the Low-Rank Adaptation (LoRA) technique. LoRA permits the training of some dense layers in a neural network indirectly by optimizing the rank decomposition matrices of the dense layers' alterations during adaptation, all while keeping the pretrained weights static. [72]

#### 1.2.3.2.8 Hugging Face OS hub

Hugging Face (HF) [73] functions as an extensive OS Deep Learning model repository, acting essentially as a collaborative platform where different teams can distribute their pre-trained models [74], [75]. It houses the broadest and most diverse range of pre-trained models and datasets, with the count surpassing 60,000 [76]. HF is highly recognized for its development of Transformers, an OS deep learning framework. This framework provides APIs and tools for downloading top-tier pre-trained models and further fine-tuning them to enhance performance.

### 1.2.4 Evaluation of QA systems

Numerous datasets have been devised to benchmark QA systems. These datasets are diverse in their structure, content, number of questions, and corresponding responses. The creation of a high-quality dataset specifically for generative question answering is a vital progression that could con-

siderably expedite the evolution of QA systems [1]. Popular QA datasets can be categorized according to the type of target responses and formulated queries. Questions can be sorted into different classifications primarily based on their complexity, response nature, or necessary methods for answering [43].

Furthermore, the richness of natural languages in terms of words and synonyms allows for an almost limitless variety of output sentences, each capable of encompassing diverse phrases with significantly different interpretations. Given this broad range of language expressions, a QA system's generated output may feature various forms of ambiguities, such as lexical, semantic, syntactic, and so forth [1].

In order to ascertain the quality of a QA system's output (generated/predicted answer), it needs to be assessed against a provided reference, also known as ground truth, gold answer, or labeled response [77], [78]. Yet, due to the complexity of natural language, the output is a structured entity rather than a singular value, complicating the evaluation process. Consequently, a multitude of evaluation scores have been proposed [77], [79]–[81]. These scores can generally be divided into two groups [1]: Human Centric Evaluation Scores (HCES) and Automatic Evaluation Score (AES).

### **1.2.4.1 Human centric evaluation score (HCES)**

HCES metrics depend on a group of individuals, such as experts or specialists, to evaluate the results of a QA system using specific guidelines and criteria, including the adequacy, fluency, and coherence of a text [1][1]. Despite being recognized as the premier evaluation method that delivers the most reliable scores, HCES metrics are susceptible to human errors arising from subjectivity and bias in judgement.

The implementation of HCES can take two forms: using a nominal scale where an evaluator assesses each pair of predicted and expected questions [9], or by ranking a set of question and answer pairs from highest to lowest quality [10]. To validate the consistency of the evaluation or annotations, HCES metrics are subjected to a reliability measure known as inter-annotator agreement (IAA) [82]. Generally, a high IAA is preferred as it measures the consensus among multiple evaluators regarding the quality ranking of an output.

### **1.2.4.2 Automatic evaluation score (AES)**

AES metrics utilize algorithms to automatically evaluate the performance or quality of machine-generated outputs or predictions, thus removing the requirement for human intervention. In the context of a Question-Answering task, this involves the quantification of discrepancies or similarities between the machine-generated response and the provided reference response. Additionally, AES approaches are a more cost-effective alternative compared to HCES, mainly because they don't need domain specialists. AES metrics

are particularly useful in large-scale tasks that have a dynamic element [83]. It's noteworthy that the scores derived from AES typically align well with those given by human evaluators [78]. AES metrics do not necessitate any prior training because its scores are independent of predetermined parameters. This parameter-free aspect makes it user-friendly, contributing to its widespread popularity. There are several subcategories within this system, determined by the level of operation among text units [1]:

- Based on characters.
- Based on words.
- Based on embeddings.

#### 1.2.4.2.1 Exact match (EM)

The EM score assigns a value of one if the predicted answer (R) matches exactly with the gold answer (A), and a value of zero if they are different. In other words, EM is a binary score that only takes values of either 0 or 1.

#### 1.2.4.2.2 F1-score

The F1-score is designed to establish the harmonic mean, equally prioritizing recall and precision. It gauges the word correspondence between the predicted answer and the actual (gold) answer in a more adaptable manner than the EM approach, as it also facilitates a balance between precision and recall. However, it doesn't account for the similarity in semantic content between two correct answers that are semantically different. The F1-score is computed as follows [1]:

$$F1 = 2 \cdot \left( \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \right)$$

**Where:**

$$\text{Recall} = \frac{\text{number of n-grams in alignment between A and R}}{\text{total count of n-grams in R}}$$

$$\text{Precision} = \frac{\text{number of matching n-grams between A and R}}{\text{total count of n-grams in A}}$$

#### 1.2.4.2.3 Bilingual evaluation understudy (BLEU)

BLEU [84], is a commonly employed evaluation metric for machine-generated translations that determines the quality of these translations by comparing them to one or more reference translations. It quantifies a similarity score by tallying the number of matching n-grams between the machine-

produced and reference translations, with translations having more coinciding n-grams getting higher scores. The process of obtaining the BLEU score involves computing the precision of n-grams in the machine translation relative to the reference translations, imposing a brevity penalty, and then calculating the geometric mean of these precision scores.

$$BP = \min\left(1, \exp\left(1 - \frac{r}{c}\right)\right)$$
$$BLEU = BP \cdot \exp\left(\sum_i w_i \cdot \log(p_i)\right)$$

**Where:** Length Penalty (BP) is the length penalty is a brevity penalty that adjusts the BLEU score to favor translations or summaries that are closer in length to the reference translations. Length of translation (r) and Length of output (c) are the total length of the reference translations and machine-generated outputs respectively. Precision (p) is calculated by dividing the number of matching n-grams in the machine-generated output by the total number of n-grams and weights (w) (usually 0.25) are assigned to each n-gram size

#### 1.2.4.2.4 Recall-oriented understudy for gisting evaluation (ROUGE)

The ROUGE metric was initially developed to evaluate the quality of summarizing systems. The metric juxtaposes machine-generated summaries with reference summaries to assess how effectively the generated summary encapsulates significant information from the references. It uses calculations involving n-gram overlap, word sequences, and various other features for this purpose. The frequently utilized metrics are ROUGE-N, measuring n-gram overlap, and ROUGE-L, identifying the longest common sequences of words, irrespective of their order [85].

#### 1.2.4.2.5 Metric for evaluation of translation with explicit ordering (METEOR)

METEOR, in contrast to BLEU, is grounded in the harmonic mean with a greater emphasis on recall over precision. It is explicitly structured to operate at the sentence level rather than the corpora level [1]. METEOR serves as a measurement tool for assessing machine-generated translations by comparing them with reference translations, factoring in both precision and recall. The METEOR score is derived by tallying matching unigrams, applying stemming, assessing word order similarity, and integrating precision, recall, and word order through a weighted harmonic mean. [86].

$$\begin{aligned}
P_r &= \frac{w}{|w|} \\
R_e &= \frac{w}{|w'|} \\
F_{mean} &= \frac{10 \cdot P_r \cdot R_e}{R_e + 9 \cdot P_r} \\
\text{Penalty} &= 0.5 \cdot \left( \frac{c}{w_n} \right)^3 \\
\text{METEOR} &= F_{mean} \cdot (1 - \text{Penalty})
\end{aligned}$$

**Where :** ( $w$ ) is the number of mapped unigrams in A that are also found in R, ( $|w|$ ) is the total number of unigrams in A and ( $|w'|$ ) is the total number of unigrams in R. ( $c$ ) is the number of chunks and ( $w_n$ ) is the number of mapped unigrams.

#### 1.2.4.2.6 Sentence Transformers

Sentence Transformers is a Python-based framework offering cutting-edge embeddings for sentences, text, and images. It includes SentenceBERT (SBERT), a modified version of the pre-trained BERT network, which leverages siamese and triplet network structures to generate semantically significant sentence embeddings that can be compared using cosine similarity. Sentence Transformer is intended to serve as an encoder for sentences and brief paragraphs. It takes an input text and produces a dense vector of a certain dimensionality that encapsulates semantic information. Sentence vectors' cosine similarity can be computed and employed for various tasks such as information retrieval, clustering and evaluation of QA systems[1], [87].

#### 1.2.4.3 Machine-trained evaluation scores (MTES)

While there have been numerous endeavors to create automated evaluation metrics for machine translation with the goal of lessening or eliminating the need for human assessment, the automatic evaluation of QA systems remains a complex task. This is attributed to the fact that traditional word-overlap metrics such as BLEU or ROUGE are deemed biased and don't align well with human evaluations. HCES can be costly, labor-intensive, and often necessitate domain expertise, rendering them unsuitable for tasks generating large data outputs or exhibiting dynamic behavior [1] In order to address these challenges, MTES combines HCES and AES with the aim of automatically assessing the score of a predicted response by comparing it to a gold-standard answer. It achieves this by implementing prediction models that utilize AES metrics as input parameters and human labels as output [1]. This approach highlights the existence of a certain level of correlation



between various AES metrics and human judgment in question answering scenarios [9].

## 2. Data

### 2.1 Source of knowledge data

The proposed knowledge source for the QA system consists of an unstructured text file primarily comprising the complete Kubernetes's documentation. In addition, real-time search queries are conducted on Google to address any potential questions that are not covered in the original Kubernetes's documentation file [47]. A systematic web scraping process is executed to extract the complete Kubernetes's documentation from its official documentation website[88]. This process involves collecting text data from 706 web links, which are then merged into a single text file, resulting in a plain text document with a total of 5,495,422 characters. Real-time Google searches are performed using the preprocessed incoming question as the search input. The complete textual information from the top three search results (if available) is extracted and appended to the foundational Kubernetes's documentation text file.

Following the text cleaning guidelines of Anandarajan *et al.* [89], the incoming question and the combined knowledge text file undergo the following preprocessing steps: conversion to lowercase, removal of HTML tags, elimination of NLTK English stopwords, removal of double spaces, and elimination of line breaks. These preprocessing steps ensure that the data is in a suitable format for utilization by the QA system.

### 2.2 Evaluation dataset

To comprehensively assess the different QA system methods examined in this project, a balanced dataset encompassing various types of questions is constructed [1], [43]. This dataset consists of 28 queries concerning Kubernetes technology, a collection sourced from typical job interview questions, frequently asked questions on Stack Overflow[90], and straight from Kubernetes documentation. Each item in the dataset comprises a question, its category, and a reference answer. The quality of questions and answers is verified by cloud engineer professionals. They also rephrase them to prevent exact matches with internet searches, aiming to create a more realistic QA task where the question may not be identical to those existing in the knowledge database. The dataset consists of four questions and respective answers in each of these seven categories: close-ended, open-ended, conceptual, situational, command-based, comparative, procedural, and multiple-answer. The complete proposed evaluation dataset is available in *Appendix A*.

## 3. Models

### 3.1 Deep Set: RoBERTa-base-Squad2

RoBERTa, among different BERT models, is considered to be one of the most consistent, providing top-tier results for extractive question-answering assignments. However, the RoBERTa model requires a diverse and extensive set of question and answer training examples for accurate implementation in QA tasks [56]. To overcome this limitation, the EPT model “deepset/roberta-base-squad2” from the HF hub [73] is chosen as the OS EPT answering model for this project. This model has a total size of 497mb and consists of 123M parameters. It is pre-trained on the SQuAD dataset [91], which comprises over 100,000 examples of questions formulated by crowdworkers on various Wikipedia articles [91]. Due to its extensive range of examples, the SQuAD dataset serves as a powerful benchmark for training and evaluating general-purpose extractive QA ML models.

### 3.2 Google: Flan-t5-xl

The “google/flan-t5-xl” model, which is a generative T5 fine-tuned by Google and available through the HF hub [73], is selected as the OS T5 response model for this project for two primary reasons. Firstly, it achieves an optimal balance between size, complexity, and the computational resources required. Secondly, it has proven its effectiveness in question-answering tasks by showcasing commendable performance [13], [34][11,12]. This Flan-T5 checkpoint model has a size of 11.4 GB and boasts a total training complexity encompassing 3 billion parameters.

### 3.3 Nomic AI: GPT4All-LoRa (7B)

The GPT model “nomic-ai/gpt4all-lora”, available in the HF hub [73], is chosen for this experiment as the OS GPT response model due to its ability to work fully locally and independently, without any API key connection. This feature ensures the handling of user data is free from privacy concerns, setting it apart from the majority of publicly and freely accessible powerful GPT models. GPT4All-LoRa model is a fine-tuned version of a LLaMA 7B instance, trained via the PEFT - LoRA method on a million carefully selected prompt-response pairs, with the GPT-3.5-Turbo OpenAI API utilized as the response model [66]. The entire GP4All model occupies 26.94 GB and embodies a complexity of 7 billion parameters.

### 3.4 Open AI: ChatGPT 3.5-turbo

ChatGPT 3.5-turbo, which is OpenAI's most capable model accessible to the public, has a training complexity of 175B parameters. It has been specifically optimized for chat applications using reinforcement learning techniques, where reward models are trained through human comparisons [92]. This proprietary benchmark model has been opted for comparison purposes in this study because it consistently delivers outstanding results [10], [13], [16], [21] and has experienced a recent surge in popularity [22].

### 3.5 Sentence transformer: all-MiniLM-L6-v2

The model "sentence-transformers/all-MiniLM-L6-v2" from the HF hub [73] is grounded in SBERT [87] and has been trained on a corpus of 1.17 billion sentence pairs. This model translates sentences and paragraphs into vectors in a 384-dimensional dense space, making it beneficial for various tasks such as semantic searches and clustering. In this project, the model is essential as it serves as the core component of the knowledge retriever for the QA systems. Additionally, it plays a significant role in the evaluation process for comparing answers. The selection of this model is based on its ability to perform semantic extraction tasks, prioritizing the semantic implications rather than solely relying on word frequency. This characteristic reduces the susceptibility to repetitive themes and concerns, particularly when overlapping parameters are introduced or unintentional duplication of content occurs within the knowledge source.

## 4. Methodology

This study consists of two main stages. The initial phase focuses on the approach and methodology employed to address the task of QA. The subsequent phase is dedicated to assessing the effectiveness of these methodologies with a proposed evaluation framework.

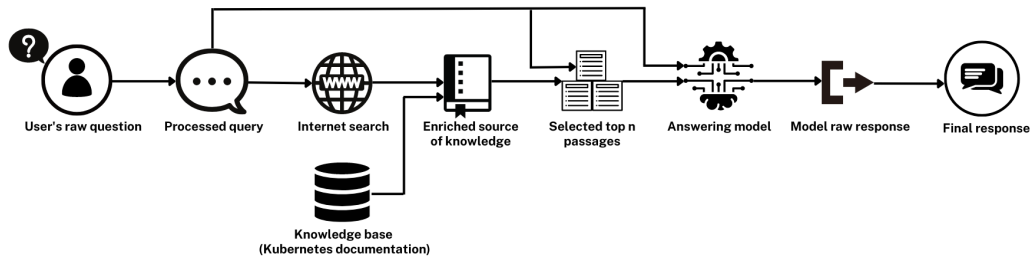
### 4.1 Suggested QA system

The methodology employed in this work follows the QA system framework proposed by Farea *et al.* [1]. The QA system encompasses three steps. The initial step is the extraction of relevant knowledge, responsible for providing a useful context that aids the answering model in responding to the presented question. This process can be adjusted by altering various parameters, which results in different context lengths and compositions. The following step involves producing a response using a specified QA model. Several models and hyperparameters can be employed at this stage. Post-processing of the model's response may also be required to ensure that the final answer is coherent and clear. The last step involves the construction of an interface that acts as a bridge between users and the QA system. This allows for an interactive exchange of information where the user can pose questions and receive answers. *Figure 4.1* provides a diagrammatic representation of this proposed QA system framework.

#### 4.1.1 Retrieval of knowledge

Considering the vast amount of knowledge available (surpassing 5M characters) and the limitations of existing response models in terms of context size, it is necessary to employ a retrieval operation to identify the specific passages that are most likely to contain relevant information for responding to the given question. This process begins by dividing the knowledge source into segments or chunks. Various variables can influence the final context to be extracted, including the selected length of each segment (measured in character count) and the indicated degree of overlap between segments. The latter is crucial in capturing any potential answers that might be split across segments.

After the enriched knowledge source has been partitioned, the next step involves identifying the most relevant segments for answering the question. In this stage, the Sentence Transformer model, specifically "all-MiniLM-L6-v2," calculate the sentence embeddings for each segment and the processed query. To enhance computational efficiency, the sentence embeddings for



**Figure 4.1:** The flow diagram of the proposed QA System framework. Initially, a user submits a question that is then refined to enhance its quality. Following this, the improved question is employed to scour the internet for pertinent information. This data is incorporated into the original system’s knowledge base to create a more comprehensive knowledge resource. This enriched information source is then divided into various segments, and the sentence embeddings of these segments, along with the question, are calculated. The most relevant segments (top-n most similar to the question) are chosen and integrated to serve as input context along with the refined user question for the response model. Finally, the response from the model is cleaned to ensure it’s quality and is presented as the system’s ultimate response.

the knowledge base are computed only once initially, while the embeddings for the internet passages are recalculated each time the query is modified. Subsequently, all sentence embeddings are combined, and the segments are ranked based on their cosine similarity with the query embeddings. The number of top "n" segments chosen also can be adjusted to generate different combinations of context. Once the top "n" segments have been obtained, they are merged into a single text block that serves as the context for the response model. This context block is prone to contain the answer to the user’s question, as it is semantically associated with the given question.

### 4.1.2 Answering model

In this stage, both the query and the extracted relevant context are inputted into the answering model. GPT and T5 models require an additional prompt that guides how to utilize the context and question for formulating a response. *Figures: 4.2, 4.3 and 4.4* illustrates the utilization of such prompts. The choice of model and specific hyperparameters are the variables in this scenario. RoBERTa and T5 use default settings, while for GPT models, the ‘temperature’ parameter is set to 0 based on the Ippolito *et al.* [93] study. This study suggests that a lower temperature value generates grammatically correct and deterministic text, indicating less randomness or variability in the responses. After receiving the unprocessed text output from the

```
Prompt: Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, don't try to make up an answer  
Context: {context}  
Question: {question}  
Answer:
```

**Figure 4.2:** The prompt employed to correlate the context and query in the input of the Flan - T5 model. This particular prompt has been found to be the most effective with this model. An extra assertion has been incorporated to avoid the system generating inaccurate (hallucinatory) answers.

model, a cleaning operation is performed to remove any double spaces, line breaks, and non-alphanumeric characters to ensure the quality of the response.

```
Prompt: Use the following pieces of context to answer the question at the end.  
### Context: {context}  
### Question: {question}  
### Response:
```

**Figure 4.3:** The prompt that is used to link the context and the question in the input of the GPT4All model. This prompt has been identified as the most effective for this model as it triggers the largest percentage of provided answers. The usage of the triple "#symbol, specified for this model, is also highlighted.

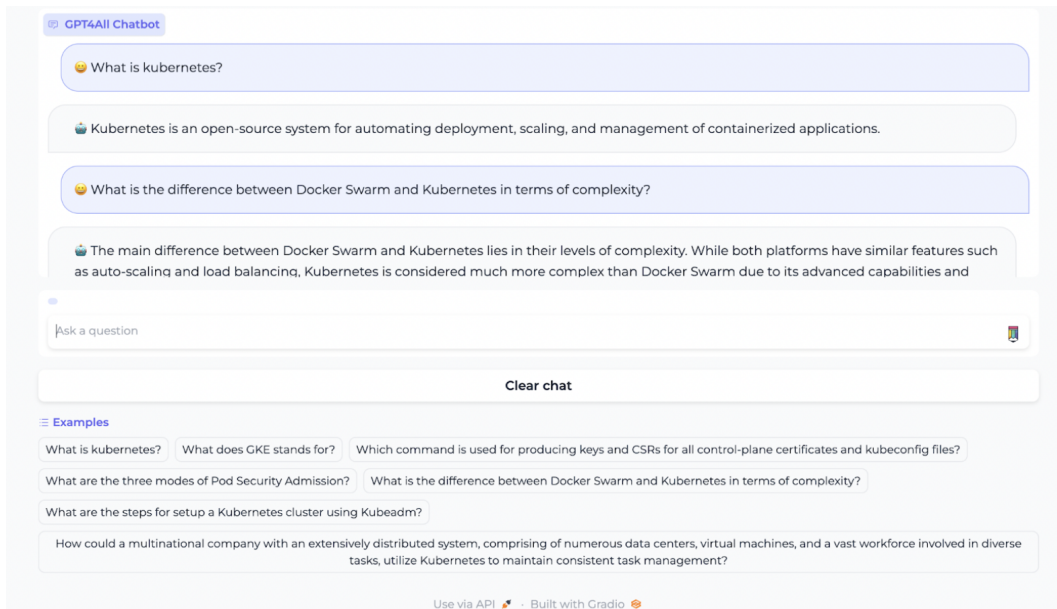
```
Prompt: You are a chatbot Your purpose is to assist users in how to use Kubernetes Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, don't try to make up an answer  
Context: {context}  
Question: {question}  
Expand with your own knowledge the answer and try to give examples if required.  
Answer:
```

**Figure 4.4:** The prompt utilized for coupling the context with the question in the input of ChatGPT 3.5-turbo model. By integrating the role of being a chatbot and broadening responses, it greatly increases the percentage of successful answers while mitigating hallucination effects.

### 4.1.3 Interface

In order to establish a communication bridge between users and the QA system, an interface is developed using the Gradio Python library for this

project. Gradio is an efficient tool for demonstrating ML models through an accessible web interface. *Figure 4.5* offers a glimpse of the constructed interface.



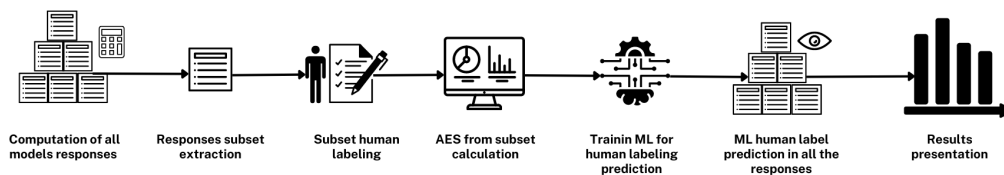
**Figure 4.5:** A snapshot of the built interface. It shows a textbox for user queries, an interactive chat-like display presenting the system’s responses, and examples of the types of questions that can be posed to the system.



## 4.2 System evaluation:

This research utilizes a thorough process, modeled after the work of Chen *et al.* [9], to assess the effectiveness of each approach. However, it goes a step further by incorporating a ML model designed to calculate the estimated human label (EHL) MTES, the process is illustrated in *Figure 4.6*.

Firstly, responses from each model are generated using various context input combinations. From these generated responses, a subset is selected and labeled by human experts based on specified guidelines. Following this, the AES metrics for this subset are computed, and a ML model is trained using these metrics as predictors and the label as the response variable. This is done with the intention of creating a model that can score the remaining responses in a way that mimics human scoring (EHL). Finally, a comprehensive evaluation of these outcomes is conducted.



**Figure 4.6:** Flow diagram of the evaluation process conducted.

### 4.2.1 Combinations' responses

To evaluate the performance variations of each model with different context sizes, certain retriever parameters are adjusted to generate different input combinations. The modified parameters include the size of the passage, which could be 1000, 2000, 4000, or 8000 characters, and the number of top- $n$  passages that are most similar to the query. For the latter, the top 1, 3, 5, or 10 passages could be selected. The generated input combinations max out at 12,000 characters, coming close to the fixed limit of 4096 input tokens for ChatGPT 3.5-turbo.

The overlap of characters between passages is fixed to 100 characters, this value has been determined as the optimum in previous research [8]. After setting these different retrieval combinations, an iterative procedure is conducted to answer the evaluation dataset previously suggested in *Appendix A* with each QA model. This procedure creates a set of answers for every combination of context and model, yielding a total of 1036 responses.

### 4.2.2 Human labeling

After generating the responses, they are consolidated into a unified dataframe. From this combined dataframe a portion consisting of 157 questions and answers, which equates to approximately 15% of the total generated from all combinations, is selected through stratified sampling in order to maintain the original proportion of question categories. Then the subset's generated responses are classified with the aid of two experts in the field of cloud technology. The classification scale ranges from 0 indicating a question that has been answered incorrectly, to 3 marking a perfectly answered question. Complete instructions for this labeling process can be found in *Appendix B*. To ensure consistency in the classification, an IAA calibration is carried out beforehand to ensure both annotators are using comparable standards. The distribution of responses assigned to each label is displayed in *table 4.1*.

Label	Number of answers
0	66
1	18
2	19
3	58

**Table 4.1:** The distribution of assigned labels along the subset of responses.

### 4.2.3 AES calculation

After the subset has been categorized, various AES metrics between the reference answers and the machine-generated responses are computed. These include BLEU, ROUGE1, ROUGE2, ROUGEL, cosine similarity of Sentence Embeddings, METEOR, F1, and EM. The outcome of this process is a dataset that includes the question, the reference and generated answers, the human-provided label, and the computed AES metrics.

### 4.2.4 EHL MTES

While the aggregated dataset might appear ready to form a ML model for label prediction, it's not immediately usable. This is predominantly due to the disparity in the number of label categories, as depicted in 4.1. To correct this, the SMOTE (Synthetic Minority Over-sampling Technique) approach is employed. This ML algorithm aids in counteracting imbalances within datasets by generating artificial samples from the lesser-represented class,

thereby achieving uniform class distribution. It does this by selecting a sample from the minority class, finding its closest neighbors, and interpolating between them to create synthetic samples [94].

After equalizing the dataset, a basic correlation analysis is undertaken to determine if the AES metrics are significant predictors of the labels. The Pearson correlation coefficient ( $r$ ) between different AES predictors and labels is shown in *table 4.2*. A random forest classification (RFC) model, with each tree limited to a maximum depth of 10, is then trained using the balanced data. For a preliminary assessment of the RDC, a cross-validation process is conducted with 10 folds, using accuracy as the evaluation metric. The mean accuracy score across all cross-validation folds is 0.778.

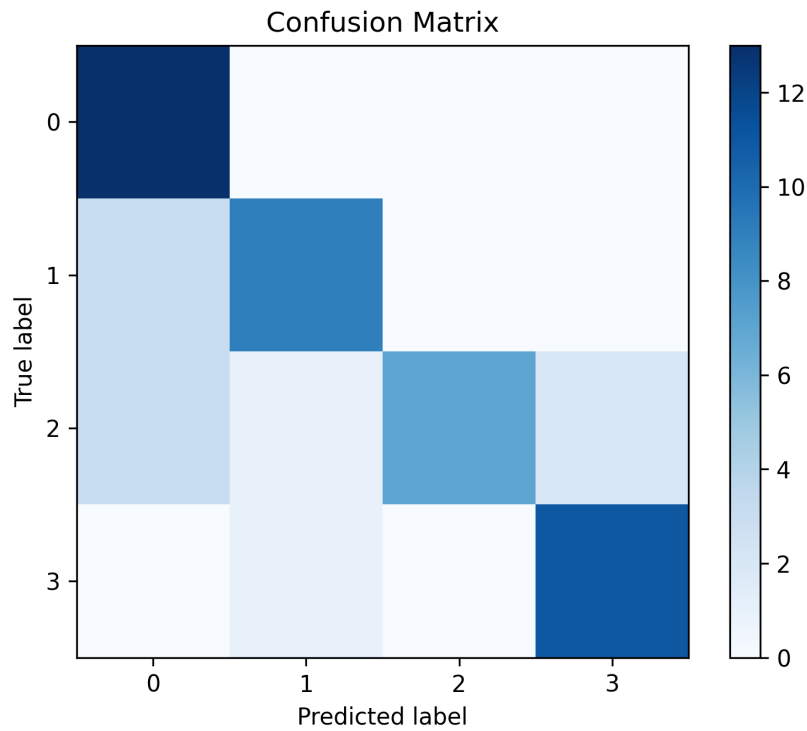
To provide a more comprehensive assessment of the RFC model's performance, a separate test using a divided training and testing dataset is implemented. The testing dataset accounts for 20% of the total data. The accuracy of the model in this test dataset is 0.8 and *figure 4.7* illustrates the corresponding confusion matrix. Finally, a manual review of some classified samples is carried out to check the consistency of the outcomes

<b>AES metric</b>	<b>r</b>
Sentence Embeddings cosine similarity	0.59
METEOR score	0.54
F1 score	0.44
ROUGE1 score	0.44
ROUGE2 score	0.41
ROUGEL score	0.38
BLEU score	0.28
exact match	0.23

**Table 4.2:** The ranked Pearson correlation coefficients between the candidate predictors and the human label scores.

#### 4.2.5 Prediction and assessment over all combinations

Once the classification model has been developed and validated to deliver reliable and acceptable predictive outputs, it is then used to make estima-

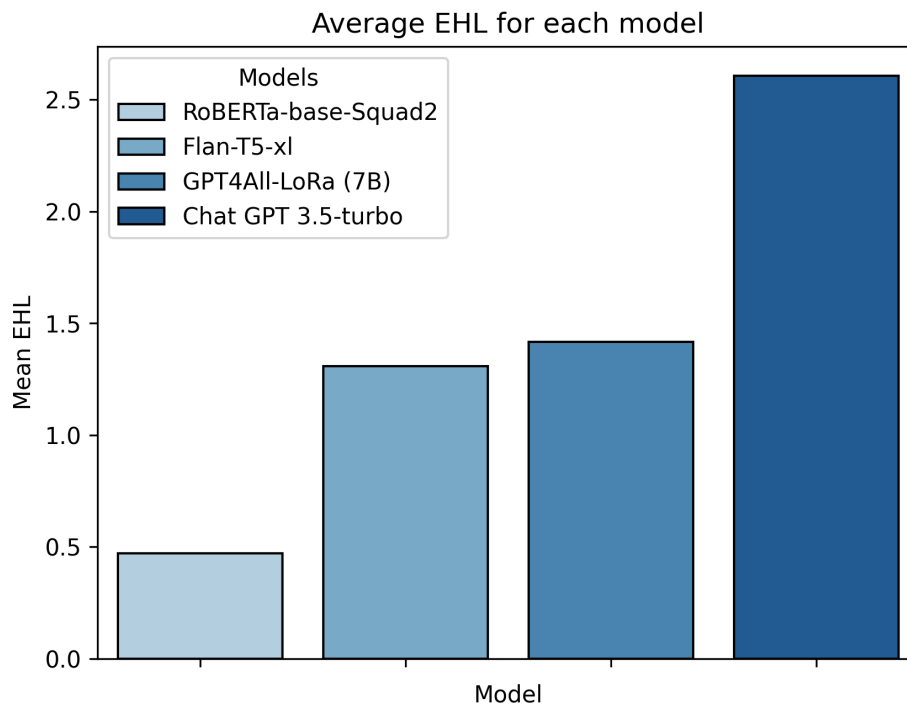


**Figure 4.7:** Confusion matrix of the RFC model on the test dataset.

tions on the labels of the remaining data in the original, full dataset. Upon generating these predictions, each system's output can be prepared for thorough analysis. This process involves evaluating the performance based on the estimated metrics derived from label estimations. Therefore, these estimations not only serve to give an outlook on unseen data but also provide a tangible measure of the systems' performance.

## 5. Results

With the introduction of the EHL MTES, a comprehensive evaluation can be conducted on various models and their potential combinations. *Figure 5.1* illustrates the average performance of each model in answering the evaluation dataset. The overall analysis reveals that the proprietary model, ChatGPT 3.5-turbo, outperforms all other models with an average EHL of 2.61 along all the combinations tested. Among the OS models, GPT4All excels slightly over Flan T5 with an EHL of 1.42 and 1.31 respectively, while RoBERTa demonstrates the lowest performance with an average EHL of less than 0.5.



**Figure 5.1:** The Bar graph showing the average performance of each model over the evaluation dataset, measured using the RFC model.

Moreover, *table 5.1* provides the results for all possible combinations of inputs and models. The EPT model, RoBERTa, achieves a maximum mean EHL score of 0.64 across the dataset. As for the text-to-text generation model, Flan T5, the average maximum EHL is 1.71. The GPT model, GPT4All, reaches a maximum EHL of 2.14 with one of its combinations. However, it should be noted that this model was unable to achieve a 100%

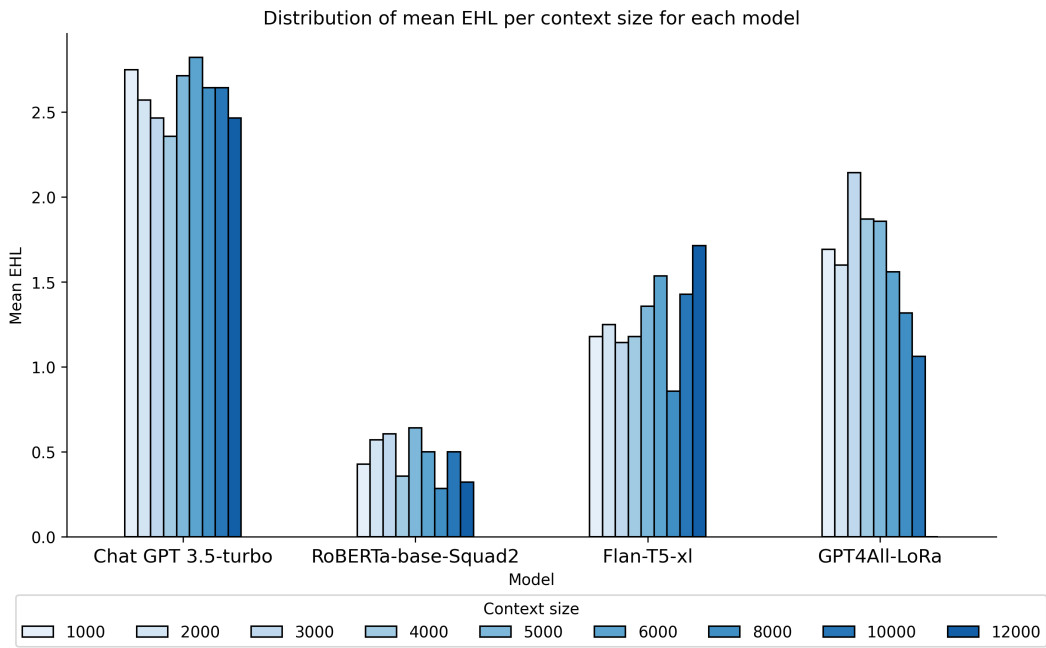
response rate. On the other hand, the proprietary model, ChatGPT3.5-turbo, demonstrates an impressive average EHL performance of 2.82 at its maximum. Although each model reaches its peak performance under different input configurations, the combination that exhibits the most reliable performance overall across all models is the use of a chunk size of 2000 characters with 3 concatenated context pages. This combination achieves an average EHL score of 1.6 for all models and provides a 97% response rate. *Appendix C*, shows the responses of the best combination for each model along the evaluation dataset.

Model-Input combination	RoBERTa-base-Squad2	Flan-T5-xl	GPT4All-LoRa (7B)	ChatGPT 3.5-turbo	Average
Chunk size: 1000 context pages: 1	Mean EHL: 0.43 Answered: 100%	Mean EHL: 1.18 Answered: 100%	Mean EHL: 1.69 Answered: 93%	Mean EHL: 2.75 Answered: 100%	Mean EHL: 1.51 Answered: 98%
Chunk size: 1000 context pages: 3	Mean EHL: 0.61 Answered: 100%	Mean EHL: 1.14 Answered: 100%	Mean EHL: 2.14 Answered: 75%	Mean EHL: 2.46 Answered: 100%	Mean EHL: 1.59 Answered: 94%
Chunk size: 1000 context pages: 5	Mean EHL: 0.43 Answered: 100%	Mean EHL: 1.36 Answered: 100%	Mean EHL: 1.86 Answered: 75%	Mean EHL: 2.71 Answered: 100%	Mean EHL: 1.59 Answered: 94%
Chunk size: 1000 context pages: 10	Mean EHL: 0.64 Answered: 100%	Mean EHL: 1.36 Answered: 100%	Mean EHL: 0.81 Answered: 57%	Mean EHL: 2.75 Answered: 100%	Mean EHL: 1.39 Answered: 89%
Chunk size: 2000 context pages: 1	Mean EHL: 0.57 Answered: 100%	Mean EHL: 1.25 Answered: 100%	Mean EHL: 1.6 Answered: 71%	Mean EHL: 2.57 Answered: 100%	Mean EHL: 1.5 Answered: 93%
Chunk size: 2000 context pages: 3	Mean EHL: 0.5 Answered: 100%	Mean EHL: 1.54 Answered: 100%	Mean EHL: 1.56 Answered: 89%	Mean EHL: 2.82 Answered: 100%	Mean EHL: 1.6 Answered: 97%
Chunk size: 2000 context pages: 5	Mean EHL: 0.43 Answered: 100%	Mean EHL: 1.5 Answered: 100%	Mean EHL: 1.31 Answered: 57%	Mean EHL: 2.53 Answered: 100%	Mean EHL: 1.44 Answered: 89%
Chunk size: 4000 context pages: 1	Mean EHL: 0.36 Answered: 100%	Mean EHL: 1.18 Answered: 100%	Mean EHL: 1.87 Answered: 82.1%	Mean EHL: 2.36 Answered: 100%	Mean EHL: 1.44 Answered: 96%
Chunk size: 4000 context pages: 3	Mean EHL: 0.32 Answered: 100%	Mean EHL: 1.71 Answered: 100%	Mean EHL: 0 Answered: 11%	Mean EHL: 2.46 Answered: 100%	Mean EHL: 1.12 Answered: 78%
Chunk size: 8000 context pages: 1	Mean EHL: 0.29 Answered: 100%	Mean EHL: 0.86 Answered: 100%	Mean EHL: 1.32 Answered: 57%	Mean EHL: 2.64 Answered: 100%	Mean EHL: 1.28 Answered: 89%

**Table 5.1:** The average EHL score and the percentage of answered responses for each potential model-input combination. The best result in terms of mean EHL is highlighted in green, while the worst is in red.

*Figure 5.2* illustrates the performance distribution when adjusting the input variables of chunk size and context pages, revealing a certain degree of variability. Both RoBERTa and ChatGPT models demonstrate their best performance within a moderate context size range of 5000 and 6000 characters respectively. Flan-T5 models exhibit a positive correlation between performance and context size, indicating bigger context sizes boost the model performance. In contrast, the GPT4All models show a distinct performance peak centered around a smaller context size of 3000 characters.

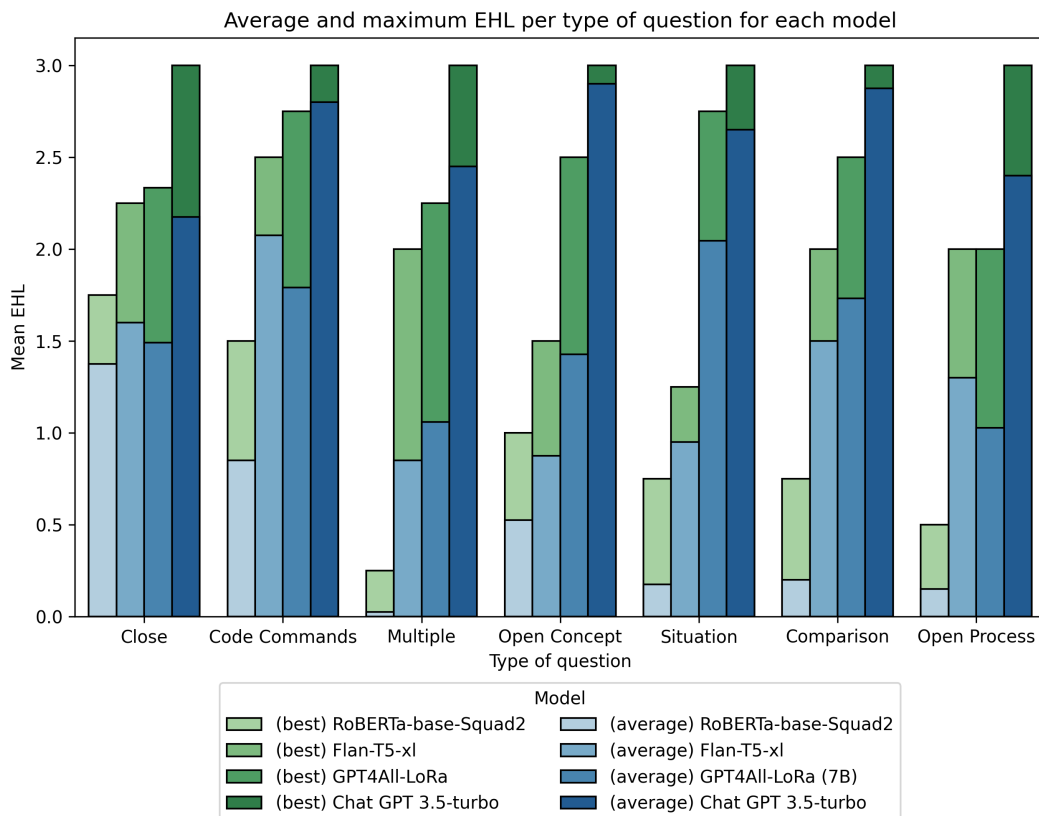
The effectiveness of each model in addressing various types of questions, as measured using the EHL metric, is presented in *figure 5.3*. In terms of overall performance, GPT3.5-turbo outperforms OS tools across all question types. However, among the OS models, GPT4-All performs better on average in comparison, situations with multiple answers, and open concept



**Figure 5.2:** The distribution of average performance of each model for different input context sizes.

questions, while Flan-T5 excels in close, command, and process questions. Overall, the performance of the Roberta model is inferior compared to the others, although it is comparable to OS models specifically in close questions.

The *figure 5.3* also highlights the best results achieved by each model for different question types. ChatGPT3.5-turbo demonstrates excellent performance and surpasses the other models with a EHL of 3 in each question class. However the gaps between ChatGPT3.5-turbo and the OS models are relatively smaller, particularly in code commands and situations, where GPT4-All exceeds 2.5 EHL. In fact, GPT4-All stands out as the most superior OS model when considering the best possible model for each category.



**Figure 5.3:** The average and best performance of each model for different types of questions.



## 6. Discussion

The outcomes of this study provide the point of departure for tackling the research questions proposed at the beginning of this work.

### 6.1 Best OS model in the experiment

*RQ1* is addressed based on the overall findings of the models presented in *figure 5.1*, as well as the results categorized by question type shown in *figure 5.3*. GPT method surpasses EPT and T5 methods in terms of overall performance when applied to this dataset using the self-devised EHL metric. Regarding performance sorted by question type, the GPT models continue to consistently exhibit superior results for most question categories, among all tested combinations that exhibited the best performance. Nevertheless, the most proficient T5 models show substantial competitiveness across a majority of question types, at times even rivaling the GPT's performance in areas such as closed, command, multiple answer, comparison, and open process questions. Furthermore, it should be acknowledged that the GPT model struggled to answer all the questions. In fact, the model with the highest average EHL was only capable of returning an answer for 75% of the questions, indicating it is the worst model in terms of response rate confidence.

### 6.2 Comparison vs ChatGPT 3.5-turbo

The response to *RQ2* can be found through the evaluation of the data from *figures 5.1 and 5.3*. It can be discerned that the ChatGPT 3.5-turbo surpasses all other analyzed OS models in terms of overall performance from a broad perspective. However, when delving deeper and focusing specifically on the combinations that have shown the best performance for each model, it becomes clear that the difference in performance is considerably narrower. Notably GPT4All model dramatically bridges this performance gap. This becomes especially evident in the context of certain types of queries, which include code-related commands, scenarios that explore open-ended concepts, and various situational instances.

### 6.3 Impact of context size on performance

In terms of *RQ3*, The analysis of *table 5.1* and *figure 5.2* suggests that the increase of the input context size does not consistently lead to enhanced per-

formance across all models. Both RoBERTa and ChatGPT models achieve their highest performance when the context size is within a moderate range, specifically around 5000 and 6000 characters respectively. On the other hand, GPT4All models exhibit their peak performance with a smaller context size of approximately 3000 characters. Flan-T5 models are the only ones that show a positive relationship between performance and context size, suggesting that larger context sizes enhance these models' performance. It is worth noting to mention that larger context sizes lead to decreased response rates for GPT4All models.

## 6.4 Limitations

While this research offers a well-grounded operational and evaluation methodology for the QA system, it's crucial to recognize various limitations that induce inherent bias in the experiment outcomes. As a result, these should be acknowledged in subsequent research and when interpreting this methodology.

### 6.4.1 Retrieval and knowledge source

While the use of dense vector representations (embeddings) is one of the most effective ways to extract context, it's important to acknowledge that inaccuracies can occur in some cases, for instance when the necessary information to answer a query isn't correctly identified. This can subsequently lead to errors in the context provided to QA models. Furthermore, the pre-processing phase prior to extraction plays a significant role and warrants exploration. Analyzing various parameters like tokenization of words before splitting, overlap, and punctuation removal can aid in optimizing performance. Lastly, it's essential to note that even though the knowledge source is supplemented by information from the internet, it may not be comprehensive enough to address a query, or worse, it may contain unverified data leading to incorrect or false responses. The techniques for retrieval and data cleaning thus require attention.

### 6.4.2 QA models

There are several constraints associated with the utilization of QA models. Firstly, some models offer the capability of integrating prompts, which are instructions on how to format response outputs for improved results. However, the effectiveness of these prompts is heavily reliant on the fine-tuning process, and different combinations can yield either superior or inferior performance. Consequently, comprehensive research into prompt engineering for each specific model is necessary to identify the most optimal combinations.

Secondly, concerning the augmentation of models' knowledge with ex-

ternal context, there is pre-existing training knowledge that is embedded in these models. In the event of inconsistent data from these two sources, it can create confusion within the model, which may result in the generation of incorrect answers.

Finally, another restriction encountered when working with OS models is the demand for computational power. Unlike proprietary LLMs which come with sophisticated APIs and servers that handle all the required infrastructure to process user queries, OS models are typically tested locally. Thus, the available computational power, particularly GPU RAM, could be a limiting factor when exploring models with billions of training parameters. This might present a challenge for studies with limited budget or access to competitive local or cloud-based infrastructure.

### 6.4.3 Evaluation

Potential sources of error exist within the evaluation process. The creation of an evaluation dataset introduces bias related to the type of questions asked, as well as the quantity and distribution of each class. This factor is present when comparing models that excel in argumentation cases, such as is the case with generative models compared to those that solely extract answers directly from the text. Nonetheless, the balance of types can be adapted according to the specific objective of the QA system's application. Another key factor is the accurate reference answer for a given question, which might not always be correct since it is determined by the knowledge of the human experts. Moreover, Multiple valid answers may exist for a specific question, particularly for queries that require diverse approaches, like situational or process-oriented inquiries. This challenge could be addressed by considering a set of multiple correct responses for some questions, but the inherent characteristics of language will invariably introduce some degree of bias.

Furthermore, human error during the labeling process should also be considered. Despite clear labeling guidelines and an IAA metric calibration conducted among annotators in this experiment, the inherent human bias in these activities remains inevitable. The human labeling error will also influence the EHL estimation with the ML model, particularly when considering that AES metrics do not align perfectly with EHL. This misalignment could result in inaccuracies in this metric's estimation, as evidenced in the confusion matrix in figure 9 and the model's cross-validation results.

## 7. Conclusion

The objective of this research was to evaluate the effectiveness of various transformer methodologies for question-answering (QA) operations in the field of cloud technology. The study aimed to analyze the performance of open-source (OS) models compared to the proprietary model ChatGPT 3.5-turbo while investigating the impact of input context size in output accuracy. The research questions were successfully addressed through a unique approach that involved the creation of a custom evaluation dataset and the development of a new Machine-trained evaluation score (MTES) called Estimated Human Label (EHL).

Based on the conducted analysis along different input-model combinations, despite being the model with the lowest rate of response, GPT4All model emerged as the most competitive among the proposed OS models in terms of performance with an average EHL of 1.42 in all the tested combinations. For the optimal input-model combinations, GPT4All demonstrated a performance comparable to the proprietary model ChatGPT 3.5-turbo with question types that exceeded an EHL of 2.5 such as code commands and situations. Additionally, it was observed that increasing the input context size does not consistently enhance performance across all models, just being beneficial for Flan-T5.

However, it is important to acknowledge the intrinsic limitations of this study, specifically the biases introduced by the retrieval process, the sources of knowledge, the implementation of QA models, and the evaluation process. These factors should be taken into account when interpreting the results, and caution should be exercised in considering them as absolute. The proposed evaluation methodology opens up possibilities for future exploration. Additional variables such as chunk overlap and model hyperparameters could be examined, along with different techniques for knowledge retrieval and prompt result comparison. Emerging robust models requiring high computational power, such as Falcon-40B, could also be considered for further research.

In the context of growing privacy concerns related to proprietary models, investigating the performance of OS models in NLP tasks is crucial. These OS models offer better data privacy safeguards and present a potential alternative. The introduction of new evaluation metrics like EHL shows promise in assessing QA system performance more effectively. The practical applications of this study are evident in its ability to enhance QA assistance platforms tailored to specific knowledge fields and provide valuable insights into the impact of context on performance, both of which are vital for designing and improving future systems.

This research makes a significant contribution to the advancement of QA systems driven by OS models. By elucidating the variables that influence their efficiency and combining human expertise with MTES, this investigation establishes a foundation for a more comprehensive understanding of QA systems. However, it also highlights the importance of ongoing research and refinement, indicating that further work in this field is both essential and holds great potential.

**Acknowledgements:** I would like to express my sincere gratitude to SUE B.V. for their generous support in facilitating this research project through their thesis internship programme. Additionally, I greatly appreciate their valuable assistance in the construction and further labeling of the evaluation dataset, leveraging their expertise in the cloud domain.

# Appendix A

N	Class	Question	Reference Answer
1	Close	Is the number of arguments specified by the configuration file for a Pod fixed?	No
2	Close	What does GKE stand for?	Google Kubernetes Engine
3	Close	Which mechanism within the control plane is utilized to impose particular configurations in the Security Context?	Security profiles
4	Close	How is called a non-terminating loop that regulates the state of a system?	Control loop
5	Code commands	Which specific command is capable of producing keys and CSRs for all control-plane certificates and kubeconfig files?	kubeadm certs generate-csr
6	Code commands	Which command is utilized to display PersistentVolumes in a sorted manner based on their capacity?	kubectl get pv --sort-by=.spec.capacity.storage
7	Code commands	Which command is utilized to retrieve the logs for a pod that were generated within the last hour?	kubectl logs --since=1h
8	Code commands	Why <code>kubectl get rc --namespace = " &lt; namespace_name &gt; "</code> command is used for?	To list the replication controllers by namespace.
9	Multiple	Provide a catalog of the different services that are offered within Kubernetes.	Cluster IP service, Load Balancer service, Node Port service, and External Name Creation service.

## Conclusion

N	Class	Question	Reference Answer
10	Multiple	What are the distinct Kubernetes objects?	Pods, Deployments, Services, ConfigMaps, Ingress, Secrets
11	Multiple	What are the three modes of Pod Security Admission?	Enforce, audit, warn
12	Multiple	Name the initial namespaces from which Kubernetes starts?	Default, kube-system, kube-public
13	Open concept	What is Kubernetes?	Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes provides a robust and scalable infrastructure for managing containerized workloads across a cluster of machines. It offers features such as automated container deployment, load balancing, self-healing, storage orchestration, and scaling. With Kubernetes, developers and system administrators can effectively manage and scale applications without manual intervention, ensuring high availability and efficient resource utilization.
14	Open concept	Define what is kubelet	The kubelet is a service agent that controls and maintains groups of pods by checking pod specification using Kubernetes. The kubelet runs on each node and allows communication between a master node and a slave node.
15	Open concept	What is the relationship between Kubernetes and Docker?	Docker builds the containers, and these containers communicate with each other via Kubernetes. So, containers running on multiple hosts can be manually linked and orchestrated using Kubernetes.
16	Open concept	What is a pod in Kubernetes?	Pods are high-level structures that wrap one or more containers.

N	Class	Question	Reference Answer
17	Situation	<p>In the scenario where a company built on monolithic architecture is experiencing issues due to the expansion of their operations, how could they make the transition to microservices and deploy their services in containers?</p>	<p>As the company's goal is to shift from their monolithic application to microservices, they can end up building piece by piece, in parallel and just switch configurations in the background. Then they can put each of these built-in microservices on the Kubernetes platform. So, they can start by migrating their services once or twice and monitor them to make sure everything is running stable. Once they feel everything is going good, then they can migrate the rest of the application into their Kubernetes cluster.</p>
18	Situation	<p>How could a multinational company with an extensively distributed system, comprising numerous data centers, virtual machines, and a vast workforce involved in diverse tasks, utilize Kubernetes to maintain consistent task management?</p>	<p>"As all of us know that I.T. departments launch thousands of containers, with tasks running across a numerous number of nodes across the world in a distributed system. In such a situation, the company can use something that offers them agility, scale-out capability, and DevOps practice to the cloud-based applications. So, the company can, therefore, use Kubernetes to customize their scheduling architecture and support multiple container formats. This makes it possible for the affinity between container tasks that gives greater efficiency with extensive support for various container networking solutions and container storage."</p>



## Conclusion

---

N	Class	Question	Reference Answer
19	Situation	Consider a situation where a company desires to update its deployment methods and create a platform that is highly scalable and responsive. How do you believe this company can accomplish this to meet the satisfaction of their customers?	In order to give millions of clients the digital experience they would expect, the company needs a platform that is scalable and responsive so that they could quickly get data to the client website. Now, to do this, the company should move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform available, i.e., Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.
20	Situation	As a consultant for a company that utilizes cloud infrastructure, what are some examples of recommended security measures you would suggest implementing for Kubernetes?	Examples of standard Kubernetes security measures include defining resource quotas, support for auditing, restriction of etcd access, regular security updates to the environment, network segmentation, definition of strict resource policies, continuous scanning for security vulnerabilities, and using images from authorized repositories.
21	Comparison	What distinguishes Docker Compose from Kubernetes?	Docker Compose deploys multi-container Docker apps to a single server, while Kubernetes is a production-grade container orchestrator that can run multiple container runtimes, including Docker's, across multiple virtual or physical machines.

N	Class	Question	Reference Answer
22	Comparison	What's the difference between Apache's Mesos and Google's Kubernetes?	Kubernetes and Apache Mesos are both open-source systems for managing containerized applications in a cluster of machines. Kubernetes is designed to be a general-purpose orchestration system, while Mesos is designed to be a more specialized cluster manager for running distributed systems.
23	Comparison	What is the difference between using Kubernetes Deployments vs StatefulSets?	"In Deployment, you specify a PersistentVolumeClaim that is shared by all pod replicas. In other words, a shared volume. The backing storage must have ReadWriteMany or ReadOnlyMany accessMode if you have more than one replica pod, while in StatefulSet, you specify a volumeClaimTemplates so that each replica pod gets a unique PersistentVolumeClaim associated with it. In other words, no shared volume. Here, the backing storage can have ReadWriteOnce accessMode."
24	Comparison	What is the difference between Docker Swarm and Kubernetes in terms of complexity?	Docker Swarm is a lightweight, easy-to-use orchestration tool with limited offerings compared to Kubernetes. In contrast, Kubernetes is complex but powerful and provides self-healing, auto-scaling capabilities out of the box.
25	Open process	What is the procedure for halting or pausing a pod in Kubernetes?	"Kubernetes doesn't support stop/pause of the current state of the pod and resume when needed. However, you can still achieve it by having no working deployments, which is setting the number of replicas to 0. kubect1 scale -replicas=0 deployment/<your-deployment>"

N	Class	Question	Reference Answer
26	Open process	I have established a Scheduled Job in Kubernetes that runs twice daily based on its predefined schedule. However, I would like to initiate it manually for testing purposes. How can I achieve this?	With: <code>kubect1 create job -from=cronjob/&lt;cronjob-name&gt; &lt;job-name&gt; -n &lt;namespace-name&gt;</code>
27	Open process	What are the steps for Setup a Kubernetes Cluster Using Kubeadm?	"1. Install container runtime on all nodes - We will be using cri-o. 2. Install Kubeadm, Kubelet, and kubect1 on all the nodes. 3. Initiate Kubeadm control plane configuration on the master node. 4. Save the node join command with the token. 5. Install the Calico network plugin. 6. Join the worker node to the master node (control plane) using the join command. 7. Validate all cluster components and nodes. 8. Install Kubernetes Metrics Server. 9. Deploy a sample app and validate the app."
28	Open process	How can Kubernetes be run on a local machine?	Kubernetes can be run locally using the Minikube tool. It runs a single-node cluster in a VM (virtual machine) on the computer. Therefore, it offers the ideal way for users who have just started learning Kubernetes.

**Table 7.1:** The evaluation dataset proposed.

# Appendix B

Label	Instructions
0	<b>The answer is not answered correctly:</b> Select this label if the provided answer is entirely incorrect or irrelevant to the question. The answer does not address the question or provide any useful information.
1	<b>The answer is partially correct:</b> Choose this label if the answer partially addresses the question and contains some relevant information. However, the answer is incomplete, inaccurate, or lacks necessary details.
2	<b>The answer is close to being perfectly answered:</b> Assign this label if the answer is mostly correct and provides relevant information. Although there may be minor errors, omissions, or slight inaccuracies, the answer demonstrates a good understanding of the question and provides valuable insights.
3	<b>The answer is completely and correctly answered:</b> Use this label if the answer is accurate, complete, and addresses all aspects of the question. The answer is comprehensive, well-structured, and provides precise information that directly answers the question without any errors or omissions.

**Table 7.2:** Labeling instructions for annotators.

Question: What distinguishes Docker Compose from Kubernetes?	
Real Answer: Docker Compose deploys multi-container Docker apps to a single server, while Kubernetes is a production-grade container orchestrator that can run multiple container runtimes, including Docker's, across multiple virtual or physical machines.	
Generated Answer	Given Label
They complement each other.	0
Docker Compose defines multi-container Docker applications and deploys them to a single server.	1
While Docker Compose is mainly focused on building and running multiple containers locally, Kubernetes is designed specifically for managing containerized workloads across different hosts.	2
Docker Compose is used to create and manage multi-container Docker applications on a single host, while Kubernetes is a production-grade container orchestrator that can run other container runtimes, including Docker's, over several machines, virtual or real. Docker Compose defines multi-container Docker applications and deploys them to a single server, while Kubernetes organizes containers according to an abstraction called a pod. Kubernetes is better suited for production environments and on a larger scale, while Docker Compose is better suited for local development and testing scenarios.	3

**Table 7.3:** Example of the label assigned to different types of responses.

# Appendix C

N	Model	Answer generated
1	RoBERTa-base-Squad2	two
	Flan-t5-xl	no
	GPT4All-LoRa (7B)	no, the number of arguments specified by the configuration file for a pod is not fixed. it depends on how many arguments are needed to execute the desired task within the container.
	ChatGPT 3.5-turbo	no, the number of arguments specified by the configuration file for a pod is not fixed. the user can define any number of arguments based on their requirements. in the example provided, the configuration file for the pod defines a command and two arguments, but the user can define more or fewer arguments as needed. additionally, the user can define arguments using environment variables, which provides more flexibility in defining arguments. for example, the user can define an argument for a pod using configmaps or secrets.
2	RoBERTa-base-Squad2	google kubernetes engine
	Flan-t5-xl	google kubernetes engine
	GPT4All-LoRa (7B)	gke stands for "google kubernetes engine".
	ChatGPT 3.5-turbo	gke stands for google kubernetes engine, which is a managed kubernetes service provided by google. it allows users to deploy and operate containerized applications at scale using google's infrastructure. gke is based on kubernetes, an open source container management and orchestration platform developed by google. gke provides the operational power of kubernetes while managing many of the underlying components, such as the control plane and nodes, for the users. gke is ideal for those who need a platform that lets them configure the infrastructure that runs their containerized apps, such as networking, scaling, hardware, and security. gke is used in a variety of industries, including robotics, healthcare, retail, education, gaming, and financial services.
3	RoBERTa-base-Squad2	profiles
	Flan-t5-xl	roles and rolebindings
	GPT4All-LoRa (7B)	the mechanism used to impose particular configurations in the security context is access control lists (acl).

N	Model	Answer generated
	ChatGPT 3.5-turbo	the mechanism within the control plane that is utilized to impose particular configurations in the security context is the admission control process. the admission control process has two phases: the mutating phase and the validating phase. mutating admission controllers, such as istio's automatic sidecar injection mechanism, can modify the objects they admit, while validating admission controllers do not. by using admission controllers, kubernetes can enforce security policies and configurations in the security context, ensuring that pods and containers are running with the appropriate security parameters. for example, pod security policies (psps) are cluster-level resources that define the security requirements that pods must meet before they can be created. psps can be used to enforce security policies such as preventing privileged containers, enforcing read-only file systems, and restricting host network access.
4	RoBERTa-base-Squad2	a control loop
	Flan-t5-xl	a control loop
	GPT4All-LoRa (7B)	
	ChatGPT 3.5-turbo	a non-terminating loop that regulates the state of a system is called a control loop. it is a fundamental concept in robotics and automation, where it is used to maintain a desired state of a system by continuously monitoring and adjusting its parameters. examples of control loops include thermostats, which regulate the temperature of a room, and cruise control systems in cars, which maintain a constant speed. in kubernetes, controllers are also control loops that watch the state of the cluster and make or request changes to bring it closer to the desired state.
5	RoBERTa-base-Squad2	kubeadm certs generate-csrthis
	Flan-t5-xl	kubeadm certs generate-csr
	GPT4All-LoRa (7B)	generate- keys-and-csrs
	ChatGPT 3.5-turbo	the specific command capable of producing keys and csrs for all control-plane certificates and kubeconfig files is "kubeadm certs generate-csr". this command is designed for use in kubeadm external ca mode, where it generates csrs that can be submitted to an external certificate authority for signing. the pem encoded signed certificates should then be saved alongside the key files, using ".crt" as the file extension, or in the case of kubeconfig files, the pem encoded signed certificate should be base64 encoded and added to the kubeconfig file in the "users > user > client-certificate-data" field. an example of using this command is "kubeadm certs generate-csr --kubeconfig-dir /tmp/etc-k8s --cert-dir /tmp/etc-k8s/pki".

## Conclusion

---

N	Model	Answer generated
6	RoBERTa-base-Squad2	kubectl
	Flan-t5-xl	kubectl get pv --sort-by='.spec.capacity.storage'
	GPT4All-LoRa (7B)	kubectl get pvc --sort-by=capacity
	ChatGPT 3.5-turbo	the command utilized to display persistentvolumes in a sorted manner based on their capacity is "kubectl get pv --sort-by=.spec.capacity.storage". this command will display the persistentvolumes in ascending order based on their storage capacity. for example, if a user wants to see the list of persistentvolumes sorted by their capacity, they can use this command to get a clear view of the available volumes and their respective capacities.
7	RoBERTa-base-Squad2	kubectl logs
	Flan-t5-xl	kubectl logs --since=1h
	GPT4All-LoRa (7B)	the correct response is "kubectl logs --since=1h".
	ChatGPT 3.5-turbo	the command utilized to retrieve the logs for a pod that were generated within the last hour is "kubectl logs --since=1h <pod-name>". this command will fetch the logs produced by the specified container within the last hour. for example, if you want to retrieve the logs produced by the nginx container within the last hour, you would run the command "kubectl logs --since=1h nginx". you can also specify the duration in hours, minutes, or seconds using the format "--since=1h30m20s" to fetch logs from the last 1 hour, 30 minutes, and 20 seconds.
8	RoBERTa-base-Squad2	no namespace was provided in the command
	Flan-t5-xl	to maintain backwards compatibility
	GPT4All-LoRa (7B)	1. kubectl getrc --namespace = " < namespace_name > " command is used to list all replication controllers (rc) present under specified namespace. it lists out rcs along with their status, creation time, labels etc. 2. by specifying the namespace name while executing the above mentioned command, we can restrict our search results only within that particular namespace. so, if we have multiple namespaces defined in our cluster, then we can filter down the search result based upon the desired namespace.

N	Model	Answer generated
	ChatGPT 3.5-turbo	the kubectl <code>get rc --namespace = " &lt; namespace_name &gt; "</code> command is used to get the list of all replication controllers (rc) in a specific namespace. replication controllers are responsible for ensuring that a specified number of pod replicas are running at any given time. by specifying the namespace, the command only returns the rc objects that belong to that particular namespace. for example, if you have multiple namespaces in your Kubernetes cluster, you can use this command to view the rc objects in a specific namespace. this can be useful for troubleshooting or monitoring purposes. additionally, you can use this command in conjunction with other kubectl commands to perform actions on specific rc objects in a particular namespace.
9	RoBERTa-base-Squad2	kubernetes service catalog
	Flan-t5-xl	the service catalog is a declarative Kubernetes API extension to discover and use cloud managed services
	GPT4All-LoRa (7B)	1) clustering services - these services help in managing clusters such as monitoring, logging, networking, security, and scheduling.2) storage services - these services provide persistent data storage solutions like file systems, block devices, and volume plugins.3) compute services - these services enable users to deploy containers on top of virtual machines or bare metal servers.4) networking services - these services allow users to connect their workloads together through networks using tools like load balancers, firewalls, VPN gateways, and NAT rules.5) monito..
10	RoBERTa-base-Squad2	estring, uid, and optionalnamespacespec
	Flan-t5-xl	pods



## Conclusion

N	Model	Answer generated
	GPT4All-LoRa (7B)	<p>There are many types of Kubernetes objects, but some of the most common ones include: 1. Deployment - A Deployment represents one or more replicas of a particular application. It defines the desired number of replicas and their desired state. 2. Service - A Service provides a way to expose a set of ports from a pod. Services allow clients outside of the cluster to access services inside the cluster. 3. Replication Controller - A Replication Controller manages a group of pods that share a single purpose. They define the desired number of replicas and monitor whether there are enough healthy replicas running. 4. Persistent Volume Claim - This type of object allows users to claim storage space in a shared file system. Users specify the size of the volume needed and where it should be mounted. 5. ClusterRoleBinding - Allows administrators to bind roles to groups of users. These binding rules determine who has permission to perform certain actions in the cluster.</p>
	ChatGPT 3.5-turbo	<p>There are many different types of Kubernetes objects, but some of the most commonly used ones include: 1. Pods: These are the smallest unit of deployment in Kubernetes and can host one or more containers. 2. Deployments: These are used to manage the lifecycle of one or more identical pods and allow you to declaratively manage the desired state of your application. 3. Services: These provide a way to expose applications running in pods and represent a set of pods that perform the same function. 4. ConfigMaps: These are used to store configuration data that can be accessed by pods. 5. Secrets: These are used to store sensitive information such as passwords or API keys that can be accessed by pods. 6. StatefulSets: These are used to manage stateful applications and provide guarantees about the ordering and uniqueness of pods. 7. Jobs: These are used to run batch or one-time tasks in a cluster. 8. CronJobs: These are used to run scheduled tasks in a cluster. 9. Ingresses: These are used to expose HTTP and HTTPS routes from outside the cluster to services within the cluster. 10. PersistentVolumes: These are used to provide persistent storage for pods. 11. Namespaces: These are used to create virtual clusters within a physical cluster and provide a way to organize and isolate resources.</p>
11	RoBERTa-base-Squad2	privileged, baseline, and restricted
	Flan-t5-xl	<p>enforce — any pods that violate the policy will be rejected- audit — pods with violations will be allowed and an audit annotation will be added- warn — pods that violate the policy will be allowed and a warning message will be</p>
	GPT4All-LoRa (7B)	There are three modes of pod security admission: warn, audit, and enforce.

N	Model	Answer generated
	ChatGPT 3.5-turbo	The three modes of pod security admission are warn, audit, and enforce. In warn mode, violations of the pod security standards are logged as warnings but not prevented. This mode is useful for testing and evaluating new security policies without disrupting existing workloads. In audit mode, violations are logged and prevented, but not enforced. This mode is useful for gradually implementing new security policies and identifying potential issues before fully enforcing them. In enforce mode, violations are both logged and prevented, and the pod security standards are fully enforced. This mode is the most restrictive and is recommended for production environments where security is a top priority. Examples of violations that could be prevented in enforce mode include running containers as root, using host network or PID namespaces, and mounting sensitive host directories into containers.
12	RoBERTa-base-Squad2	Kubernetes namespaces
	Flan-t5-xl	default, kube-system, and kube-public
	GPT4All-LoRa (7B)	The initial namespaces from which Kubernetes starts include "default" and "kube-system".
	ChatGPT 3.5-turbo	Kubernetes starts with three initial namespaces: default, kube-system, and kube-public. The default namespace is where services and apps are created by default. The kube-system namespace is used for Kubernetes components, and the kube-public namespace is used for public resources. These namespaces are created automatically when Kubernetes is installed and cannot be deleted. It is recommended to create additional namespaces to organize and manage resources effectively. For example, a team can create a namespace for their specific project or application, and all resources related to that project can be created within that namespace. This helps to avoid conflicts and makes it easier to manage resources.
13	RoBERTa-base-Squad2	an open-source container orchestration engine
	Flan-t5-xl	an open-source system for automating deployment, scaling, and management of containerized applications
	GPT4All-LoRa (7B)	

## Conclusion

---

N	Model	Answer generated
	ChatGPT 3.5-turbo	Kubernetes is an open-source container orchestration engine that automates the deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community. It is a powerful tool for managing containerized applications in a cloud-native environment. With Kubernetes, you can easily deploy, scale, and manage containerized applications across multiple hosts and clusters. Some examples of companies that use Kubernetes include Babylon, Booz Allen Hamilton, Booking.com, and AppDirect.
14	RoBERTa-base-Squad2	1.27.x-00
	Flan-t5-xl	node
	GPT4All-LoRa (7B)	kubelet is a daemon running inside each node of a Kubernetes cluster. It manages pods and containers running within those nodes according to the policies defined in the Kubernetes master.
	ChatGPT 3.5-turbo	kubelet is an agent that runs on each node in a Kubernetes cluster. Its main responsibility is to ensure that containers are running on the node as expected. It communicates with the Kubernetes API server to receive instructions on which containers to run and how to run them. kubelet also monitors the health of the containers and restarts them if necessary. Additionally, it manages the node's resources, such as CPU and memory, to ensure that containers have the necessary resources to run. Examples of kubelet configuration parameters include <code>--kubeconfig</code> , which specifies the path to a kubeconfig file for connecting to the API server, and <code>--eviction-hard</code> , which sets eviction thresholds for pods based on resource usage.
15	RoBERTa-base-Squad2	a group of one or more application containers
	Flan-t5-xl	Kubernetes is a container runtime and Docker is a container runtime.
	GPT4All-LoRa (7B)	Kubernetes and Docker are closely related technologies. Both were developed around the same time and share many similarities. However, they serve distinct purposes within modern cloud computing environments. While Docker was designed primarily as a tool for building and managing software containers, Kubernetes was built specifically to manage clusters of containers across multiple hosts. In other words, while Docker focuses on individual containers, Kubernetes focuses on groups of containers known as pods. This means that if you want to deploy a single containerized app into production, then Docker may suffice; however, if you need to scale up your deployment to handle hundreds or thousands of instances, then Kubernetes becomes essential.

N	Model	Answer generated
	ChatGPT 3.5-turbo	Kubernetes and Docker are two different technologies that can be used together to create and manage containerized applications. Docker is a containerization platform that allows developers to package their applications and dependencies into a single container that can be easily deployed and run on any system. Kubernetes, on the other hand, is a container orchestration platform that automates the deployment, scaling, and management of containerized applications. Kubernetes can work with any containerization platform, including Docker, to manage and orchestrate containers. For example, a developer can use Docker to create a containerized application and then use Kubernetes to deploy and manage that application across a cluster of servers. Kubernetes can also manage containers created by other containerization platforms, such as rkt or cri-o. Overall, the relationship between Kubernetes and Docker is complementary, with Kubernetes providing a powerful platform for managing containerized applications and Docker providing a flexible and easy-to-use containerization platform.
16	RoBERTa-base-Squad2	containers
	Flan-t5-xl	a container
	GPT4All-LoRa (7B)	a pod is the smallest deployable unit of computing capacity in Kubernetes. Each pod contains one or more containers, which together make up the application logic of the system.
	ChatGPT 3.5-turbo	In Kubernetes, a pod is the smallest and simplest unit of deployment. It is a logical host for one or more containers, and it provides a shared network namespace and storage volumes for those containers. A pod can contain multiple containers that share the same network namespace and can communicate with each other using localhost. Pods are designed to be ephemeral, meaning they can be created, destroyed, and replaced easily. They are also designed to be scalable, meaning multiple identical pods can be created to handle increased traffic or workload. For example, a pod can contain a web server container and a database container, and they can communicate with each other within the same pod. The pod can be managed by a deployment, which ensures that the desired number of replicas of the pod are running at all times.
17	"RoBERTa-base-Squad2"	a unified unit that is self-contained and independent from other applications
	Flan-t5-xl	using compass

## Conclusion

N	Model	Answer generated
	GPT4All-LoRa (7B)	The company should start by identifying which aspects of its current infrastructure need improvement. They may consider using tools such as Kubernetes or Mesosphere to help them manage their containerized applications. Additionally, they might look into adopting a CI/CD pipeline to automate the process of building, testing, and deploying their microservices. Finally, they will likely need to invest in training employees who work with these new technologies so that everyone understands what's involved in this shift toward microservices.
	ChatGPT 3.5-turbo	To make the transition from monolithic architecture to microservices, the company would need to break down their application into smaller, independent services that can be developed, deployed, and scaled separately. This would require a comprehensive understanding of the application's architecture and dependencies, as well as the ability to identify and isolate individual services. Once the services have been identified, they can be deployed in containers using containerization platforms like Docker or Kubernetes. Containerization allows for the creation of lightweight, portable, and scalable environments that can be easily replicated and managed. This enables the company to deploy multiple instances of the same service, all running in the same environment, and to scale up or down as needed. To ensure the success of the transition, the company would need to have a clear understanding of the benefits and challenges of microservices architecture, as well as the technical expertise to implement and manage the new architecture. They may also need to invest in new tools and technologies to support the transition, such as container orchestration platforms, monitoring and logging tools, and automated deployment pipelines. An example of a company that successfully transitioned from monolithic architecture to microservices is Netflix. In 2012, Netflix began migrating its application to a microservices architecture, which allowed them to scale their platform to support millions of users and to release new features and updates more quickly. They used containerization and container orchestration platforms like Docker and Mesos to manage their services and developed a suite of tools and technologies to support their new architecture. Today, Netflix is one of the largest users of microservices architecture, with over 700 services running in production.
18	"RoBERTa-base-Squad2"	Kubernetes enables businesses to manage IT workloads efficiently
	Flan-t5-xl	utilizing multiple cloud providers and distributed across multiple regions to provide a truly global multi-cluster environment
	GPT4All-LoRa (7B)	

N	Model	Answer generated
	ChatGPT 3.5-turbo	<p>A multinational company with an extensively distributed system can utilize Kubernetes to maintain consistent task management by setting up and managing multi-cluster deployments. Kubernetes can help in managing all the infrastructure and applications from a centralized location, providing better visibility with standardized processes and proper DevOps practices. This can significantly reduce the overall management complexity and help in managing operational expenditure. For example, a company with data centers in different regions can set up isolated Kubernetes clusters in each region to cater to specific geographic regions. This can limit the scope of regulatory requirements and enable organizations to target specific clusters to meet different requirements and gain regulatory compliance relatively easily. Additionally, a multi-cluster environment can help in isolating production issues to a specific cluster, allowing developers to investigate and fix issues without affecting other clusters. This can also ensure availability as other clusters step in to prevent downtime in case of a complete cluster failure. Moreover, a multi-cluster Kubernetes environment can be the ideal platform for distributed computing, especially with increasing edge computing requirements and the popularity of IoT services. Developers can create containerized applications that can be deployed in Kubernetes while providing a powerful orchestration engine distributed across multiple regions. This eliminates the need to cater applications to different environments and infrastructure. Overall, Kubernetes can help a multinational company with an extensively distributed system to maintain consistent task management by providing a centralized platform for managing infrastructure and applications, isolating production issues, and enabling distributed computing.</p>
19	"RoBERTa-base-Squad2"	in order to give millions of clients
	Flan-t5-xl	<p>to move from their private data centers (if they are using any) to any cloud environment such as AWS. Not only this, but they should also implement the microservice architecture so that they can start using Docker containers. Once they have the base framework ready, then they can start using the best orchestration platform available, i.e., Kubernetes. This would enable the teams to be autonomous in building applications and delivering them very quickly.</p>

## Conclusion

---

N	Model	Answer generated
	GPT4All-LoRa (7B)	<p>In order for this company to upgrade their deployment methods, they need to first identify what type of platform they want to use. They can choose between public clouds like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, etc., or they can opt for an on-premises solution if they prefer not to rely on third parties. The choice depends on factors like cost, security concerns, compliance requirements, and availability of resources. After choosing the appropriate infrastructure, the company must ensure that all necessary components are installed correctly and configured properly. To make sure everything works smoothly, they might consider automating some tasks through scripting languages like Python, Ruby, Bash, or PowerShell. Additionally, they can adopt containerization technologies like Docker or Kubernetes to deploy and manage multiple services efficiently. These tools allow developers to package software into lightweight packages called containers, making it easier to run different versions of the same application side by side without affecting each other's performance. Finally, the company can optimize their network connectivity to improve response times and minimize latency issues. By doing these things, the company can create a highly scalable and responsive platform that meets customer demands while reducing operational overhead.</p>
	ChatGPT 3.5-turbo	<p>To achieve a highly scalable and responsive platform, the company can adopt a microservices architecture and containerization using Kubernetes. By breaking down their monolithic codebase into smaller, independent services, they can scale each service independently as needed. This approach also allows for easier maintenance and updates, as changes can be made to individual services without affecting the entire system. Additionally, the company can move their infrastructure to a cloud environment, such as AWS or Google Cloud, to take advantage of the scalability and flexibility offered by cloud computing. This would allow them to easily add or remove resources as needed to handle fluctuations in traffic. Furthermore, the company can implement DevOps practices, such as continuous integration and continuous deployment, to streamline their development and deployment processes. This would allow them to quickly and efficiently deploy updates and new features to their platform, improving their responsiveness to customer needs. Overall, a combination of microservices architecture, containerization with Kubernetes, cloud infrastructure, and DevOps practices can help the company achieve a highly scalable and responsive platform that meets the satisfaction of their customers. Examples of companies that have successfully implemented these practices include Netflix, Amazon, and Google.</p>
20	"RoBERTa-base-Squad2"	securing the entire Kubernetes infrastructure
	Flan-t5-xl	continuous security vulnerability scanning

N	Model	Answer generated
	GPT4All-LoRa (7B)	<p>here are some recommendations for implementing security measures for kubernetes: - improve visibility into container activity through monitoring tools such as prometheus or datadog. these tools allow administrators to detect anomalies and potential attacks early on. - ensure that all components within the kubernetes stack are up-to-date with patches and fixes. - configure access control lists (acls), firewall rules, and ips policies based on the specific needs of the organization. - enable logging and auditing capabilities so that any suspicious activities can be easily traced back to their source. - utilize encryption technologies such as tls/ssl certificates and vpn tunnels to protect data transfers across the network. - regularly review and update security configurations to ensure they remain effective against evolving attack vectors.</p>
	ChatGPT 3.5-turbo	<p>Here are some recommendations for implementing security measures for Kubernetes: - Improve visibility into container activity through monitoring tools such as Prometheus or Datadog. These tools allow administrators to detect anomalies and potential attacks early on. - Ensure that all components within the Kubernetes stack are up-to-date with patches and fixes. - Configure access control lists (ACLs), firewall rules, and IPS policies based on the specific needs of the organization. - Enable logging and auditing capabilities so that any suspicious activities can be easily traced back to their source. - Utilize encryption technologies such as TLS/SSL certificates and VPN tunnels to protect data transfers across the network. - Regularly review and update security configurations to ensure they remain effective against evolving attack vectors.</p>
21	"RoBERTa-base-Squad2"	lacks the scalability and high availability features
	Flan-t5-xl	Docker Compose is limited to single-host deployments and lacks the advanced orchestration capabilities of Kubernetes.
	GPT4All-LoRa (7B)	



## Conclusion

---

N	Model	Answer generated
	ChatGPT 3.5-turbo	<p>Docker Compose and Kubernetes are both container orchestration frameworks, but they have some key differences. Docker Compose is designed to run containers on a single host machine, while Kubernetes can run containers over a number of computers, virtual or real. Docker Compose is best suited for small-scale applications that don't require complex orchestration, while Kubernetes is ideal for large-scale applications that require high availability, scalability, and fault tolerance. Another key difference is that Docker Compose defines multi-container Docker applications and deploys them to a single server, while Kubernetes is a production-grade container orchestrator that can run other container runtimes, including Docker's, over several machines, virtual or real. Kubernetes provides advanced features such as automatic scaling, rolling updates, and self-healing, which are essential for managing complex applications in production environments. For example, if you have a small web application that consists of a web server and a database, you can use Docker Compose to define the services and configurations required to run your application and launch it with a single command. On the other hand, if you have a large e-commerce platform that consists of multiple microservices, you can use Kubernetes to manage the deployment, scaling, and monitoring of your application across multiple nodes and clusters.</p>
22	"RoBERTa-base-Squad2"	container orchestration tools
	Flan-t5-xl	Mesos is designed to be a more specialized cluster manager for running distributed systems.
	GPT4All-LoRa (7B)	<p>1. apache mesos is a framework for building highly available and scalable distributed systems. it allows you to build and operate large-scale data centers or clouds. in contrast, kubernetes is focused on providing a simpler approach to managing containerized applications across multiple machines. 2. one key differentiator between these two technologies is how they handle application lifecycles. while mesos uses a centralized scheduler to coordinate resources, kubernetes relies on pods to define the relationship between containers and the host machine. this means that kubernetes offers greater flexibility when it comes to defining the behavior of individual containers within a larger application. 3. another important distinction is the level of abstraction provided by each technology. mesos focuses on abstracting away details like network topology and hardware configuration, allowing administrators to concentrate on higher-level concerns like service discovery and load balancing. by comparison, kubernetes emphasizes ease of use and simplicity, making it easier for non-technical teams to get started with containerization</p>

N	Model	Answer generated
	ChatGPT 3.5-turbo	<p>Mesos and Kubernetes are both container orchestration platforms, but they have some differences in their design and target use cases. Mesos is designed to be a general-purpose cluster manager that can handle a variety of workloads, including traditional applications, big data processing, and containerized applications. It provides a unified framework for managing resources across different types of workloads and supports multiple schedulers, allowing users to choose the one that best fits their needs. Kubernetes, on the other hand, is primarily focused on containerized applications and provides advanced features specifically tailored for managing containerized workloads at scale. It is optimized for running cloud-native applications and provides built-in support for containerization, service discovery, load balancing, and scaling. Kubernetes also has a strong ecosystem and community support, making it a popular choice for managing containerized applications in production environments. If the company's primary focus is on containerized applications and they want a platform that is specifically designed for managing containers, Kubernetes would be a better choice. However, if they have a diverse set of workloads and want a more general-purpose cluster manager that can handle different types of applications, Mesos could be a suitable option. Ultimately, the choice between Mesos and Kubernetes depends on the specific requirements and use cases of the company.</p>
23	RoBERTa-base-Squad2	they maintain sticky identities for each pod
	Flan-t5-xl	deployments are useful for managing stateless applications, while statefulsets are useful for managing stateful applications that require stable network identities and persistent storage
	GPT4All-LoRa (7B)	
	ChatGPT 3.5-turbo	<p>the main difference between using kubernetes deployments and statefulsets is that deployments are typically used for stateless applications, while statefulsets are more suited for stateful applications. deployments allow you to manage sets of identical pods using common configurations, while statefulsets treat each pod individually and require pods with unique identities. deployments are lightweight and provide granular control over the rollout of new pod versions and rollback to previous versions. they are ideal for applications that do not require persistent storage and can easily be replaced if a pod fails. on the other hand, statefulsets are used when state has to be persisted, and require persistent storage to enable the application to save its state and data across restarts. for example, if you are deploying a web application that does not require persistent storage, you can use a deployment. however, if you are deploying a database or a messaging system that requires persistent storage and unique identities for each pod, you should use a statefulset.</p>

## Conclusion

N	Model	Answer generated
24	RoBERTa-base-Squad2	an open-source platform for container orchestration
	Flan-t5-xl	kubernetes is well suited for complex applications. on the other hand, docker swarm is designed for ease of use, making it a preferable choice for simple applications.
	GPT4All-LoRa (7B)	1. docker swarm is simpler to install and configure compared to kubernetes.
	ChatGPT 3.5-turbo	the major difference between docker swarm and kubernetes is based on complexity. docker swarm is designed for ease of use, making it a preferable choice for simple applications. on the other hand, kubernetes is more powerful, customizable, and flexible, which comes at the cost of a steeper initial learning curve. kubernetes is well suited for complex applications that require automatic scaling, intelligent scheduling, and high availability of services through replication. however, the installation process is complex, especially for beginners, and it takes specialized knowledge to master kubernetes. in contrast, docker swarm is easier to install and configure, and instances are typically consistent across operating systems. docker swarm excels within smaller apps with fewer containers. overall, the right tool depends on your organization's needs and the complexity of your applications.
25	RoBERTa-base-Squad2	restarted while waiting for processes to terminate
	Flan-t5-xl	3/5when theprestopcompletes, the kubelet sends the sigterm signal to the container. from that point, the container should close all long-lived connections and prepare to terminate.
	GPT4All-LoRa (7B)	
	ChatGPT 3.5-turbo	there is no specific procedure for halting or pausing a pod in kubernetes. however, you can delete a pod using the kubectl delete command, as shown in the context above. this will immediately terminate the pod, but it will be relaunched by the cluster since it is part of a replicaset. if you want to temporarily stop a pod, you can scale the number of replicas to 0 using the kubectl scale command. this will effectively pause the pod until you scale the replicas back up again. for example, if you have a deployment named my-deployment, you can pause it by running:kubectl scale deployment my-deployment --replicas=0this will scale the deployment down to 0 replicas, effectively pausing all pods associated with it. to resume the deployment, you can scale it back up to the desired number of replicas:kubectl scale deployment my-deployment --replicas=3this will scale the deployment back up to 3 replicas, effectively resuming the pods associated with it.
26	RoBERTa-base-Squad2	9

N	Model	Answer generated
	Flan-t5-xl	kubectl create job --from=cronjob/cronjob-name> job-name> -n namespace-name>
	GPT4All-LoRa (7B)	to trigger a kubernetes cronjob manually, follow these steps: 1. install the latest version of 'kubectl'. you can download it from <a href="https://github.com/googlecloudplatform/kubernetes/releases/">https://github.com/googlecloudplatform/kubernetes/releases/</a> . 2. run 'kubectl version' to ensure that you have the correct version of 'kubectl'. 3. create a new cron job using the 'kubectl create job' command followed by specifying the name of the existing cron job ('--from') and then creating a new cron job with the same frequency and time zone as the existing cron job. 4. specify the namespace where the cron jobs should exist when creating them. 5. finally, start the newly created cron job using the 'kubectl apply' command.
	ChatGPT 3.5-turbo	to trigger a kubernetes scheduled job manually for testing purposes, you can create a simple job based on your scheduledjob. if you already run a scheduledjob, there are jobs in history. you can export one of these jobs using the command " <i>kubectl get job &lt; job - name &gt; -oyaml &gt; myjob.yaml</i> ". then edit the yaml a little bit, erasing some unnecessary fields and run it manually using the command " <i>kubectl create -f myjob.yaml</i> ". alternatively, you can use the command " <i>kubectl create job --from = cronjob/ &lt; cronjob - name &gt; &lt; job - name &gt; -n &lt; namespace - name &gt;</i> " to create a new job from an existing cronjob. this command is available with kubectl v1.10.1+ and is backwardly compatible with older clusters. it is important to note that kubernetes skips scheduled starts if the last run hasn't finished. to replace incomplete runs when the next job is scheduled, you can use the " <i>--replace</i> " flag. additionally, you can apply the concurrency policy to the cluster to create cronjobs that only permit a single run at any time. if you encounter any errors while triggering the scheduled job manually, make sure your kubectl version is up to date (21+) and check the starting deadline, successful jobs history limit, and failed jobs history limit values. you can also monitor cronjobs with mechanisms like the kubectl command and retrieve container logs for individual jobs.
27	RoBERTa-base-Squad2 apt-get upgrade oryum update	runapt-get update
	Flan-t5-xl	preparing the hostsinstall a container runtime and kubeadm on all the hosts.

## Conclusion

N	Model	Answer generated
	GPT4All-LoRa (7B)	to setup a kubernetes cluster using kubeadm, follow these steps: 1. install the necessary packages such as docker, coreos, etc., if they aren't already present. 2. create an admin user account and configure the systemd service files. 3. configure the kubelet and kube-proxy daemonsets. 4. run 'kubeadm init' to generate the initial configmap and secret files needed during installation. 5. edit the generated configmap and secret files to customize them according to your needs. 6. run 'kubeadm join -token xxxxxx' to join the new master node to the existing cluster. 7. verify that the newly joined master node is properly configured and functional.
	ChatGPT 3.5-turbo	to set up a kubernetes cluster using kubeadm, follow these steps:1. install kubeadm on all nodes that will be part of the cluster.2. initialize the cluster using the kubeadm init command. this will download and install the control plane components and create the necessary configuration files.3. deploy a pod network to the cluster using the kubectl apply command. this will enable communication between the nodes in the cluster.4. join any number of machines to the cluster using the kubeadm join command. this will add the nodes to the cluster and enable them to run workloads.5. customize the control plane components and other configuration options as needed using the kubectl edit command.6. make sure to keep a record of the kubeadm join command that was output during initialization, as this will be needed to add additional nodes to the cluster.for example, to initialize a cluster with kubeadm, you could run the following command:""sudo kubeadm init --pod-network-cidr=192.168.0.0/16""this would initialize the cluster with a pod network cidr of 192.168.0.0/16, which would enable communication between the nodes in the cluster. once the initialization is complete, you would need to deploy a pod network to the cluster using the kubectl apply command, such as:""kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml""this would deploy the calico network plugin to the cluster, which would enable communication between the nodes. finally, you could join additional nodes to the cluster using the kubeadm join command that was output during initialization, such as: <i>""sudokubeadmjoin192.168.1.100 : 6443 --tokenabcdef.1234567890abcdef --discovery --token --ca --cert --hashsha256 : 1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef""</i> this would add a new node to the cluster and enable it to run workloads.
28	RoBERTa-base-Squad2	kindkind
	Flan-t5-xl	minikube
	GPT4All-LoRa (7B)	

N	Model	Answer generated
	ChatGPT 3.5-turbo	<p>kubernetes can be run on a local machine using tools like kind, minikube, and kubeadm. kind is a tool that allows you to run kubernetes on your local computer. it requires docker to be installed and configured. the kind quick start page provides instructions on how to get started with kind. minikube is another tool that lets you run kubernetes locally. it runs an all-in-one or a multi-node local kubernetes cluster on your personal computer (including windows, macos, and linux pcs) so that you can try out kubernetes or use it for daily development work. the official get started! guide provides instructions on how to install and use minikube. kubeadm is a tool that can be used to create and manage kubernetes clusters. it performs the actions necessary to get a minimum viable, secure cluster up and running in a user-friendly way. once installed, you can use it to create a cluster. overall, running kubernetes on a local machine can be useful for testing and development purposes before deploying to a production environment</p>

**Table 7.4:** Responses of the best model-input combination along the evaluation dataset.

# Bibliography

- [1] A. Farea, Z. Yang, K. Duong, N. Perera, and F. Emmert-Streib, "Evaluation of question answering systems: Complexity of judging a natural language," *arXiv.org*, Sep. 2022. [Online]. Available: <https://arxiv.org/abs/2209.12617>.
- [2] M. B. Hoy, "Alexa, siri, cortana, and more: An introduction to voice assistants," *Medical Reference Services Quarterly*, vol. 37, no. 1, pp. 81–88, 2018. DOI: 10.1080/02763869.2018.1404391.
- [3] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd. 2020, Visited on 2021-06-04. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>.
- [4] M. Saeidi, M. Bartolo, P. Lewis, *et al.*, "Interpretation of natural language rules in conversational machine reading," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. DOI: 10.18653/v1/d18-1233.
- [5] M. Mensio and G. Rizzo, "The rise of emotion-aware conversational agents: Threats in digital emotions," in *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Barcelona, Spain, 2018, pp. 1541–1544. DOI: 10.1145/3184558.3191607.
- [6] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," in *Proceedings of the International Semantic Web Conference*, Auckland, New Zealand, 2019, pp. 69–78. DOI: 10.1007/978-3-030-30796-7\_5.
- [7] A. Bagga, T. Strzalkowski, and G. B. Wise, "Partsid: A dialogue-based system for identifying parts for medical systems," in *Proceedings of the Sixth Applied Natural Language Processing Conference*, Association for Computational Linguistics, Seattle, WA, USA, 2000, pp. 29–36.
- [8] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage bert: A globally normalized bert model for open-domain question answering," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 5878–5882.
- [9] A. Chen, G. Stanovsky, S. Singh, and M. Gardner, "Evaluating question answering evaluation," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 119–124.

- [10] L. Ouyang, J. Wu, X. Jiang, *et al.*, “Training language models to follow instructions with human feedback,” *arXiv.org*, Mar. 2022. [Online]. Available: <https://arxiv.org/abs/2203.02155>.
- [11] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [12] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, Curran Associates Inc., 2017, pp. 5998–6008.
- [13] G. Izacard and E. Grave, “Leveraging passage retrieval with generative models for open domain question answering,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Association for Computational Linguistics, Online, 2021, pp. 874–880.
- [14] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.
- [15] C. Raffel, N. Shazeer, A. Roberts, *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research (JMLR)*, 2020.
- [16] T. B. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [17] N. Carlini, F. Tramèr, E. Wallace, *et al.*, “Extracting training data from large language models,” in *Proceedings of the 30th USENIX Security Symposium*, Aug. 2021, pp. 978-1-939133-24-3. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting>.
- [18] Y. Zhang, S. Sun, M. Galley, *et al.*, “Dialogpt: Large-scale generative pretraining for conversational response generation,” in *Proceedings of the ACL Demo Track*, 2020.
- [19] A. Radford, J. Wu, D. Amodei, *et al.*, “Better language models and their implications,” *OpenAI Blog*, 2019.
- [20] R. Zellers, A. Holtzman, H. Rashkin, *et al.*, “Defending against neural fake news,” in *NeurIPS*, 2019.
- [21] D. Yin, L. Dong, H. Cheng, *et al.*, “A survey of knowledge-intensive nlp with pre-trained language models,” *arXiv preprint arXiv:2202.07971*, Feb. 2022.
- [22] K. Sunder, P. Khuwaja, and K. Dev, “Chatgpt needs spade (sustainability, privacy, digital divide, and ethics) evaluation: A review,” 2023.
- [23] H. Li, D. Guo, W. Fan, *et al.*, “Multi-step jailbreaking privacy attacks on chatgpt,” *arXiv.org*, Apr. 2023. [Online]. Available: <https://arxiv.org/abs/2304.05197>.
- [24] OpenAI, *Privacy policy*, <https://openai.com/policies/privacy-policy>, [Accessed: April 5, 2023].



- [25] I. Lyall, *Chatgpt bug raises privacy concerns*, <https://www.proactiveinvestors.co.uk/companies/news/1009978/chatgptbug-raises-privacy-concerns-1009978.html>, [Accessed: April 5, 2023].
- [26] S. Khowaja, I. Lee, K. Dev, M. Jarwar, and N. Qureshi, "Get your foes fooled: Proximal gradient split learning for defense against model inversion attacks on iomt data," *IEEE Transactions on Network Science and Engineering*, 2022.
- [27] S. Khowaja, P. Khuwaja, K. Dev, and A. Antonopoulos, "Spin: Simulated poisoning and inversion network for federated learning-based 6g vehicular networks," *arXiv preprint arXiv:2211.11321*, 2022.
- [28] S. Khowaja, K. Dev, N. Qureshi, P. Khuwaja, and L. Foschini, "Toward industrial private ai: A two-tier framework for data and model security," *IEEE Wireless Communications*, vol. 29, no. 2, pp. 76–83, 2022.
- [29] M. Zaib, W. E. Zhang, Q. Z. Sheng, A. Mahmood, and Y. Zhang, "Conversational question answering: A survey," *eprint arXiv:2106.00874*, Jun. 2021. DOI: 10.48550/arXiv.2106.00874.
- [30] Y. Chali, S. A. Hasan, and S. R. Joty, "Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels," *Information Processing & Management*, vol. 47, no. 6, pp. 843–855, 2011. DOI: 10.1016/j.ipm.2010.10.002.
- [31] S. K. Dwivedi and V. Singh, "Research and reviews in question answering system," *Procedia Technology*, vol. 10, pp. 417–424, 2013. DOI: 10.1016/j.protcy.2013.12.378.
- [32] A. Ansari, M. Maknojia, and A. Shaikh, "Intelligent question answering system based on artificial neural network," in *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, IEEE, 2016, pp. 758–763. DOI: 10.1109/icetech.2016.7569350.
- [33] S. P. Lende and M. Raghuvanshi, "Question answering system on education acts using nlp techniques," in *World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, IEEE, 2016, pp. 1–6. DOI: 10.1109/startup.2016.7583963.
- [34] D. Su, M. Patwary, S. Prabhumoye, *et al.*, "Context generation improves open domain question answering," in *Findings of the Association for Computational Linguistics: EACL 2023*, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 793–808.
- [35] P. Nadkarni, L. Ohno-Machado, and W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association: JAMIA*, vol. 18, no. 5, pp. 544–551, Sep. 2011. DOI: 10.1136/amiajnl-2011-000464.
- [36] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Masque/sql: an efficient and portable natural language query interface for relational databases," in *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh, 1993, INSERT PAGE NUMBERS HERE.

- [37] B. F. Green, A. K. Wolf, C. L. Chomsky, and K. Laughery, "Baseball: An automatic question-answerer," in *Proceedings of IRE-AIEE-ACM '61 (Western)*, 1961.
- [38] W. A. Woods, R. M. Kaplan, and B. Nash-Webber, "The lunar sciences natural language information system: Final report," BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, MA, Tech. Rep., Jun. 1972.
- [39] R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg, "Question answering from frequently asked question files: Experiences with the faq finder system," *AIMag*, vol. 18, no. 2, p. 57, Jun. 1997.
- [40] E. Riloff and M. Thelen, "A rule-based question answering system for reading comprehension tests," in *Proceedings of the ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, 2000.
- [41] R. Soricut and E. Brill, "Automatic question answering using the web: Beyond the factoid," *Information Retrieval*, vol. 9, pp. 191–206, 2006. DOI: 10.1007/s10791-006-7149-y.
- [42] B. F. Katz, S. Yuret, D. Ibrahim, and B. Temelkuran, "Omni-base: Uniform access to heterogeneous data for question answering," in *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB)*, 2002.
- [43] A. Mishra and S. K. Jain, "A survey on question answering systems with classification," *Journal of King Saud University - Computer and Information Sciences*, vol. 28, no. 3, pp. 345–361, 2016, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2014.10.007.
- [44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [45] A. Burchardt, N. Reiter, S. Thater, and A. Frank, "A semantic approach to textual entailment: System evaluation and task analysis," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Association for Computational Linguistics, Prague, 2007, pp. 10–15.
- [46] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017. DOI: 10.18653/v1/p17-1171.
- [47] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," *arXiv preprint arXiv:2203.00916*, Mar. 2022.
- [48] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" *arXiv preprint arXiv:2002.08910*, 2020.
- [49] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural In-*

- formation Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 9459–9474.
- [50] L. Weng. “How to build an open-domain question answering system?” [lilianweng.github.io](https://lilianweng.github.io). (Oct. 2020), [Online]. Available: <https://lilianweng.github.io/posts/2020-10-29-odqa/>.
- [51] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of Documentation*, 1972. DOI: 10.1108/eb026526.
- [52] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, *et al.*, “Okapi at trec-3,” *Nist Special Publication Sp*, vol. 109, p. 109, 1995.
- [53] V. Karpukhin, B. Oguz, S. Min, *et al.*, “Dense passage retrieval for open-domain question answering,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Online, 2020, pp. 6769–6781.
- [54] Y. Hao, L. Dong, F. Wei, and K. Xu, “Visualizing and understanding the effectiveness of bert,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, Nov. 2019, pp. 4143–4152. DOI: 10.18653/v1/D19-1424. [Online]. Available: <https://www.aclweb.org/anthology/D19-1424>.
- [55] Microsoft, *What is semantic kernel?* Online, May 2023. [Online]. Available: <https://learn.microsoft.com/pdf?url=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fsemantic-kernel%2Ftoc.json>.
- [56] S. Casola, I. Lauriola, and A. Lavelli, “Pre-trained transformers: An empirical comparison,” *Machine Learning with Applications*, vol. 9, p. 100334, 2022, ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2022.100334. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827022000445>.
- [57] Y. Liu and *et al.*, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019. [Online]. Available: <https://arxiv.org/abs/1907.11692>.
- [58] Z. Lan and *et al.*, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019. [Online]. Available: <https://arxiv.org/abs/1909.11942>.
- [59] H. Chung, L. Hou, S. Longpre, *et al.*, *Scaling instruction-finetuned language models*, Online, Oct. 2022. [Online]. Available: 10.48550/arXiv.2210.11416.
- [60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, Online, 2019.
- [61] S. Black, S. Biderman, E. Hallahan, *et al.*, “GPT-NeoX-20B: An open-source autoregressive language model,” in *Proceedings of BigScience Episode 5 - Workshop on Challenges & Perspectives in Creating Large*

- Language Models*, virtual+Dublin, May 2022, pp. 95–136. DOI: 10.18653/v1/2022.bigscience-1.9. [Online]. Available: <https://aclanthology.org/2022.bigscience-1.9>.
- [62] S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman, *Gpt-neo: Large scale autoregressive language modeling with meshtensorflow*, 2021.
- [63] B. Wang and A. Komatsuzaki, *Gpt-j-6b: A 6 billion parameter autoregressive language model*, 2021.
- [64] W. Zeng, X. Ren, T. Su, *et al.*, *Pangu-: Large-scale autoregressive pre-trained chinese language models with autoparallel computation*, Version 1, 2021. arXiv: 2104.12369 [cs.CL].
- [65] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, Online, Meta AI, 2023. [Online]. Available: <https://arxiv.org/pdf/2302.13971.pdf>.
- [66] Y. Anand, Z. Nussbaum, B. Duderstadt, B. Schmidt, and A. Mulyar, *Gpt4all: Training an assistant-style chatbot with large scale data distillation from gpt-3.5-turbo*, Online, 2023. [Online]. Available: [https://s3.amazonaws.com/static.nomic.ai/gpt4all/2023\\_GPT4All\\_Technical\\_Report.pdf](https://s3.amazonaws.com/static.nomic.ai/gpt4all/2023_GPT4All_Technical_Report.pdf).
- [67] Y. Anand, Z. Nussbaum, B. Duderstadt, B. Schmidt, A. Mulyar, and A. Treat, *Gpt4all-j: An apache-2 licensed assistant-style chatbot*, Online, 2023. [Online]. Available: [https://static.nomic.ai/gpt4all/2023\\_GPT4All-J\\_Technical\\_Report\\_2.pdf](https://static.nomic.ai/gpt4all/2023_GPT4All-J_Technical_Report_2.pdf).
- [68] Y. Anand, Z. Nussbaum, B. Duderstadt, B. Schmidt, A. Mulyar, and A. Treat, *Gpt4all: An ecosystem of open-source assistants that run on local hardware*, Online, 2023. [Online]. Available: [https://gpt4all.io/reports/GPT4All\\_Technical\\_Report\\_3.pdf](https://gpt4all.io/reports/GPT4All_Technical_Report_3.pdf).
- [69] J. White, Q. Fu, S. Hays, *et al.*, “A prompt pattern catalog to enhance prompt engineering with chatgpt,” *arXiv*, vol. abs/2302.11382, 2023.
- [70] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, and S. Paul, *Peft: State-of-the-art parameter-efficient fine-tuning methods*, Online, 2022. [Online]. Available: <https://github.com/huggingface/peft>.
- [71] H. Liu, D. Tam, M. Muqeeth, *et al.*, *Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning*, Online, May 2022.
- [72] E. J. Hu, Y. Shen, P. Wallis, *et al.*, “Lora: Low-rank adaptation of large language models,” in *International Conference on Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [73] H. Face, *Hugging face - the ai community building the future*, Online, 2021. [Online]. Available: <https://huggingface.co/>.
- [74] S. Oladele, *ML model registry: What it is, why it matters, how to implement it*, Online, 2022. [Online]. Available: <https://neptune.ai/blog/ml-model-registry>.
- [75] Databricks, *MLflow model registry*, Online, 2022. [Online]. Available: <https://databricks.com/product/mlflow-model-registry>.

- [76] W. Jiang, N. Synovic, M. Hyatt, *et al.*, “An empirical study of pre-trained model reuse in the hugging face deep learning model registry,” *figshare*, vol. 2, Feb. 2023.
- [77] A. Gatt and E. Kraemer, “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, 2018.
- [78] C. Van Der Lee, A. Gatt, E. Van Miltenburg, S. Wubben, and E. Kraemer, “Best practices for the human evaluation of automatically generated text,” in *Proceedings of the 12th International Conference on Natural Language Generation*, 2019, pp. 355–368.
- [79] A. B. Sai, A. K. Mohankumar, and M. M. Khapra, “A survey of evaluation metrics used for nlg systems,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–39, 2022.
- [80] A. Celikyilmaz, E. Clark, and J. Gao, *Evaluation of text generation: A survey*, arXiv preprint arXiv:2006.14799, 2020.
- [81] C.-Y. Lin and E. Hovy, “Automatic evaluation of summaries using n-gram co-occurrence statistics,” in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003, pp. 150–157.
- [82] N. Ide and J. Pustejovsky, *Handbook of Linguistic Annotation*, 1st. Springer Publishing Company, Incorporated, 2017.
- [83] E. Reiter and A. Belz, “An investigation into the validity of some metrics for automatically evaluating natural language generation systems,” *Computational Linguistics*, vol. 35, no. 4, pp. 529–558, 2009.
- [84] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318.
- [85] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, 2004, pp. 74–81.
- [86] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.
- [87] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- [88] *Kubernetes*, <https://kubernetes.io/>, Accessed: June 22, 2023.
- [89] M. Anandarajan, C. Hill, and T. Nolan, “Text preprocessing,” in *Practical Text Analytics (Advances in Analytics and Data Science)*, 2nd, Advances in Analytics and Data Science. Springer, 2017, pp. 45–59.
- [90] *Stack Overflow*, <https://stackoverflow.com/>, Accessed: June 22, 2023.
- [91] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *Proceedings of the*

- 2016 *Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, Nov. 2016, pp. 2383–2392.
- [92] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, “Chatrec: Towards interactive and explainable llms-augmented recommender system,” Mar. 2023, Preprint.
- [93] D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch, “Comparison of diverse decoding methods from conditional language models,” in *Proceedings of the Association for Computational Linguistics (ACL)*, 2019.
- [94] N. Chawla, K. Bowyer, L. Hall, and W. P. Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 16, pp. 321–357, Jun. 2002. DOI: 10.1613/jair.953.