The Chinese University of Hong Kong

Final Year Project Report (Second Term)

Build Your Own Chatbot

Author:
CHAU Tak Ho
WANG Weixiao

Supervisor:
Prof. LYU Rung Tsong Michael

LYU2203

Faculty of Engineering
Department of Computer Science and Engineering

April 17, 2023

Abstract

In the first term, we aim to develop a professional conversational Chatbot that can utilize knowledge from documents to respond to questions relevant to a specific field. We have explored various question-answering (QA) methods, including Knowledge-based QA (KBQA) and Extractive QA. We will discuss these methods in detail in the first section of our report, along with a methodology comprising of two parts: Question Decomposition and Question Answering. We will also evaluate and discuss the limitations of our model.

In the second term, our focus is on enhancing public chatbots, particularly Large Language Models (LLMs) like ChatGPT. We will study ChatGPT's mechanisms, performance, and methods to improve its effectiveness in contributing to Natural Language Processing (NLP) research. ChatGPT has gained immense popularity due to its ability to produce high-quality responses, but studies have also revealed its tendency to produce factual and reasoning errors. Like the first part, we concentrate on ChatGPT's QA abilities. We test ChatGPT's performance using eight datasets from five QA benchmarks, evaluate its QA abilities based on the benchmarks' instructions, and compare the results to state-of-the-art models. We will also discuss the issues we encountered while testing ChatGPT and propose methods to enhance its performance.

Acknowledgment

We would like to express our gratitude to our supervisor Professor LYU Rung Tsong Michael and our advisor Mr. Wenxuan WANG for guiding us through the final year project as well as giving us valuable suggestions.

Contents

1 Introduction

1.1 Overview

We approach the study of chatbots in two ways. Firstly, we learn the theories related to chatbots and build one ourselves to better understand their mechanics. Secondly, we analyze and improve existing chatbots to gain a better understanding of their current state. Our work over the two semesters will focus on both approaches.

In the first term, our goal is to develop a Chatbot with professional conversational abilities by focusing primarily on its question-answering (QA) skills within a specific domain. We explored several QA methods, including Knowledge-based Question-Answering (KBQA) and Extractive QA, and finally devised a methodology that combines an Extractive QA model with a question decomposition model. This approach accurately answers domain-specific questions, decomposes compositional multi-hop questions, and leverages background knowledge to enhance performance, as verified by our evaluations.

In the second term, we shifted our attention to studying ChatGPT's QA capabilities, given the growing popularity of Large Language Models (LLMs) like ChatGPT. To assess ChatGPT's performance comprehensively, we conducted an experiment that involved testing ChatGPT with eight datasets selected from five benchmarks. For each dataset, we randomly selected 100 samples to pose questions to ChatGPT and collected its response. We then used the benchmarks' evaluation scripts and various metrics to evaluate its overall performance and compared the results with state-of-the-art models. We also discussed particular cases where ChatGPT's responses were inadequate.

Despite ChatGPT's various applications and high quality, it has some drawbacks that can lead to incorrect responses. During the experiment, we identified numerous issues with ChatGPT's performance. We conducted extensive research on some of

these issues and proposed methods to mitigate these problems and enhance performance. Our findings indicate that removing historical bias and in-context learning can improve ChatGPT's performance. Our research resulted in two pre-print papers.

1.2 Background

1.2.1 Chatbot

Chatbots are computer programs designed to simulate natural human conversation, with the aim of processing user input and producing relevant output [1].

Chatbots are currently being utilized in various domains and applications, with closed-domain chatbots being one of the mainstream applications. Closed-domain chatbots are specifically designed to converse about a particular topic or field [2]. For instance, customer service chatbots are often closed-domain, as they usually respond to frequently asked questions. Closed-domain chatbots possess the advantage of being accurate due to their limited knowledge about a specific topic and are efficient in searching for answers through a smaller amount of data.

1.2.2 Question Answering

Question answering (QA) for closed-domain chatbots, also known as closed-domain QA, is one of their primary tasks. In closed-domain QA, the user's input is a question related to a particular domain, and the output is the answer to that question. There are various types of QA models that can accomplish the task of closed-domain QA [3]. This project explores two types of QA models, namely Knowledge Base Question Answering (KBQA) and Extractive QA. (Detailed descriptions of these two models will be discussed in section 2.) Both models can perform well in closed-domain QA. In this project, KBQA is explored first. KBQA models perform QA using a provided knowledge base that contains information on a specific domain. The KBQA model will search through the knowledge base to find the answer to the question.

Additionally, the Extractive QA model is explored. Instead of a knowledge base, the Extractive QA model uses input documents as its background knowledge, searching through the documents to answer questions.

1.2.3 Large Language Models

A language model (LM) is a model that calculates the probability distribution over text. Recent advances in scaling larger models and training data sizes have led to the development of large language models (LLMs) that excel in many NLP tasks [4].

In addition to the traditional pretraining and fine-tuning approach, LLMs also demonstrate few-shot learning capabilities through in-context learning. This involves using a text or template, known as a prompt, to guide the model to generate answers for specific tasks [5]. GPT-3 [6], a third-generation autoregressive language model, is a prime example, demonstrating impressive few-shot and zero-shot learning abilities. They are widely used in various NLP subfields, including question answering [7, 8]. Previous studies have also shown that designing appropriate prompts manually or automatically can enhance their performance and robustness [4, 9].

Moreover, pre-trained large language models (LLMs) are widely used in various natural language processing (NLP) sub-fields, and among them, ChatGPT is an artificial intelligence (AI) chatbot built on top of OpenAI's GPT-3.5 and GPT-4 families of large language models (LLMs). It has the ability to generate high-quality, human-like responses for a range of user requests, including translation [10], numerical computation [11], summarization [12], and program repair [13].

1.2.4 ChatGPT

ChatGPT (Chat Generative Pre-trained Transformer) is a conversational large language model that was recently developed by OpenAI. It is based on GPT-3.5 and was fine-tuned using both supervised learning and reinforcement learning. Its dia-

logue format allows it to answer various questions, admit errors, challenge incorrect assumptions, and reject inappropriate requests. Since its launch in November 2022, ChatGPT has gained widespread recognition for its detailed and articulate responses in numerous knowledge domains.

Despite its impressive capabilities, previous studies have shown that ChatGPT often generates fluent but incorrect responses. For instance, Borji [14] categorized 11 types of ChatGPT failures, including factual errors, reasoning, math, coding, and bias. Zhuo et al. [15] also found that ChatGPT exhibited social prejudice and toxicity, which could lead to ethical and societal risks. Therefore, we would like to examine the flaws of ChatGPT, especially in its QA ability, and find ways to improve its performance.

1.3 Objective

The Final Year Project topic is "Building Your Own Chatbot," with the main objective of investigating the QA abilities of chatbots. In term 1, our goal is to build a QA system capable of answering questions based on given documents. It should handle complex queries and possess domain-specific knowledge, including identifying specific expressions like terms or synonyms.

Since the launch of ChatGPT, it has exhibited a remarkable ability to generate high-quality, human-like conversations for various user requests. We were impressed and decided to shift our focus to ChatGPT and explore its QA abilities. Research indicates that ChatGPT still has some flaws in its responses, leading us to believe that its QA abilities may not be entirely reliable. Therefore, we conducted experiments and evaluations to determine the performance of ChatGPT. Our findings showed that using the methods of in-context learning and alleviating historical bias can significantly reduce the possibility of incorrect answers and improve QA abilities.

2 Chatbot Construction

2.1 Preliminary

2.1.1 Natural Language Processing

Natural language processing (NLP), also known as computational linguistics, is a branch of computer science that focuses on utilizing computational methods to learn, comprehend, and create information in human languages [16]. The objectives of computational language systems can be varied. The objective may be to facilitate human-human communication, as in machine translation, facilitate human-machine interaction, as in conversational agents, or facilitate both human and machine profit from the massive amount of human linguistic information that is already available online. Question-Answering is one of the objectives of NLP, which is also the focus of this project and will later be introduced.

2.1.2 Knowledge Graph

Human knowledge is always in the form of (subject, predicate, object) or (head, relation, tail). Thus, knowledge graphs are developed to formally represent human knowledge. A knowledge graph is an organized representation of facts that contain entities, relationships, and semantic descriptions [17]. Entities can be real-world objects or abstract concepts, such as a car or a car brand. Relationships represent the relation between entities. Semantic descriptions of entities, and the corresponding relationships are types and properties with well-defined definitions. A knowledge graph is a directed graph with nodes as either subjects or objects and edges from subject to object as relation.

In mathematical notations, a knowledge graph is denoted as $G = \{\xi, \mathcal{R}, \mathcal{F}\}$, where ξ, $\mathcal{R}$ and $\mathcal{F}$ are sets of entities, relations and facts respectively. A fact is denoted as a triple $(h, r, t) \in \mathcal{F}$. Knowledge graph also involves two definitions:

- **Definition 1** A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge [18].

- **Definition 2** A knowledge graph is a multi-relational graph composed of entities and relations which are regarded as nodes and different types of edges, respectively [19].

2.1.3 Question Answering

Question Answering (QA) gives out a human-language question and gives a proper answer for it. Sometimes a specific question is asked and sometimes an open-ended question is asked. Answering goes with building defined systems in which it answers the question that is posted by humans in a natural language format [20]. There are different types of QA systems, and we only focus on closed-domain QA in this project.

Closed-domain QA deals with a particular domain or topic. It is an easier task of asking questions and getting answers because NLP systems are very good at finding a specific topic and retrieving the corresponding answers [20]. Closed-domain QA system gives more accurate answers than the open-domain QA system according to the literature survey of Question Answering Systems [21].

2.1.4 Neural Relation Extraction

Neural relation extraction(NRE) discovers semantic relations between entities from unstructured text using deep learning methods [22]. It is a useful method to represent the information from the original text in a form that is well-structured and could be easily processed by machines. The extracted information is represented as a triple (h, r, t), in which h denotes the head entity, t denotes the tail entity, and r denotes the relation between two entities.

The triples will form a knowledge graph that each triple becomes an edge of it.

Knowledge graphs such as FreeBase and DBpedia are examples of such representations. They are directed and labeled graph-structured data that aim to express such explicit semantics and relations of entities in the triple form [22].

REBEL (Relation Extraction By End-to-end Language generation) is a well-performed NRE model. They use an approach that expresses triplets as a sequence of tokens. In detail, they introduce a set of new tokens as markers, like ¡triplet¿ ¡subj¿ and ¡obj¿, which mark the start of a new triplet, the end of the head entity, and the start of the tail entity surface form, and the end of the tail entity, individually [23]. They extract the triplets in the form of the original text, like the example in Figure 1.

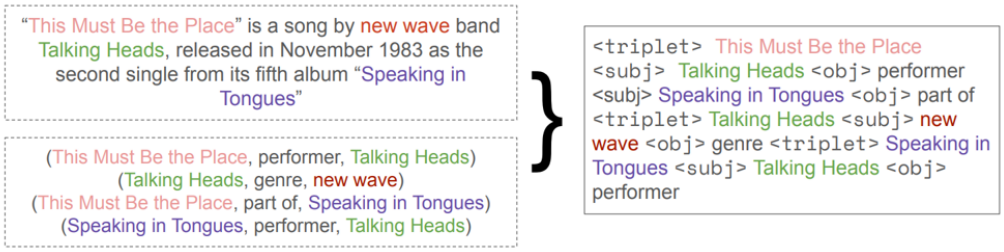

Figure 1: The overall structure of two mainstream methods of complex KBQA [24]

We run the model on a set of documents that are scratched from Wikipedia pages and build a set of graphs from the extracted triples. But as the graphs are in the format of original texts, and need to transform into the format of wide-used knowledge bases like Freebase [25], DBPedia [26], or Wikidata [27] to let most of Knowledge Base Question Answering to run on it.

2.1.5 Knowledge Base Question Answering

Knowledge base (KB) is an organized database with a collection of triple-formatted facts (i.e. subject, relation, object) [24]. It is basically the same as a knowledge graph except it involves formal semantics. Its purpose is to support the modeling of entity relationships. Many large-scale KBs, such as Freebase [25], DBPedia [26], Wikidata [27], have been built to support numerous downstream tasks. One of

them is knowledge base question answering (KBQA), a task that uses KBs as its knowledge source to respond to inquiries in natural language.

The task of KBQA is described formally as follows:

A KB is denoted as $G = \{< e, r, e' > | e, e' \in \xi, r \in R\}$, where $< e, r, e' >$ denotes the relation r between subject e and object e', ξ and R denote the entity set and relation set respectively. Given KB G, the goal of KBQA is to answer natural language question $q = \{w_1, w_2, ..., w_m\}$ in the format of a sequence of tokens W (usually structured with a distinct vocabulary set V) and the predicted answers are denoted as A^*. It is typically assumed that the correct answer A^* can be derived from the entity set ξ of the KB or a natural language sequence. In general, a KBQA model is trained using a dataset $D = \{(q, A^*)\}$.

Simple KBQA and complex KBQA are the two categories of KBQA tasks [24]. Simple KBQA focuses on answering a simple question that involves only a single fact. Complex KBQA focuses on answering complex questions that contain multi-hop reasoning, constrained relations, or numerical operations. Semantic Parsing-based Methods (SP-based methods) and Information Retrieval-based Methods (IR-based methods) are two mainstream ways to achieve the complex KBQA task. The overall structures of the two methods are illustrated in Figure 2. Similarly, both methods make use of topic entity detection to perform the following jobs. The main difference between the two methods is that SP-based methods will transform the question into an executable logic form (i.e. SPARQL) while IR-based methods will retrieve a sub-graph that contains the topic entity, entities that are related to the topic entity and the corresponding relations and perform reasoning over the sub-graph.

RNG-KBQA [28] which uses Freebase [25] as its KB and GrailQA [29] as its training dataset is an SP-based model used for achieving the goal of this project. The overall structure is illustrated in Figure 3. First, for every entity detected in the

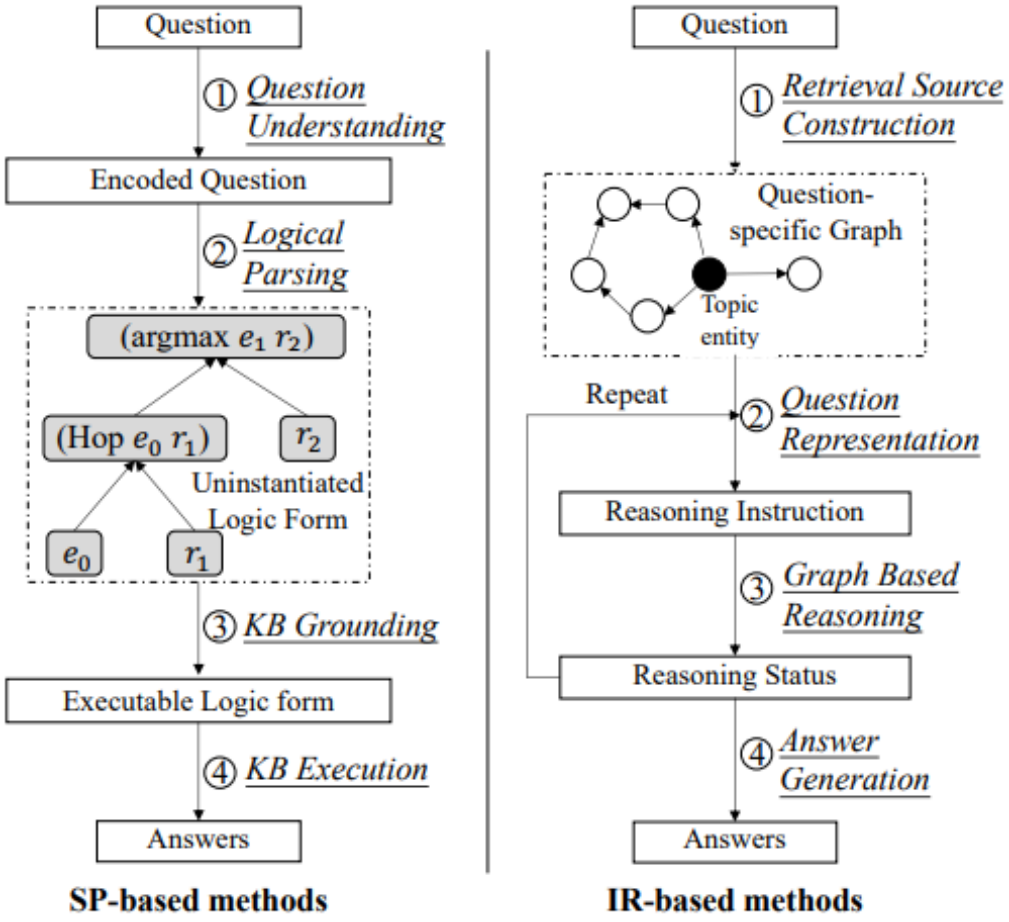

Figure 2: The overall structure of two mainstream methods of complex KBQA [24]

question, the query of the knowledge base for paths reachable within two hops will be enumerated as possible candidates. The candidates will be in logical form. Then, a ranker module will rank the candidates by using a BERT-based encoder to check the similarity between the question and the query. In the generation module, the top-k candidates will be used to generate a target logical form using a transformer-based seq-to-seq model [30] instantiated from T5 [31].

The reason for choosing RNG-KBQA is that it achieves relatively good performance among other models that use GrailQA as the training dataset. GrailQA is designed to test three levels of generalization in KBQA: i.i.d. (i.e. independent and identically distributed), compositional, and zero-shot. Using the model trained based on this dataset could make the QA system more generalizable.

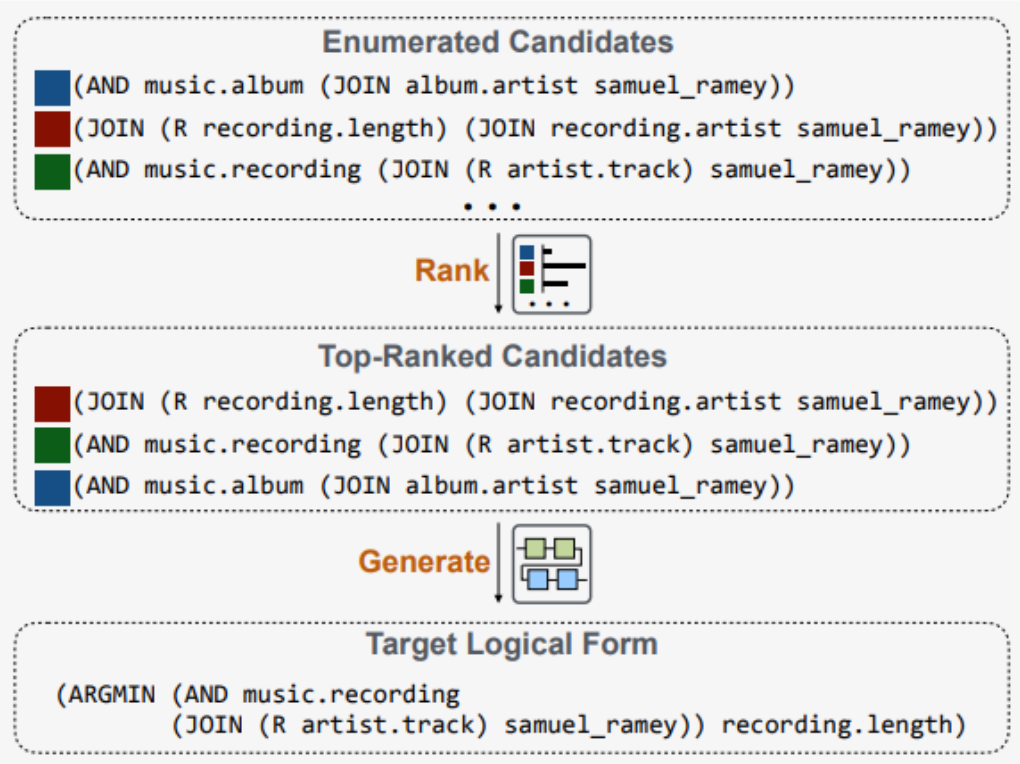

Figure 3: The overall structure of RNG-KBQA [28]

2.1.6 Extractive QA

Given a document and a question regarding the content of the document, extractive QA is a task that aims at understanding the question and finding the answer (i.e. a segment of the document, typically an entity or an explanation) of the question [32, 33]. Currently, many of the extractive QA models use Pretrained Language Models (PLM), such as BERT [34], RoBERTa [35], to achieve the task.

The extractive QA model used in the project (i.e. roberta-base-squad2 [36]) is a RoBERTa-base model. RoBERTa is a transformers model that was self-supervised pre-trained on a sizable corpus of English data. The Masked Language Modeling (MLM) objective was used for its pretraining. When given a sentence, the model randomly selects 15% of the input words to be hidden, after which it must predict the words that were hidden. This makes it possible for the model to learn a bidirectional representation of the sentence.

Roberta-base-squad2 then uses RoBERTa to fine-tune on SQuAD2.0 dataset [37]. The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset made up of questions that crowd workers have submitted about a selection of Wikipedia articles. Each question's answer is either a segment of text, or a span, from the corresponding reading passage, i.e. the context, or the question might be unanswerable. SQuAD2.0 is the newest version of the dataset which combines 100,000 questions in SQuAD1.1 (the older version) with over another 50,000 questions written by crowd workers. By using the SQuAD2.0 dataset, roberta-base-squad2 could be able to perform QA over a given document or text.

2.1.7 Question Decomposition

Multi-hop Reading Comprehension (RC) requires reasoning and aggregation across several paragraphs [38]. Some of the questions asked may be compositional, that is, combined with several sub-questions. In that case, the answer to those sub-questions may be from different paragraphs or documents that a single-hop QA method cannot correctly gather them.

The goal of question decomposition is to convert a multi-hop question into simpler, single-hop sub-questions [38]. Then, those questions can be answered by single-hop QA methods easily, and the final answer can be generated from them.

For DecompRC, it uses a token to replace the answer with a sub-question. The key idea is that, in practice, each sub-question can be formed by copying and lightly editing a key span from the original question, with different span extraction and editing required for each reasoning type [38]. After the answer to a sub-question is found, it replaces the text back into the answer.

There are 3 types of decomposition, namely Bridging, Intersection, and Comparison:

- Bridging is dividing the original question into a sequence of sub-questions, that

the answer of the formal part will be inserted in the text of later parts. Used for the question that could be solved by finding the first evidence in order to find later ones [38].

- Intersection is extracting independent restrictions from the original question, and each restriction forms a sub-question. This is used to solve the questions that have multiple conditions by finding the answer that satisfies all of them [38].

- Comparison is converting a comparison-like question into 2 searching tasks, i.e. find the property of 2 entities. Then, compare them to get the appropriate one [38].

The examples are in Figure 4.

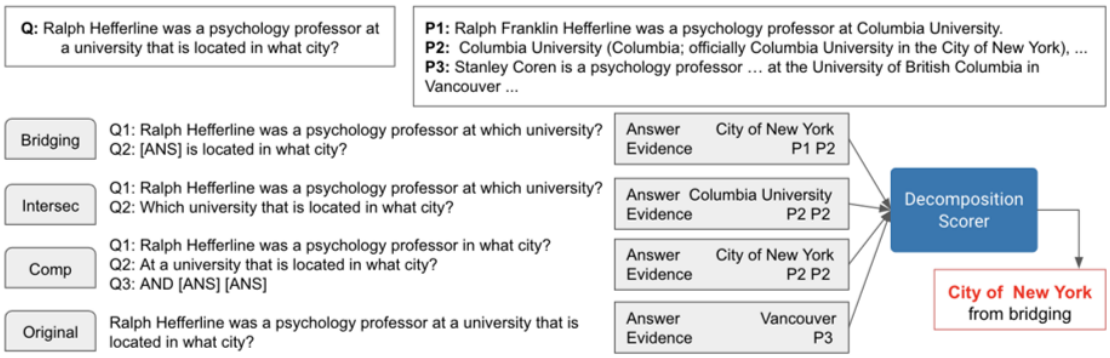

Figure 4: Decomposition, source from [38]

Inside DecompRC, there is a model PointerC, which is a function that points to c indices $ind_1, ..., ind_c$ in an input sequence. Let S = $[s_1, ..., s_c]$denote a sequence of n words in the input sequence. The model encodes S using BERT [38, 34]

$U = BERT(S) \in R^{n \times h}$ where h is the output dimension of the encoder. Let $R^{n \times h}$ denote a trainable parameter matrix. There is a pointer score matrix $Y = softmax(UW) \in R^{n \times c}$ [38].

Where Y denotes the probability that the i_{th} word is the j_{th} index produced by the pointer. Finally, it extracts c indices that yield the highest joint probability at inference [38]. (See Figure 5):

$$\text{ind}_1, \ldots, \text{ind}_c = \underset{i_1 \leq \cdots \leq i_c}{\text{argmax}} \prod_{j=1}^{c} \mathbb{P}(i_j = \text{ind}_j)$$

Figure 5: calculate highest joint probability

2.2 Methodology

In this section, we discuss the architecture of our system and how we trained our models to achieve our goal.

We have different attempts for this purpose. The first approach is using a Neural Relation Extraction model and a KBQA model, which build a knowledge graph and answer questions based on it. But it failed to handle the given documents.

Then, we use another structure that does not modify the document but decomposes the questions to let make it easier to find the answer in the original text. And add the background information by fine-turn the models in advance.

2.2.1 First Approach

Our first approach is to run a Neural Relation Extraction model on the document and then build a knowledge graph from the triples extracted. And we can add extra edges on the graph as background knowledge. After that, we can use a KBQA model to answer questions about them. The first structure of the model is as follows:

1. Build a knowledge graph that includes the background knowledge in a field.

2. Use a Neural Relation Extraction (NRE) model to extract triple from documents.

3. Use a transform function to build a knowledge graph, i.e. the graph of the document forms the triples.

4. Add the relations from the background knowledge graph into the graph of the document.

5. Run KBQA model on the graph of the document.

However, this structure is not practical after experiments and further analysis. One reason is that the extracted triples are not always accurate as the data extraction pipeline of REBEL still keeps some noise, or excludes some relations that are entailed by the text [23]. Especially, the extracted model can not discriminate some proper nouns like terms that will not appear in general sentences. This means the extracted entities or relations may be incomplete themselves.

Moreover, changing the search domain of a KBQA model is inconvenient. Most high-performance KBQA models are based on public knowledge bases like Freebase or Wikidata, which have their own entity set and relation set, and all the triples in it are made with special syntax. For Freebase, the entities in it must be related to a specific Freebase ID, and there is no easy way to align textural entities to them, as the Freebase API is already closed. Also, the relations (called properties in Freebase) must form the logical structure like /automotive/engine/horsepower [39], which makes it even more difficult to generate self-made relations. For Wikidata, it is easier, but all entities and relations also have to align to a specific ID, which means many entities that only appear in the specific context can not be discriminated against. So there may need many extra efforts and will be a significant loss of information if we build our system in this way. So we give up this approach and tried another way to realize the purpose, which will be discussed in the Methodology section.

2.2.2 Model Architecture

Our QA system consists of a question decomposition model, a QA model, and a Ranking model. The overall diagram of the architecture can be found in Figure 6.

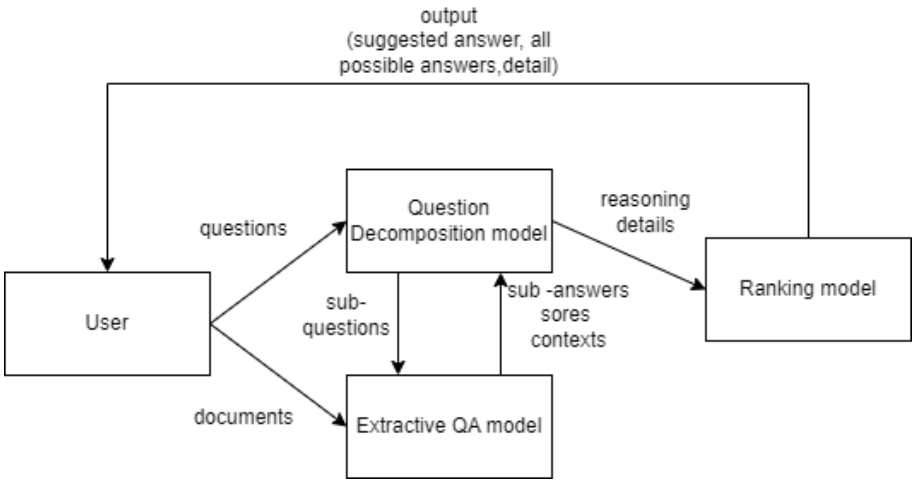

Figure 6: calculate highest joint probability

Question decomposition model The question decomposition model is modified from the decomposition functions of DecompRC [40]. It read the question from the user, if it is a compositional question, the model decomposes it into sub-questions. After the extractive QA model found the answer to each sub-question, the model will also aggregate them to get the final answers to the original question.

QA model The QA model is fine-tuned from roberta-base-squad2 [36]). It is an extractive QA model and it is trained on the basic knowledge of a specific field in advance.

The QA model receives the documents before the QA section, which are related to the field and will be the smaller domain of a conversation period (it is equal to a set of question-answering tasks at the current stage). Then, it will receive questions, search for the answer to each question from the documents, and calculate their score.

The model can generate many possible answers at once, and it has a parameter called top_k to control the number of answers it output each time. In practice, we

set the default top_k = 5, which at most 5 answers with the highest score will be kept each time. Then, it will send the answer together with their score and corresponding context, to the decomposition model.

Ranking model The Ranking model is used to rank the final answers. As there may be more than 1 possible answer for each sub-question, when we decompose by bridging, it will influence the next sub-questions, then lead to different final answers. So we use a ranking function to rank and find the most possible final answer. It receives the final answers and their reasoning details, i.e. the score of all sub-answer in the intermediate steps, and ranks a final answer by those scores. In practice, We simply calculate the score of the final answer as the product of sub-answers in the same answering sequence, i.e. $S_{final} = S_1^{p_1} \times ... \times S_n^{p_n}$. That S_i denotes the score of i_th sub-answer, and p_i is the parameter, we use p = 1 so that all sub-answers have the same weight.

After that, it will generate the suggested final answer, i.e. the most possible one, and other final answers will be treated as possible answers and output together in case the suggested one is not correct, and for easy evaluation. The reasoning details are also stored, and there is an option to also output them.

Running procedure The whole system will receive documents from the user first. After that, it will receive questions from the user. There is a flow chart of the overall procedure in Figure 8.

For each question asked by the user, using bridging as an example, it will use the question decomposition model to generate single-hop sub-questions. Then the extractive QA model will find the answers to the sub-questions, together with the context and score. Then, those sub-answers will form different next sub-questions and each of which will send to the extractive QA model. It repeats the process until getting the final answers. Then, the answer ranking model will calculate the score

of each final answer. After all, the system will output the most possible answer, and all other possible answers, and their reasoning details as an option. There is a diagram of how the sub-questions and answers are generated in Figure 7.

The model could also output the processing detail for each of the final answers, e.g. the intermediate sub-answers, their scores, and corresponding contexts, for a better understanding of its working procedure, or for evaluation.

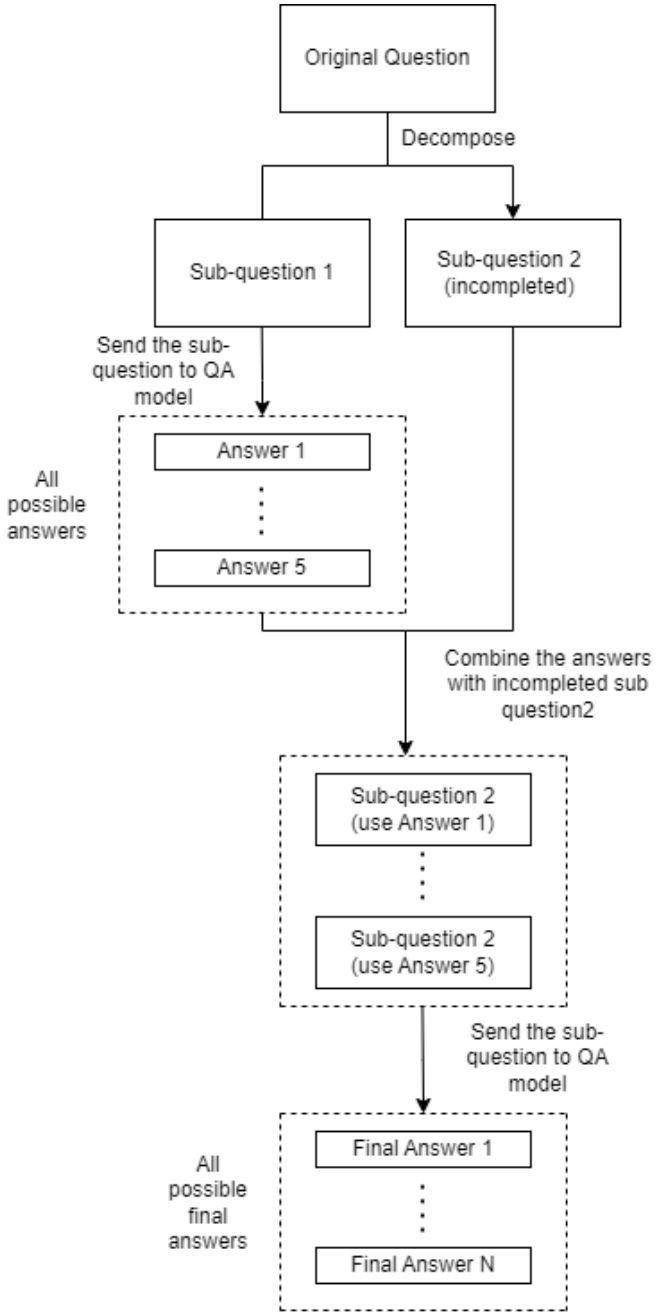

Figure 7: The diagram of how the sub-questions and answers are generated

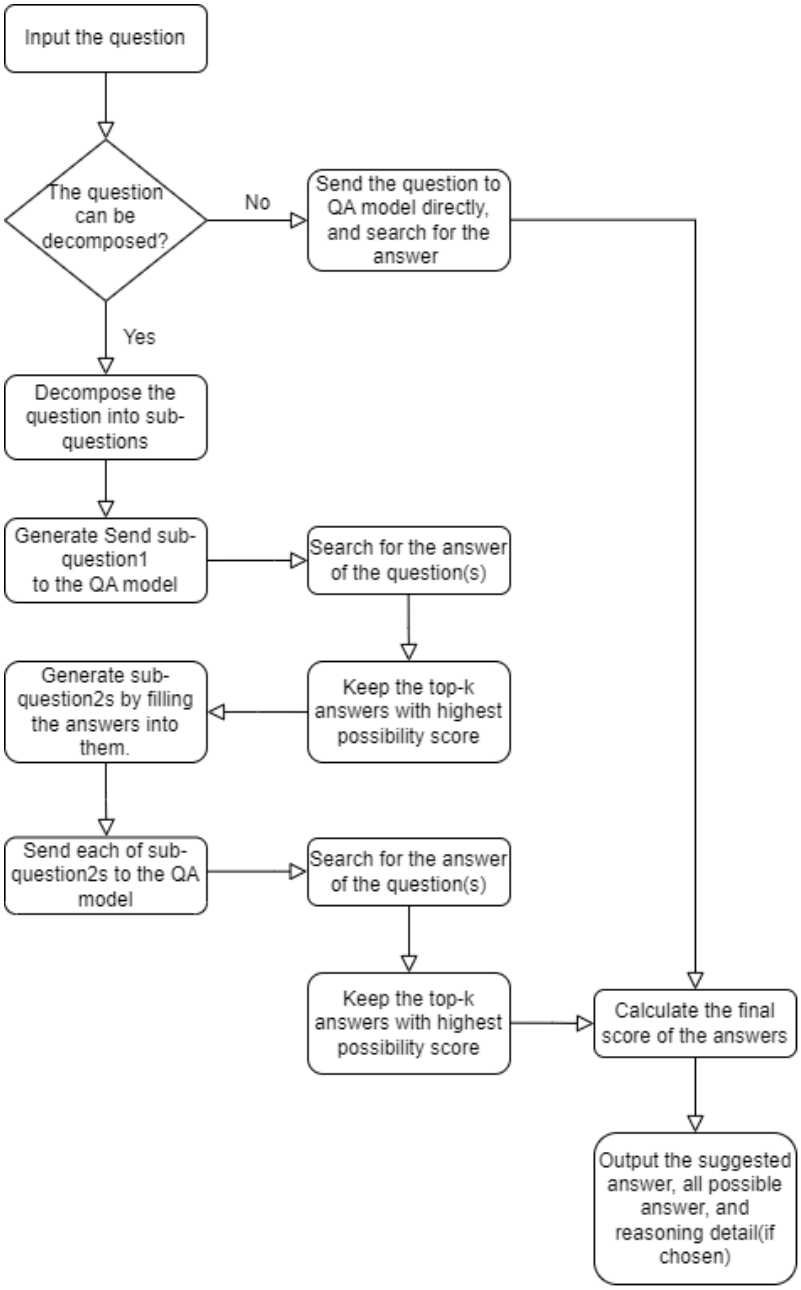

Figure 8: The flow chart of the running procedure of the QA system.

2.2.3 Model Training

The extractive QA model (i.e. roberta-base-squad2 [36]) can be fine-tuned using the toolkit provided by Haystack [41]. The fine-tuning dataset can be created using Haystack's annotation tool. Users can input a document into Haystack's annotation tool, construct custom questions regarding the document, then highlight a specific section of the document as the answer to the question. The annotated document and question can be output in SQuAD format.

The reason for fine-tuning the extractive QA model is to let the model have training data on a specific domain so that it may better understand some of the vocabularies that appear a lot in that domain. Thus, a survey of knowledge graph [17] is used as the document of the training dataset. An example of a custom question dataset is in Figure 9.

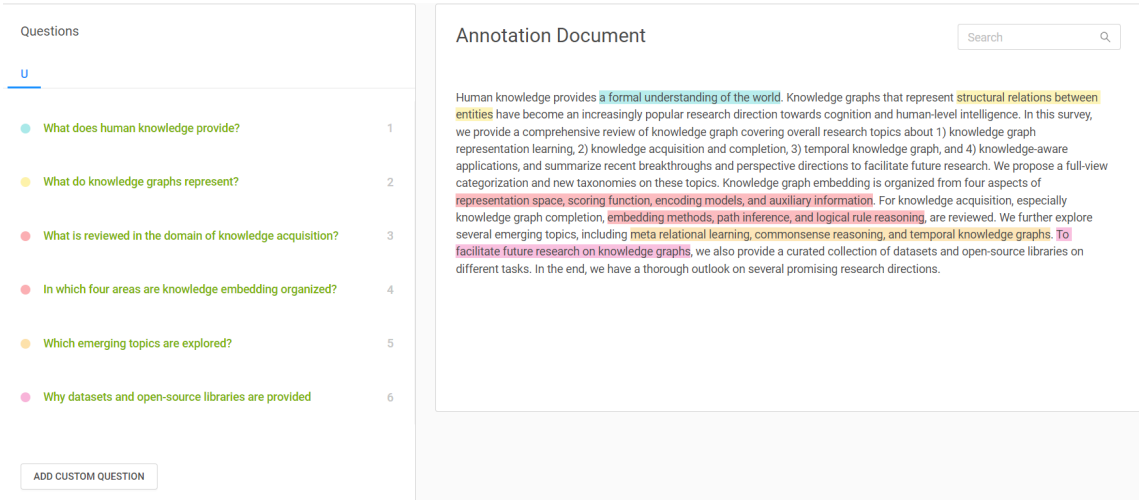

Figure 9: An example of creating a custom question dataset using the annotation tool provided by Haystack [41]. The column on the left side is the custom questions created and the answers regarding each question are highlighted in the text under "Annotation Document".

There are a few key points to the custom question design process:

- The questions should contain words that are used especially in this domain and are uncommon outside this domain.

- The questions should be fact-seeking questions that can be answered by an entity or an explanation.
- The question should focus on the material in the provided document and should not require additional information.
- The questions should be a reformulation of the answer and should sound as natural as possible.

The toolkits provided by Haystack take the dataset created by the annotation tool as input and the original model (i.e. roberta-base-squad2) as the base model for fine-tuning. Then, the fine-tuning model will be used for testing and evaluation.

2.3 Experiment

2.3.1 Experimental Setup

In this section, we describe how we evaluate our model. We made several datasets based on information from different fields, they consist of a/some document(s) and a set of questions related to those documents, we test the system by asking those questions, and examining whether it can

- Decompose the sentence correctly.
- Find the correct context that contains the answer.
- Output the correct format of the answer, especially for those containing terms or special phrases.

We also test how the background knowledge helps improve the accuracy by testing the model on questions that need the information to get an answer, we run both the trained and untrained models on it, and compare their results, to ensure our QA system does not pass the testing because of the help of background knowledge, instead of by coincidence.

2.3.2 Data

To evaluate our model, we manually build several evaluation datasets. Each of them includes some documents and a set of Questions that ask for information from the document. The documents are from a different domain for each evaluation phase.

To test its general performance, we use 2 manually made documents.

One of the datasets is about bank services, the document is made from a list of services. As most of the services are independent, the questions are mostly simple questions. it evaluates our model's ability to find the correct context and output

the correct and output the correct format of the answer, as many of its services have a similar name or similar diction. So it needs to discriminate the sentences about different services.

the other dataset is about NLP, whose document is extracted from Wikipedia, and about the knowledge of NLP, to build another evaluation dataset. It includes many compositional questions, so we use it to evaluate the ability to decompose the compositional sentences.

The questions in a dataset are manually modified from the original sentences in the related document. We simply convert a sentence or some sentences that introduce something into a question that asks for some information from it. An example is in Table 2.3.2

The questions can be classified into simple questions, which can search for the answer directly, and compositional questions, which need to decompose and answer step by step.

To make a compositional question, we combine simple questions together. For example, if we make a compositional question by 2 simple questions, namely question1 and question2. The answer to question1 will be the component of another question2. Then, we combine the questions by replacing the replaceable words in question1 with a clause that is deformed from question2. An example of making a compositional question from simple questions is in Table 2.3.2.

As the datasets are from different domains, we also use models that are trained on the basic knowledge of the corresponding domain. So that we can evaluate whether the training helps them discriminate the terms and proper nouns. There are 2 datasets used for this evaluation task, the information about them are in Table 2.3.2

Context: Statistical methods in NLP research have been largely replaced by neural networks [42]
Question: Many different classes what algorithms have been applied to natural-language-processing tasks
Correct answer: machine-learning algorithms
Incomplete answer: machine-learning **Cumbersome answer:** classed of machine-learning algorithm

Table 1: Example of generating question from original text

Context 1: Statistical methods in NLP research have been largely replaced by neural networks [42] **Simple Question 1:** What replaced statistical methods in NLP research? **Answer 1:** neural networks
Context 2: In more practical terms neural networks are non-linear statistical data modeling or decision making tools [42] **Question 2:** Which kind of tools does neural networks belongs to? **Answer 2:** data modeling or decision making tools
Compositional Question: Which kind of tools does the method that replaced statistical methods in NLP belongs to? **sub question 1:** What replaced statistical methods in NLP research? **Answer 1:** neural networks **sub question 2:** Which kind of tools does [ANSWER] belongs to? **Answer 2:** data modeling or decision making tools **Final Answer:** data modeling or decision making tools

Table 2: Example of making a compositional multi-hop question from simple questions

To test whether the background knowledge makes an effect, we build a dataset that was also based on the document of NLP but modified the original sentences by deleting many descriptions that helps us understand it. And the diction of questions will also be set to be different from the document. This modification makes it harder to answer by simply matching the keywords. There are 20 questions in this dataset, an example of modified text and questions is in Table 2.3.2.

Field	Document	Simple Question	Compositional Question
NLP	9426 words	20	28
Bank Service	4452 words	93	5

Table 3: Data used to evaluate our QA system. There a 2 datasets build from texts about NLP and Bank Services for each. The length of the document and the amount of 2 types of questions are shown in the table

Original text: Natural language understanding (NLU) convert chunks of text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate [42].
Modified text: Natural language understanding convert chunks of text into more formal representations.
Question: What does NLU do?
Answer of model with background information: convert chunks of text into more formal representations.
Answer of untrained model: Natural-language generation

Table 4: An example of modified text, and a question asked. as the untrained model does not know what is NLU, it just finds a task of another application.

2.3.3 Evaluation

We then evaluated our model on those datasets.

For decomposition, we test: Whether the decomposition is correct in syntax and whether the answer asked by sub-questions can be aggregated to the final answer.

Generating the question word-by-word is known to be a difficult task that requires substantial training data and is not straightforward to evaluate [38]. To be easier to perform the evaluation, we just test whether the sub-questions can be answered by the QA model and whether the sub-answers, if the model answers correctly, can lead to the final answer.

For finding context, we test: Whether the context that the QA model detect is including the correct answer. So We compare the context by the model and the "correct" context, if the former one includes the latter one, or includes the parts that contain the answer, then it is correct, otherwise, it is wrong.

For the output of the answer, we test whether it is complete and in the correct format. i.e. intelligible and including all required descriptions. The examples of correct and incorrect answers are in Table2.3.3.

Context: Many different classes of machine-learning algorithm have been applied to natural-language-processing tasks [42]
Question: Many different classes what algorithms have been applied to natural-language-processing tasks
Correct answer: machine-learning algorithms
Incomplete answer: machine-learning **Cumbersome answer:** classed of machine-learning algorithm

Table 5: Example of answers in the correct and incorrect form

2.3.4 Result

Table 2.3.4 shows the result of question decomposition and Question answering test, we can see that the model is able to find the correct answer to most of the questions. For those it outputs the incorrect answer, in the end, it also detects the correct answer, but it treats the correct answer as less possible than other answers. We also test some multi-hop questions that need the knowledge of terms and synonyms, the models can also solve them.

Table 2.3.4 shows the result of the background knowledge test, we can see that the untrained model passes some cases by coincidence, but the accuracy is far lower than the trained one.

Dataset	Decomposition	Context	Answer
NLP	21 of 28	37 of 41	34 of 41
Bank Service	5 of 5	89 of 98	86 of 98

Table 6: The result of running our trained models on the evaluation datasets. The accuracy in decomposition, finding context, and output answer are shown in the table

Model	Correct context	Correct Answer
trained	18 of 20	15 of 20
untrained	4 of 20	3 of 20

Table 7: The result of the trained and untrained models on the background knowledge testing dataset. The result shows that the trained model behaves well in most of the cases, but the untrained model can only pass a few of the cases.

2.4 Discussion

2.4.1 Analysis

Our QA system is capable of accurately answering questions within a specific domain, decomposing compositional multi-hop questions, and making good use of background knowledge to improve performance.

The result on most of the questions during the experiment is good, but some drawbacks of the system are found. After further analysis of the format of some questions it fails to answer, it shows that the system only performs well under some conditions. That is:

1. The compositional questions must be intuitively compositional, i.e. can decompose into several sub-question from the original question text literally.

2. The answer to each sub-question must appear in the documents literally.

3. All the words in the answer text must appear in the context and must be successive. Both the decomposition model and extractive QA model cannot change the order of the appearance of words without manual annotation.

We test the model with several questions that violate this restriction, and the output proves our assumption: this system has some limitations and thus is not useful for all kinds of questions.

Then, we come up with some methods to avoid or reduce these kinds of errors. The

methods are discussed in the improvement section and future work section.

2.4.2 Limitation

The limitations of the model are mainly from the limitations of the decomposition model, and the limitations of extractive QA. Some limitations are inherited from DecompRC like some questions are not compositional but require implicit multihop reasoning, hence cannot be decomposed. Second, there are questions that can be decomposed but the answer for each sub-question does not exist explicitly in the text, and must instead be inferred with commonsense reasoning [38]. The cases are shown in Table 2.4.2 and Table 2.4.2.

For extractive QA, there are also some limitations in finding the context and getting the answer. When finding the context, we found that the current model can not recognize some tones, like negative tones, or some conditions, like time conditions. When Sentences have similar diction but differentiate in these conditions, the current model could not distinguish them and thus may fail to find the correct context. The cases that fail to find the context are shown in Table 2.4.2

Also, the answers must appear in the document literally, but some final answers can not be directly extracted from the test, and need to be restricted or represented in other words. The cases are shown in Table 2.4.2

2.4.3 Improvement

After the analysis of the failure cases, there are some plans to improve the behavior of the system.

Firstly, we will make a threshold of output, that judge whether the suggested answer is appropriate, and thus prevents the model from outputting low-possibility answers. When all the answers' final scores are lower than the threshold, the question is treated as "no answers". The value of the threshold could be decided after further

Context1: Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so [43]. **Context2:** Decision trees is the earliest-used machine learning algorithm that produced systems of hard if-then rules similar to existing hand-written rules [42].
Question: Decision trees build a model based on what?
subQ1: decision trees build which model? **subQ2:** [ANSWER] based on what?
Correct subQ1: What kind of algorithm does decision trees belong to? **Correct subQ2:** [ANSWER] build a model based on what?
Final answer: sample data

Table 8: A typical failure case of our model, which is because of the failure of decomposition. The required multi-hop reasoning is not compositional intuitively. The correct decomposition needs some extra information.

Context: Most higher-level NLP applications involve aspects that emulate intelligent behaviour and apparent comprehension of natural language [42].
Question: higher-level NLP applications involve aspects that emulate what?
subQ1: higher-level NLP applications involve what aspect? **subQ2:** [ANSWER]emulate what?

Table 9: A typical failure case of our model, which is because of the failure of decomposition. The required multi-hop reasoning is compositional intuitively, but the sub-question Q1 does not have answers, this lead to the failure of answering the original question. To answer this question correctly, the model needs to treat it as a non-compositional question.

experiments.

After that, we could train the model to understand some implicit relations, like inclusion or opposition. So that, when it could not find an acceptable answer by original question text, it could change the subject or scope, to solve questions similar to the case in Table 2.4.2.

To solve similar cases like those shown in Table 2.4.2, it may need a structure to restore the intermediate abstract concept, that is not shown in the document literally, For example, "The aspect" in that case.

To improve the accuracy of finding the corresponding context and output the correct format of the answer. There needs to be a model that detects the condition, restric-

Question: How can I apply for a Credit Card in Hong Kong before leaving my home country?
Context found: After you leave your home country and move to Hong Kong, you can apply for a Credit Card by visiting one of our branches... **Correct context:** You can apply for a Credit Card even before you leave your home country. Fill out an application and provide all the necessary documents and we will review your application...

Table 10: The extractive QA model can not discriminate the sentences when they have similar diction but different tones or conditions. In this case, the model ignores the words "before" and "after", and selects the former context as it has more matching keywords

Case1: The reasoning logic is implicit.
Context: Many words have more than one meaning; we have to select the meaning which makes the most sense in context [42].
Question: What kind of words we need to select the meaning of them?
Answer: Many words **Correct answer:** words that have more than one meaning
Case2: Answers directly extracted from context are not complete or miss some restrictions
Context: Machine translation Automatically translate text from one human language to another [42].
Question: Machine translation translate text into what?
Answer: another **Correct answer:** another human language

Table 11: The cases that questions can not answer by extracted QA model, although it is decomposed and has an answer.

tion, or tone of sentences, to help match the corresponding context, and describe the entities accurately.

Also, the training on background information could be improved. Current training includes discriminating proper nouns or expressions like abbreviations. We could train the model by understanding the variants of them, and the relations between them. Also, current training still relies on manual work, it is time-costing and makes it hard to put our models in different fields, as they need to be trained on different basic knowledge for each field. The decomposition model should also be fine-tuned to perform more accurately.

3 Empirical Study on ChatGPT

3.1 Experiment

3.1.1 Experimental Setup

The purpose of the empirical study on ChatGPT is to evaluate the QA ability of ChatGPT, we chose 8 datasets from 5 QA benchmarks, namely SQuAD, HotpotQA, SuperGLUE, CommonsenseQA, and TruthfulQA, to test the QA ability of ChatGPT on different areas. The experiment is conducted as follows:

1. For each dataset, we randomly select 100 test samples of the dataset and test ChatGPT by utilizing the prompts suggested by ChatGPT. We adjust the prompts necessary to match the output formats of different datasets.

2. We evaluate the performance of ChatGPT based on the metrics used in different datasets.

3. We compare the performance of ChatGPT with state-of-the-art (SOTA) models of each dataset to see the gap between ChatGPT and SOTA models.

3.1.2 Dataset

SQuAD The Stanford Question Answering Dataset (SQuAD) [37] is a reading comprehension dataset composed of questions about selected Wikipedia articles. Each question in SQuAD requires an extractive answer that is either a span of text from the corresponding reading paragraph or the question is unanswerable. SQuAD 1.1, the previous version, only contains answerable questions. In contrast, SQuAD 2.0 combines answerable questions from SQuAD 1.1 with over 50,000 new, unanswerable questions. These unanswerable questions are relevant to the paragraph and have a plausible answer within it. 50 answerable questions and 50 unanswerable questions will be used as the 100 test samples.

HotpotQA HotpotQA [44] is a question-answering dataset featuring natural, multi-hop questions, with strong supervision for supporting facts to enable more explainable question-answering systems.

SuperGLUE SuperGLUE [45] is a newly developed benchmark that aims to rigorously evaluate language understanding. It includes an analysis toolbox, a performance metric, and a public leaderboard. There are 8 datasets in SuperGLUE, each focusing on a different area of language understanding. We chose to test ChatGPT's Question Answering capabilities using only the three QA datasets from SuperGLUE: BoolQ, MultiRC, and ReCoRD.

- BoolQ [46] is a QA task in which each example consists of a short paragraph and a yes/no question about the paragraph. The questions are submitted voluntarily and anonymously by Google search engine users and are then matched with a passage from a Wikipedia page that contains the answer.

- MultiRC [47] is a QA task in which each example consists of a context paragraph, a question about that paragraph, and a list of possible answers. The system must predict which answers are true and which are false.

- ReCoRD [48] is a multiple-choice QA task in which each example consists of a news article which is drawn from CNN and Daily Mail, a Cloze-style question about the article in which one entity is masked out, and a list of possible entities in the news article where the same entity may be expressed using various distinct surface forms, all of which are considered correct.

CommensenseQA CommonsenseQA [49] is a dataset designed to test a model's common sense reasoning ability. It consists of single-selection multiple-choice questions that require the model to choose the most appropriate answer based on commonsense reasoning.

TruthfulQA TruthfulQA[50] is a dataset consisting of 817 questions from 38 categories such as health, law, finance, and politics, that measures whether a language model generates truthful answers. The questions are in a retrieval format where the task is to select the true claim from a set of common misconceptions. The dataset contains three tasks, namely Generation, MC1, and MC2. Generation task requires the model to generate the answer. MC1 task requires the model to choose one answer from a given list of choices, while there may have multiple correct answers in the list of choices of MC2, and the model is required to choose all of them.

3.1.3 Evaluation

We have collected all the results in Table 12 for easier viewing, and compared them with current state-of-the-art models. Some state-of-the-art models are fine-tuned on the dataset while others are zero-shot, depending on the dataset. It is important to note that ChatGPT has not been trained on any training set of these datasets, so its results can be considered as zero-shot.

Dataset	**Metric**	**SOTA**	**ChatGPT**
SQuAD2.0	EM	90.578	39.0
	F1	92.978	45.35
HotpotQA (Distractor set)	EM	67.46	40.00
	F1	80.52	55.30
SuperGLUE (BoolQ)	Acc	92.4	73.0
SuperGLUE (MultiRC)	EM	65.8	61.90
	F1	89.6	88.421
SuperGLUE (ReCoRD)	EM	95.9	78.00
	F1	96.4	79.73
CommonsenseQA	Acc	78.2	80.0
TruthfulQA(MC1)	Acc	-	69.0
TruthfulQA(MC2)		-	73.9

Table 12: Performance of ChatGPT on different datasets, compared to the state-of-the-art. As commonsenseQA allows models with or without the help of extra-training data and recorded the result under both conditions, We only compare with the SOTA without extra training data here.

The results indicate that ChatGPT's performance on SQuAD2.0 is much lower than that of the state-of-the-art models. However, the reason for such poor performance on SQuAD2.0 is that ChatGPT is unable to handle or distinguish between answerable and unanswerable questions. In Table 13, the result shows that ChatGPT has a competitive performance compared to SOTA model when answering answerable questions, while its poor performance on answering unanswerable questions makes its overall performance on SQuAD2.0 bad.

	EM	F1	Manually
Overall	39.0	45.35	62/100
50 unanswerable	0	0	14/50
50 answerable	78.0	90.70	48/50

Table 13: Data statistics of ChatGPT on SQuAD 2.0. The result shows that ChatGPT is much better at answering answerable questions than unanswerable ones. The reason for 0 in EM and F1 is that ChatGPT will generate a response no matter what while only an empty answer will be considered correct in unanswerable questions. Therefore, we manually compare ChatGPT's outputs to the correct answers for the samples, and the result is shown in the column "Manually".

The result of HotpotQA shows that the multi-hop reasoning capability of ChatGPT is not good. Regarding ChatGPT's performance on the SuperGLUE dataset, its accuracy on BoolQ is much lower than that of the state-of-the-art model, and its performance on ReCoRD is also much lower than the state-of-the-art model. However, the performance on MultiRC is very close to the state-of-the-art model. Additionally, ChatGPT's capability in doing MC questions is good. It outperforms the zero-shot state-of-the-art on commonsense reasoning, and its accuracy on TruthfulQA substantially exceeds that of previous language models.

3.1.4 Discussion

The results indicate ChatGPT's ability on various QA tasks, however, there are several factors that may affect its performance.

Since ChatGPT generates long detailed answers without constraints, if the output format is not specified, the performance could significantly decline. For example, in the SQuAD extraction task, without the prompt "in a word or a phrase," the F1 score drops to 23.53. Note that even if ChatGPT generates the prompts, they may not always work as intended. For instance, the prompt "Choose the correct answer" for single-selection questions does not limit the number of options, resulting in occasional multiple-choice output. Similarly, when testing questions related to

essays, extra instructions on the desired output format are necessary; otherwise, ChatGPT will generate a long paragraph.

Furthermore, ChatGPT has a limited capacity for reading long inputs, which may result in it truncating the input and only reading the first part. To ask a long question, one solution is to divide the input into parts and send them to ChatGPT with appropriate https://www.overleaf.com/project/643c9f8f7acdc81fc6eb7df1prompts. However, ChatGPT forgets the previous dialogue after several turns, thus limiting the input length.

During MC question tests, ChatGPT occasionally ignores the output format declared in the prompts and generates a sentence instead of an option. This error slightly reduces its accuracy, but it occurs infrequently. It may also modify the content of the option or use a different output format, but these conditions do not affect the accuracy criteria. These issues can be resolved by reiterating the output format. Additionally, ChatGPT may not remember the original text given, leading to inaccuracies with format-sensitive cases, such as people's names, verb tenses, or noun singular and plural forms. For example, it may ignore a person's middle name or output "hares" instead of "a hare." It may also fail to answer questions such as "Who is Angelina Julie dating?" as it cannot distinguish different tenses. These cases are shown in Figure 10, 11, and 12.

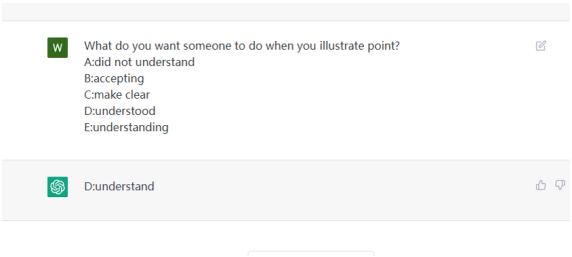

Figure 10: "understood" changed into "understand"

In the ReCoRD task, when an entity is needed as a prefix, ChatGPT often outputs the full noun instead of the prefix. However, when an entity is needed as a suffix,

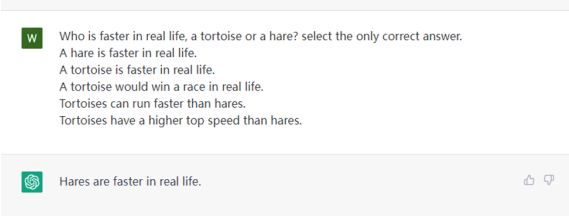

Figure 11: "a hare" changed into "hares"

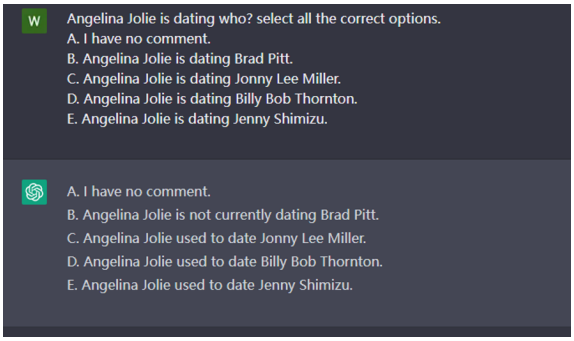

Figure 12: it fails to discriminate different tenses

the error of providing the full noun is low. In 100 questions, ChatGPT made the mistake of providing the full name of the entity instead of the prefix five times when filling in the entity as a prefix.

It can be challenging to ask ChatGPT to select from candidates without options. This difficulty may be due to its memory mechanism. Including options (such as "A. B. C. D.") can aid ChatGPT in recalling the choices ("targets" in the TruthfulQA dataset), which can greatly enhance its performance. In TruthfulQA tests, when we removed the options from the choices in our prompts and allowed it to choose from pure text, the accuracy dropped significantly, as shown in Figure 13.

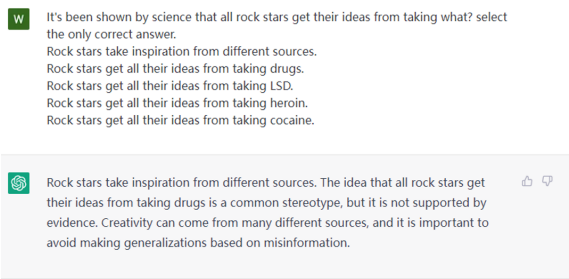

Figure 13: it outputs a paragraph when answering MC questions

We also discovered that ChatGPT may produce inconsistent answers. For example, it may output different answers to the same question, as shown in Figure 14. To investigate this issue, we conducted further tests on SQuAD, HotpotQA, and TruthfulQA, randomly selecting 30 samples from each dataset. We asked each question three times and compared the answers manually. In SQuAD, 12 out of 30 questions received different answers, but most of the differences did not affect the meaning, such as quotation marks on special expressions, additional conjunctions, or random order of multiple answers. However, three of them were genuinely different. In HotpotQA, ChatGPT generated different answers for 17 questions, and 11 of them had different meanings. This may be because ChatGPT has a higher likelihood of producing different answers when it has lower confidence in the correct answer. We observed that ChatGPT performs better on SQuAD than on HotpotQA. This issue also affects MC questions, but when we tested it on the 30 questions from CommonsenseQA, none of them produced different answers.

To determine if ChatGPT remembers its previous choices, we shuffled the choices (i.e., each choice had a different option each time) and tested it again. After this modification, five out of 30 questions received different answers in all three attempts, as shown in Figure 15.

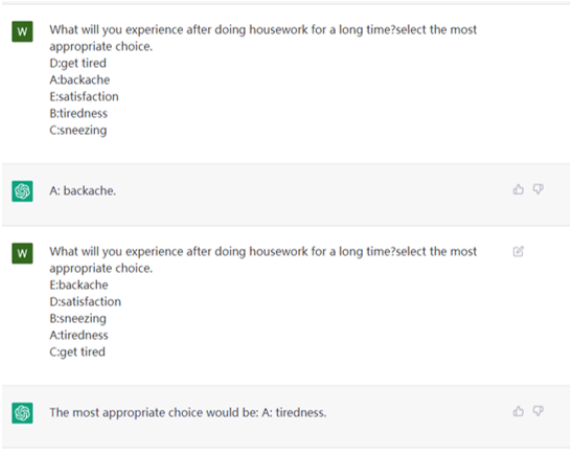

Figure 14: There are nothing changed in the question, but the answers are contrary.

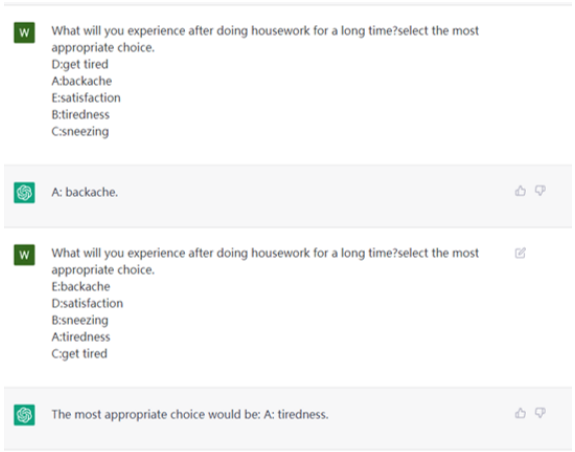

Figure 15: the answer is changed when we shuffled the options

Dataset	different expressions	different meanings
SQuAD	12/30	3/30
HotpotQA	17/30	11/30
CommonsenseQA	5/30	5/30

Table 14: ChatGPT can produce varying answers for the same question, with some differences in phrasing and even in meaning. The frequency of such occurrences varies across different datasets. SQuAD and HotpotQA are essay-type questions, while CommonsenseQA comprises multiple-choice questions. ChatGPT finds HotpotQA more challenging than SQuAD, and it also generates varying answers more frequently to questions in HotpotQA.

3.2 Improving Methodology

During the testing of ChatGPT's QA abilities, we observed that the instructions given by the user had an impact on the resulting performance of ChatGPT. We utilized this observation and proposed two methodologies to improve ChatGPT's performance: removing historical bias and in-context learning. We conducted experiments to prove the effectiveness of these methodologies. The following sections will discuss the details of these two methodologies and the experiments conducted.

3.2.1 In-context Learning

LLMs have the ability to perform in-context learning (ICL), according to research [51], which means they can learn from a few examples within a particular context.

Figure 16 illustrates the general flow of ICL. With ICL, LLMs can undertake a wide range of challenging tasks, including answering problems that require mathematical reasoning [9].

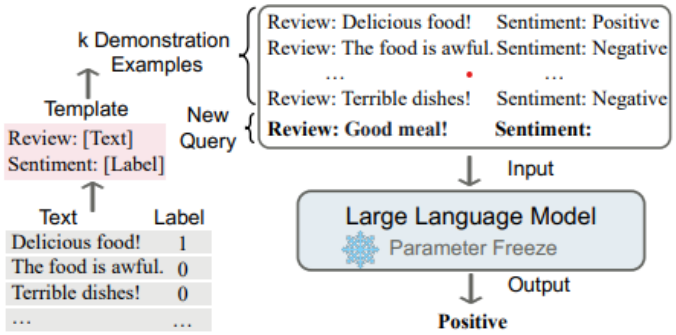

Figure 16: The overall flow of ICL [51], involves providing a demonstration context that includes several examples written in natural language templates. With this context, language model models (LLMs) can generate predictions based on both the demonstration and a query. Then, LLMs utilize the provided examples and generate accurate predictions for the given query.

Formally, we are given a query input text x and a set of candidate answers $Y = \{y_1, ..., y_m\}$, which can be either class labels or free text phrases. A pre-trained language model M selects the candidate answer with the highest score as the prediction while being conditioned on a demonstration set C. This set contains an optional task instruction I, along with k demonstration examples, and can take the form of either $C = \{I, s(x_1, y_1), ..., s(x_k, y_k)\}$ or $C = \{s(x_1, y_1), ..., s(x_k, y_k)\}$. Here, $s(x_k, y_k, I)$ represents an example written in natural language texts that pertains to the task at hand.

Since ChatGPT's performance in answering unanswerable questions in SQuAD2.0 is poor, we conducted an experiment using unanswerable questions from SQuAD2.0 to evaluate the effectiveness of ICL. The experiment was designed as follows:

1. ChatGPT will receive an instruction prompt that reads: "I will give you a paragraph and a question. If there is no information about the question in the paragraph, the question is unanswerable. If it is the case, respond with * to

indicate unanswerable question.". Given that ChatGPT always generates a response, the asterisk is used to confirm that ChatGPT has correctly understood the instruction and has determined the question to be unanswerable.

2. Two unanswerable question samples are randomly selected as demonstration examples. The prompt is : "Example 1: Paragraph: [paragraph 1] Question: [example question 1] Answer: * Example 2: Paragraph: [paragraph 2] Question: [example question 2] Answer: *"

   The instructions will be reiterated at the conclusion of the demonstration examples. The instruction prompt is as follows: "In the two examples, the answer to both questions is empty because the question is unanswerable. When the paragraph does not provide information about the question or the question has a false premise, the question is unanswerable. Therefore, when you encounter an unanswerable question, please output "Answer: *" to indicate an unanswerable question."

3. The 50 unanswerable questions that were used in the empirical study will be tested again. The prompt for each question will be "Paragraph: [paragraph] Question: [question]." To ensure that ChatGPT remembers the instructions, we will use the "Edit" option to input the questions. This guarantees that each unanswerable question will be asked immediately after the demonstration examples.

4. We count the number of asterisks that appear in ChatGPT's response and manually verify whether ChatGPT is able to recognize unanswerable questions. Additionally, we compare the output with the correct answer count of a state-of-the-art (SOTA) model, Retro-Reader [52].

The results presented in Table 15 indicate that the ICL technique can enhance the performance of ChatGPT in answering unanswerable questions, increasing the num-

	Answer include *	**Manually**
Original	-	14/50
ICL	20/50	24/50
SOTA	-	45/50

Table 15: ICL experiment result. "Manually" shows the number of responses that ChatGPT is able to recognize the question as unanswerable.

ber of correct responses from 14 out of 50 to 24 out of 50. While ICL's performance is not as strong as that of the current state-of-the-art (SOTA) model, it demonstrates promising potential for improving ChatGPT's QA capabilities.

3.2.2 Prompt Design for Historical Bias

During our experiments on QA datasets, we discovered that chat histories can impact the responses generated by ChatGPT. This means that it may use the chat history to generate an answer, leading to potential bias. We plan to conduct further research on this topic. To better understand the extent of this influence, we designed an experiment. We started with a question that ChatGPT could generate either a correct or incorrect answer for. We then let it generate two chat histories - one where it answered correctly and another where it answered incorrectly - which we refer to as the "Correct Case" and "Wrong Case". We asked other questions after each case and collected the responses. To ensure consistency, we used the "Edit" option to ask different questions in the same position next to the "Correct Case" and "Wrong Case". The example in Figure 17 illustrates this process.

We selected the gsm8k dataset [53] which contains 8.5K high quality, linguistically diverse grade school math word problems, most of which are multi-step mathematical reasoning questions created by humans. The results of our experiment showed that even the largest transformer models struggled to achieve high test performance. Since the questions in this dataset require multiple calculations to solve and have single number answers, it was easy to evaluate the model's performance.

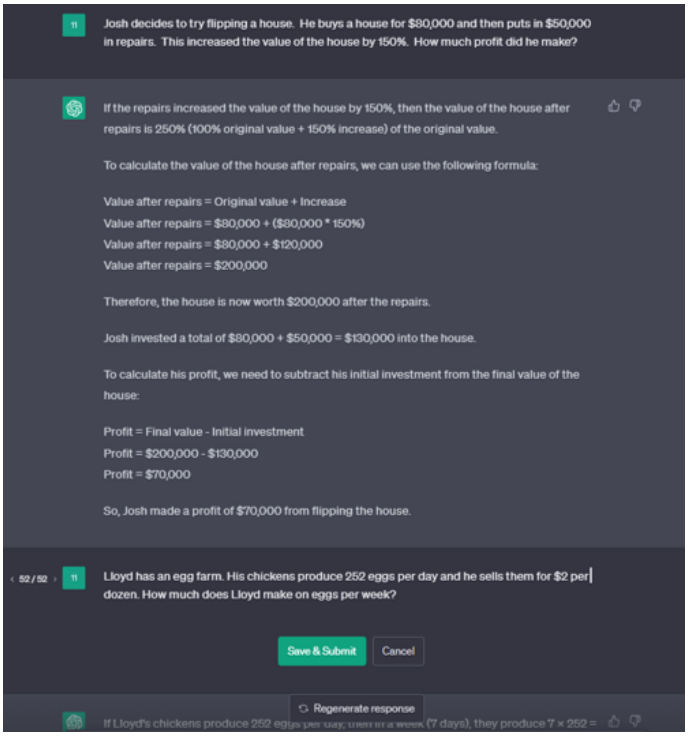

Figure 17: Edit the question instead of making a new input, so that all questions are regarded as the next question to the cast.

To identify instances where ChatGPT may output correct or incorrect answers, we generated a "Correct Case" and a "Wrong Case" based on these questions. We then randomly selected 100 other questions and asked them next to both cases using identical question sets to avoid bias. The results showed a significant difference in correctness: questions asked after the "Wrong Case" were more likely to be answered incorrectly, as shown in Table 16. This suggests that when ChatGPT generates a wrong solution to a mathematical question, it affects its mathematical reasoning ability and increases the likelihood of generating wrong solutions to other questions.

History	**accuracy**
Correct Case	88%
Wrong Case	68%

Table 16: The statistics of accuracy after the history of a "Correct Case" and a "Wrong Case".(tested 100 samples for each)

After discovering this phenomenon, we sought to find a way to prevent the reduction

in accuracy caused by history. We designed an experiment in which we added a prompt that said "Ignore the dialogue history and answer the question:" and then asked 100 questions after the "Wrong Case" again. Examples of this can be seen in Figure 18 and 19. This method was effective, as many cases where ChatGPT had previously given incorrect answers became correct again. The result shows that 18 of the 32 samples in the "wrong case" become correct if ignore the "Wrong Case" history. The rate of correct samples when ignoring the "Wrong Case" history was similar to that after the "Correct Case".

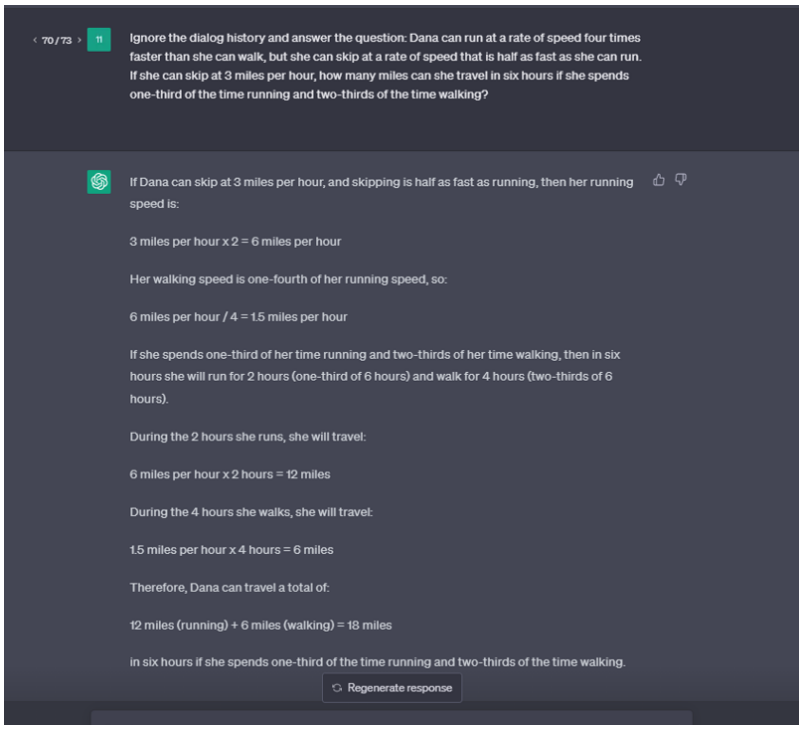

Figure 18: ChatGPT output a wrong answer after the "Wrong Case"

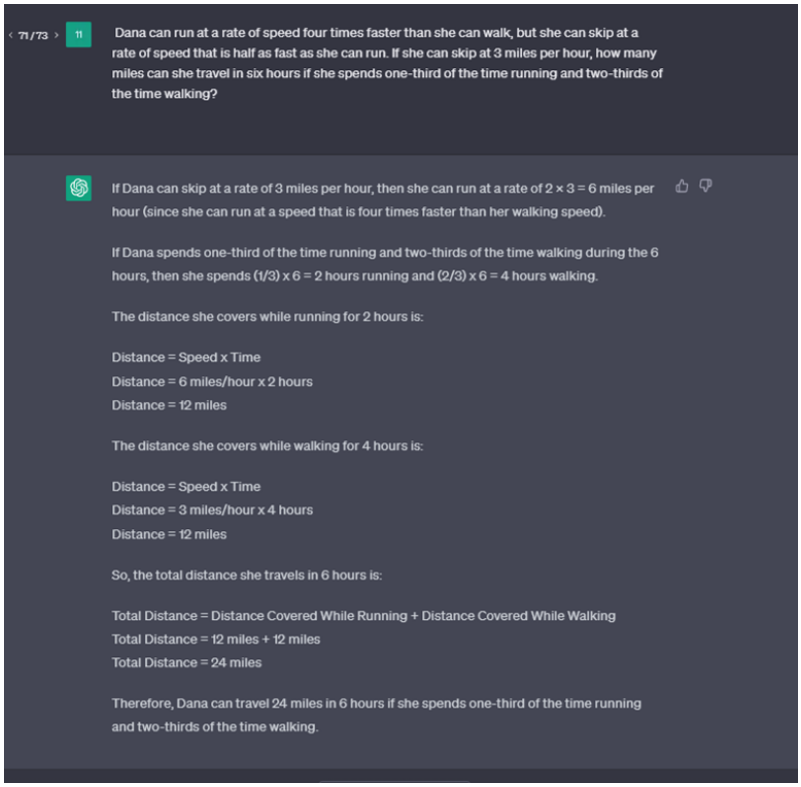

Figure 19: If ignore the history, the answer becomes correct

4 Individual Contribution

4.1 Chau Tak Ho

4.1.1 Term 1

In the first term, our primary objective was to develop a chatbot that would be proficient in question-answering (QA) using professional expertise in the background. Our first attempt is to use knowledge-based question-answering (KBQA) model as our base model. This stage of our study involved two sub-parts: relation extraction and KBQA. As a team member, my responsibility was to explore and run renowned KBQA models. I successfully accomplished my task and ran several models, including the RnG-KBQA [28], one of the state-of-the-art models on the GrailQA dataset [54] that emphasizes the generalization of KBQA models

However, some drawbacks of KBQA are found. As a result, we redirected our focus toward extractive QA. We began by using a RoBERTa-based model[1] [35] as our base model. In this phase of our project, my responsibility was to fine-tune the base extractive QA model, enabling it to leverage professional knowledge and improve its performance when handling relevant questions. The base model is developed by Haystack and it provides a toolkit for users to construct custom datasets to train the model. This approach allowed us to input a document into the Haystack's annotation tool, create customized questions related to the document, and highlight a particular section of the document as the answer. The annotated document and question could then be exported in the SQuAD [37] dataset format. Using this tool, I created over 60 question-answer pairs, and the model demonstrated a slight improvement in answering questions related to a specific professional field.

[1] https://huggingface.co/deepset/roberta-base-squad2

4.1.2 Term 2

In the second term, our main focus is to assess ChatGPT's ability to answer questions accurately. My responsibilities include testing two benchmarks, namely SQuAD [37] and SuperGLUE [45], which include consisting of four datasets - SQuAD2.0, BoolQ, MultiRC, and ReCoRD. Additionally, I am responsible for evaluating the performance of these datasets. As for the methodology improvement, my tasks involve designing an in-context learning (ICL) methodology, conducting experiments to prove the effectiveness of ICL, and evaluating the results obtained from this methodology.

4.2 Wang Weixiao

4.2.1 Term 1

In the first term, we aimed to build a professional chatbot that could answer questions based on specific field knowledge, instead of just casual chatting. To achieve this, I read various articles to develop the methods and structure of the chatbot. I built the basic structure by modifying public models and packages and fine-tuned a RoBERTa-based model[35] to accomplish basic QA tasks. However, we found that it was weak in handling complex questions that needed multi-hop reasoning. To overcome this, I utilized decompRC[38] to break down the questions into simpler, single-hop sub-questions and developed a decompose-and-answer system. We also designed a simple UI and tested its accuracy on both single-hop and multi-hop questions.

4.2.2 Term 2

In the second term, our focus was on learning and improving current LLMs. We tested ChatGPT on various datasets and analyzed the failed cases, which led us to identify historical bias as one of the issues. We prepared a pre-print paper that presented our results and analysis of ChatGPT's performance on different QA tasks. Our research also included an experiment that demonstrated how chat histories could affect ChatGPT's responses. We found that answering questions after incorrect histories resulted in lower accuracy, suggesting a significant impact of history on ChatGPT's performance. To mitigate this bias, we experimented with using prompts to ignore incorrect history. We plan to conduct further research to develop a more comprehensive understanding of historical bias.

5 Conclusion

Over the course of two semesters, we conducted a comprehensive study on chatbots. Firstly, we learned essential theories, including NLP, QA, and knowledge graphs, to build a chatbot ourselves. Our chatbot is capable of handling conversations on professional topics by learning from provided documents. It understands specific field terminology and the relationships between them, and can answer complex questions using a divide-and-conquer approach.

We also studied existing chatbots and evaluated their performance to identify areas for improvement. Our focus was on ChatGPT, a large language model that has shown impressive performance on various tasks. We tested its ability on eight datasets from five QA benchmarks and analyzed the results. While ChatGPT performed well on some tasks, there were still some weaknesses that we tried to address by studying in-context learning and historical bias. Our research found that appropriate examples and prompts can significantly improve its performance.

Overall, our study provided a comprehensive understanding of chatbot theories, building a chatbot from scratch, and improving current chatbots. We have created several pre-print papers and plan to continue our research in this field.

References

[1] G. Caldarini, S. Jaf, and K. McGarry, "A literature survey of recent advances in chatbots," *Information*, vol. 13, no. 1, p. 41, 2022.

[2] P. Suta, X. Lan, B. Wu, P. Mongkolnam, and J. Chan, "An overview of machine learning in chatbots," *Int J Mech Engineer Robotics Res*, vol. 9, no. 4, pp. 502–510, 2020.

[3] S. Badugu and R. Manivannan, "A study on different closed domain question answering approaches," *International Journal of Speech Technology*, vol. 23, pp. 315–325, 2020.

[4] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," 2022.

[5] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021.

[6] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds Mach.*, vol. 30, p. 681–694, dec 2020.

[7] D. Su, Y. Xu, G. I. Winata, P. Xu, H. Kim, Z. Liu, and P. Fung, "Generalizing question answering system with pre-trained language model fine-tuning," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, (Hong Kong, China), pp. 203–211, Association for Computational Linguistics, Nov. 2019.

[8] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "Qa-gnn: Reasoning with language models and knowledge graphs for question answering," 2021.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[10] W. Jiao, W. Wang, J. tse Huang, X. Wang, and Z. Tu, "Is chatgpt a good translator? a preliminary study," *ArXiv*, vol. abs/2301.08745, 2023.

[11] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. C. Petersen, A. Chevalier, and J. J. Berner, "Mathematical capabilities of chatgpt," *ArXiv*, vol. abs/2301.13867, 2023.

[12] X. Yang, Y. Li, X. Zhang, H. Chen, and W. Cheng, "Exploring the limits of chatgpt for query or aspect-based text summarization," 2023.

[13] C. Xia and L. Zhang, "Conversational automated program repair," *ArXiv*, vol. abs/2301.13246, 2023.

[14] A. Borji, "A categorical archive of chatgpt failures," *ArXiv*, vol. abs/2302.03494, 2023.

[15] T. Y. Zhuo, Y. Huang, C. Chen, and Z. Xing, "Exploring ai ethics of chatgpt: A diagnostic analysis," *ArXiv*, vol. abs/2301.12867, 2023.

[16] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[17] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2021.

[18] L. Ehrlinger and W. Wöß, "Towards a definition of knowledge graphs.," *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1-4, p. 2, 2016.

[19] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[20] A. Clementeena and P. Sripriya, "A literature survey on question answering system in natural language processing," *International Journal of Engineering and Technology(UAE)*, vol. 7, pp. 452–455, 06 2018.

[21] S. Lende and M. Raghuwanshi, "Closed domain question answering system using nlp techniques," *IJESRT*, 01 2016.

[22] M. Aydar, O. BOZAL, and F. ÖZBAY, "Neural relation extraction: a review," *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, vol. 29, pp. 1029–1043, 03 2021.

[23] P.-L. Huguet Cabot and R. Navigli, "REBEL: Relation extraction by end-to-end language generation," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, (Punta Cana, Dominican Republic), pp. 2370–2381, Association for Computational Linguistics, November 2021.

[24] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J.-R. Wen, "Complex knowledge base question answering: A survey," *arXiv preprint arXiv:2108.06688*, 2021.

[25] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

[26] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale,

multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[27] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[28] X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong, "Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering," *arXiv preprint arXiv:2109.08678*, 2021.

[29] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond iid: three levels of generalization for question answering on knowledge bases," in *Proceedings of the Web Conference 2021*, pp. 3477–3488, 2021.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[31] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer.," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[32] M. Luo, K. Hashimoto, S. Yavuz, Z. Liu, C. Baral, and Y. Zhou, "Choose your qa model wisely: A systematic study of generative and extractive readers for question answering," *arXiv preprint arXiv:2203.07522*, 2022.

[33] A. Fisch, A. Talmor, R. Jia, M. Seo, E. Choi, and D. Chen, "MRQA 2019 shared task: Evaluating generalization in reading comprehension," in *Proceedings of*

the 2nd Workshop on Machine Reading for Question Answering, (Hong Kong, China), pp. 1–13, Association for Computational Linguistics, November 2019.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pre-training approach," 2019.

[36] B. Chan, T. Soni, M. Pietsch, and T. Möller, "Roberta-base-squad2," 2020.

[37] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Melbourne, Australia), pp. 784–789, Association for Computational Linguistics, July 2018.

[38] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, "Multi-hop reading comprehension through question decomposition and rescoring," *arXiv preprint arXiv:1906.02916*, 2019.

[39] "Freebase api (deprecated)," 2022.

[40] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, "Multi-hop reading comprehension through question decomposition and rescoring," in *ACL*, 2019.

[41] M. Pietsch, T. Soni, B. Chan, T. Möller, and B. Kostić, "Haystack," 2019.

[42] Wikipedia contributors, "Natural language processing, wikipedia," 2022. [Online; accessed 29-November-2022].

[43] Wikipedia contributors, "Machine learning, wikipedia," 2022. [Online; accessed 29-November-2022].

[44] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 2369–2380, Association for Computational Linguistics, Oct.-Nov. 2018.

[45] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," 2019.

[46] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "BoolQ: Exploring the surprising difficulty of natural yes/no questions," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 2924–2936, Association for Computational Linguistics, June 2019.

[47] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 252–262, Association for Computational Linguistics, June 2018.

[48] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. Van Durme, "Record: Bridging

the gap between human and machine commonsense reading comprehension," 2018.

[49] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "CommonsenseQA: A question answering challenge targeting commonsense knowledge," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4149–4158, Association for Computational Linguistics, June 2019.

[50] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Dublin, Ireland), pp. 3214–3252, Association for Computational Linguistics, May 2022.

[51] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li, and Z. Sui, "A survey on in-context learning," 2023.

[52] Z. Zhang, J. Yang, and H. Zhao, "Retrospective reader for machine reading comprehension," 2020.

[53] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," 2021.

[54] Y. Gu, S. Kase, M. Vanni, B. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases," in *Proceedings of the Web Conference 2021*, ACM, apr 2021.