Problem Chosen	2024 MCM/ICM Summary Sheet	Team Control Number
Project one		1

Analysis and Research on Training Methods Based on LLM Reading Literature

Summary

This study will explore the use of some feasible large language models like Kimi, Chatgpt, Claude(We just focusing on chatgpt) to train in different way and find a better training method to improve the model's ability to read and analyze scientific literature, so that its answers better meet user requirements. Firstly, we will introduce some basic operating principles of LLM in text processing, and then develop relevant evaluation models to provide standards for the results. Moreover, we will investigate the impact of different questioning training methods to enable the model to read the literature provided by users, and evaluate the training results based on the established standards. Ultimately, we anticipate that our research will not only contribute to advancing the capabilities of large language models in scientific literature comprehension but also facilitate the development of more intelligent and user-centric information retrieval systems in the domain of academic research and knowledge dissemination.

Keywords: Large Language Model (LLM), Literature reading, Model training

Contents

1 Restatement of the problem

Scientific literature reading is an important part of scientific research work, but it can be extremely tedious and troublesome to read through dozens or even hundreds of pages of literature. Since the advent of interactive artificial intelligence models like ChatGPT, this time-consuming task can be assisted by them. The advantage of using these tools is the ability to quickly summarize the main content of the literature, allowing researchers to achieve a preliminary understanding. However, the disadvantage is that the answers provided are uneven, and heavily **influenced by how users train them through questioning**. Therefore, we hope to formulate criteria for judging answers, design multiple questioning training methods, test and evaluate different approaches, in order to find more effective questioning training directions.

2 Analysis of the Problem

First, let's discuss the basic working principle of the large language model and its related word segmentation operations.

Large language models primarily operate based on deep neural networks and extensive training on massive text data, enabling them to comprehend and generate text similar to humans. These models typically employ unsupervised learning techniques, training on large datasets without specific labels or targets. The goal is to learn the underlying structure of the data and use it to generate new data similar to the original. When generating text, the model begins with a given sequence of words and then predicts the next word in the sequence based on the probability of words in the training corpus, repeating this process until the desired text length is reached. Large language models often utilize the Transformer architecture, enabling the model to process entire text sequences in parallel. By introducing an "attention mechanism," the model can track the relationships between words in the text sequence both forward and backward. The encoder part employs self-attention mechanisms to focus on the current word and its context, while the decoder part is responsible for generating text based on the encoder's output. Next, let's provide a more detailed introduction to the implementation and principles of LLM.

When a user sends a piece of text to the model, the computer cannot directly read and understand its content. The model first needs to perform segmentation on the user's query text, breaking it down into the smallest basic units – words. Common segmentation models encompass both traditional classical models and widely used deep learning models, each with its own characteristics and applicable scenarios. Traditional classical models include rule-based approaches such as Forward Maximum Matching, Reverse Maximum Matching, and Bidirectional Maximum Matching, as well as statistical models like Hidden Markov Models (HMM) and Conditional Random Fields (CRF). These models rely on rules, heuristics, or statistical methods to segment text and are suitable for simple tasks or moderate-sized annotated datasets. Deep learning models encompass recurrent neural network (RNN)-based models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs), as well as Transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) and its deriva-

tives. These models leverage neural network architectures to capture complex dependencies in sequential data and learn rich representations of language. They excel on large-scale datasets and complex tasks due to their strong generalization capabilities and language understanding abilities. They begin by converting words or tokens into dense vector representations known as word embeddings or token embeddings. These embeddings capture semantic and syntactic similarities between words, enhancing the model's ability to understand language. These models employ transformer architectures, which are neural networks designed for sequential data processing. Transformers utilize self-attention mechanisms to weigh the importance of different words in a sequence, enabling the model to grasp long-range dependencies and contextual information effectively. Large language models are trained on extensive datasets using unsupervised or semi-supervised learning techniques. They analyze vast amounts of text data to learn statistical patterns and language structures. Through this training, the model develops the ability to generate coherent and contextually relevant text. Additionally, these models undergo fine-tuning for specific tasks by training on task-specific datasets. This process adapts the pre-trained model to the nuances of the target task, leveraging the knowledge and representations learned during the initial training phase. In summary, large language models leverage word embeddings, transformer architectures, extensive data training, and task-specific fine-tuning to achieve their impressive language understanding and generation capabilities.

3 Performance Judgement of AI

3.1 Model Assumptions

1. User feedback can affect the quality of responses in large language models.

   For example, if users encourage the LLM, it will more likely to answer in the same pattern, while users criticism will make LLM avoid to answer so.

3.2 Notations of our model

Abbreviation	Description
PF	Performance of AI
AKI	Accuracy of key information extraction
ALE	Accuracy of language expression scoring
CI	Completeness of information scoring
TR	Topic relevance score

Table 1: Factors' Abbreviations Used in the Formula

3.3 Preliminary Formula of Performance

Here, we set up the formula of judging the performance.

$$PF = 0.2AKI + 0.1ALE + 0.2CI + 0.5TR \tag{1}$$

Explanation

Here we introduce four different estimating index. We then synthesized the evaluation model by weighting the four judgment indices. In order to simplify the processing of data, we chose linear model to achieve our purpose. Additionally, The coefficients were set not only by reading some articles[1] but also by our experience. And eventually we decided to make the **Topic relevance score** to be predominant. Because Cosine similarity is calculated based on TF-IDF, which reduces the effect of article length, and the other terms are used as auxiliary judgments which are strongly influenced by length of the article.

3.4 Methods of Calculating the Factors

3.4.1 Accuracy of key information extraction

3.4.2 Accuracy of language expression scoring

BLEU score (Bilingual Evaluation Understudy Score) is a widely used metric for evaluating the performance of machine translation systems. The name "Bilingual Evaluation Understudy" suggests that it is designed as an auxiliary tool for bilingual evaluation. BLEU score assesses translation quality by measuring the degree of overlap between the output of a machine translation system and a set of reference translations in terms of n-grams.

The principle behind BLEU score is based on the assumption that if the output of a machine translation system exhibits a high degree of overlap with reference translations in terms of n-grams, then it is more likely to be a correct translation. Here, an n-gram refers to a sequence of n consecutive words or characters. BLEU score calculates the frequency of n-grams in the machine translation output that appear in the reference translations.

3.4.3 Completeness of information scoring

Jaccard distance is a measure of dissimilarity between two sets, defined as the complement of the Jaccard similarity coefficient. The Jaccard similarity coefficient is a statistic used for comparing the similarity and diversity of two finite sample sets. It is calculated as the size of the intersection of the two sets divided by the size of their union. Conversely, the Jaccard distance is the ratio of the size of the symmetric difference of the two sets to the size of their union. A higher Jaccard distance indicates a greater dissimilarity between the two sets, while a lower value suggests a higher degree of similarity. This metric is often used in areas such as bioinformatics, text mining, and image analysis to measure the dissimilarity between different sets of data.

3.4.4 Topic relevance score

Cosine similarity

Cosine similarity is an assessment of how similar two vectors are by calculating the cosine of their angle. Cosine similarity plots vectors, based on their coordinate values, into a vector space, such as the most common two-dimensional space.

Cosine similarity measures the similarity between two vectors by measuring the cosine of the angle between them.The cosine of a 0-degree angle has a value of 1, whereas the cosine of any other angle has a value no greater than 1; and its minimum value is -1. Thus, the cosine of an angle between two vectors determines whether the two vectors are pointing roughly in the same direction. The cosine similarity is usually used in positive space, and therefore gives a value between -1 and 1.

Note that this upper and lower bounds hold for any dimension in vector space, and that cosine similarity is most commonly used in high-dimensional positive space. In information retrieval, for example, each lexical item is given a different dimension, and a dimension is represented by a vector whose values on the various dimensions correspond to the frequency with which the lexical item appears in the document. Cosine similarity thus gives the similarity of two documents in terms of their subject matter.

3.5 Operate Process

We try to analyze the accuracy of Chatgpt's grasp of article content under two different conditions: supervised training without feedback and supervised training with feedback. And throughout this study, we have been using chatgpt 4.0.

3.5.1 With Feedback

When we utilizing this training choice, we set up a specific mission for the model in the very begining.

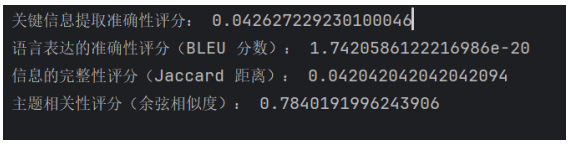
Then enter the following content. "Next, I will conduct 10 rounds of training on reading scientific literature for you. Firstly, I will provide you with a document that requires you to read the content and summarize its main content. At the end of each round, I will provide you with feedback on the accuracy score of key information extraction, accuracy score of language expression (BLEU score), completeness score of information (Jaccard distance), and topic relevance score (cosine similarity). You should try to improve these scores as much as possible during the training process."

Then after each round of training, we gave feedback on the ratings and asked for improvements. we repeated the process for nine times and clutch the value of each round. On the last occasion, we throw the model the first round of the paper document sent for testing, get the tenth round of scores, and then compare them with the first round to finalize the training results.

In our code, Our first input is the **full-text** pdf file of the first article we are looking at. And then we followed the different sections, such as summary and introduction to input the content one by one. Eventually, through the calculation of our method, we got a total of four kinds of data.

Here, we display partial round of data by showing it in the figures.

(a) Round1



(b) Round3

(c) Round5

(d) Round9

Figure 1: Examples of with feedback

3.5.2 Without feedback

Under this method, we didn't need to offer feedback to the AI. We just followed the basic step in the former method. In the first nine rounds, we sent different nine thesis documents to test, and in the last round we sent the documents from the first round to compare the results of the tenth round with the first round of testing. And the some of the data was exhibited below:

(a) Round1

(b) Round4

(c) Round7

(d) Round10

Figure 2: Examples of without feedback

3.6 Model Refinement about the formula

Through our calculation, we can find the value of Accuracy of language expression scoring is too weak to be taken into consideration. Thus we modified the expression of our formula.

$$PF = 0.2AKI + 0.2CI + 0.6TR \tag{2}$$

For the reason of reducing the effect of article length, we decided to continue to amplify the weight of topic relevance score. Thus we changed the coefficient from 0.5 to 0.6.

3.7 Result

For each method, we first compared the judging score of tenth round with the judging score of first round and calculated the spectrum of increment. Eventually, we recorded the result in table 2 below.

Choice	spectrum of increment(%)
Training with feedback	5.59
Training without feedback	3.54

Table 2: Result of increment

To visualize the change of each round and the difference between two choices, we generated figure 3 below to display the result.

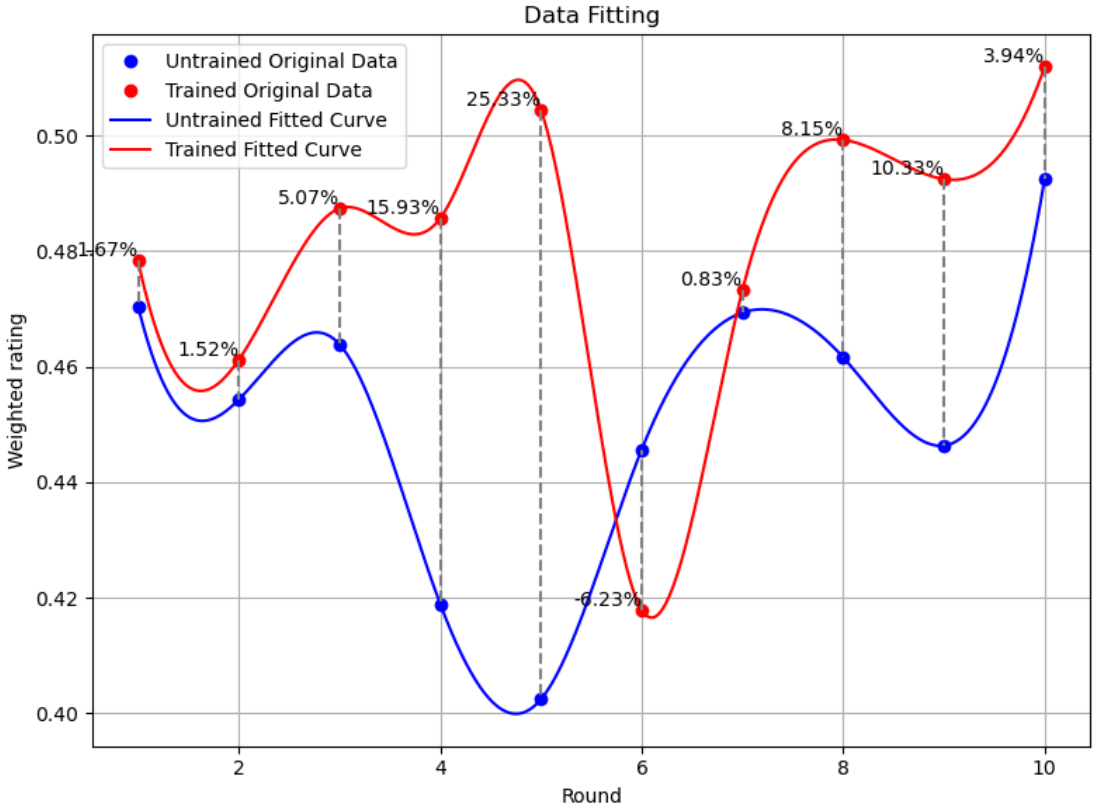


Figure 3: Changing figure

From the above figure, it is clear that the red line almost always higher than the blue line. Therefore, we can extrapolate that training own some positive influence on the performance of Chatgpt.

4 Limitation and future discussion

In our study, we just use Chatgpt 4 as our researching subject, so it cannot represent all the Large Language Model. Thus, in the future, our team will continue to examine other different large language model to have better understanding the effect of training with feedback.

References

[1] X. Zhang, S. Dadkhah, A. G. Weismann, M. A. Kanaani, and A. A. Ghorbani, "Multimodal fake news analysis based on image–text similarity," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 1, pp. 959–972, 2024. DOI: 10.1109/TCSS.2023.3244068.

Appendices

Appendix A First appendix

Python source code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import SmoothingFunction
from nltk.metrics import jaccard_distance
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import PyPDF2


nltk.download('punkt')


def extract_text_from_pdf(pdf_path):
    text = ""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfFileReader(file)
        for page_num in range(reader.numPages):
            page = reader.getPage(page_num)
            text += page.extractText()
    return text
```

```
def accuracy_score(summary, reference):

    summary_tokens = set(word_tokenize(summary))
    reference_tokens = set(word_tokenize(reference))

    intersection =
       len(summary_tokens.intersection(reference_tokens))

    total_words = len(reference_tokens)

    accuracy = intersection / total_words if total_words > 0
       else 0
    return accuracy
```

```
    def fluency_score(summary, reference):

```

```
    summary_tokens = word_tokenize(summary)
    reference_tokens = word_tokenize(reference)

    smoothing = SmoothingFunction().method1
    bleu_score = sentence_bleu([reference_tokens],
       summary_tokens, smoothing_function=smoothing)
    return bleu_score


    def completeness_score(summary, reference):

    summary_tokens = set(word_tokenize(summary))
    reference_tokens = set(word_tokenize(reference))

    completeness = 1 - jaccard_distance(summary_tokens,
       reference_tokens)
    return completeness
```

```
    def relevance_score(summary, reference):

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([summary,
       reference])
    similarity_matrix = cosine_similarity(tfidf_matrix)
    relevance = similarity_matrix[0, 1]
    return relevance
```

```
if __name__ == "__main__":
    summary = """Text"""
    pdf_reference_path = r'C:\Users\***\Desktop\X.pdf' (X from
       1-9)
    reference = extract_text_from_pdf(pdf_reference_path)
    accuracy = accuracy_score(summary, reference)
    fluency = fluency_score(summary, reference)
    completeness = completeness_score(summary, reference)
    relevance = relevance_score(summary, reference)
    print("Accuracy of key information extraction:", accuracy)
    print("Accuracy of language expression scoring(BLEU
       fraction):", fluency)
    print("Completeness of information scoring(Jaccard
       distance):", completeness)
    print("Topic relevance score(cosine similarity):", relevance)
```