Extracting Tuple-Based Service Demands with Large Language Models for Automated Service Composition

Chih-Jung Hsu
Institute of Information Science
Academia Sinica
Taipei, Taiwan
hsu8591@iis.sinica.edu.tw

Yu-Xiang Luo
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
b10902037@csie.ntu.edu.tw

Yi-Chen Mao
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
b10902103@csie.ntu.edu.tw

Chien-Min Wang
Institute of Information Science
Academia Sinica
Taipei, Taiwan
cmwang@iis.sinica.edu.tw

Yang Syu
Department of Information Science
National Taipei University of Education
Taipei, Taiwan
yangsyu@mail.ntue.edu.tw

***Abstract*—Existing Automated Service Composition (ASC) approaches typically require inputs to be in a designated form. These, namely tuples, pose challenges due to the significant divergence from the most commonly used and straightforward formats for expressing software requirements. In our previous work, we developed a rule-based approach that necessitated substantial resources for analyzing the content of requirements and establishing appropriate rules. Given the recent successes in field research involving large language models (LLM)—where significant achievements have been made in real-time automatic text generation tasks—we propose leveraging LLM for ASC to extract critical tuple-based information. We have created a new dataset to simulate everyday service demands and have established clear guidelines regarding service demand types (e.g., input and output). Moreover, we have designed a suitable workflow that optimizes LLM performance. Our experiments and results demonstrate that our proposed LLM-based approach not only achieves extraordinary performance and reliability at a lower cost but also outperforms the complex rule-based solutions that were previously employed.**

***Index Terms*—Automatic Service Composition, Service Description, Large Language Model, Natural Language Processing, Named Entity Recognition**

I. Introduction

Automated Service Composition (ASC) has been a widely studied research subject in academia for many years [1]–[3]. Research on ASC [2] focus on the complete generation of a composite service from scratch to meet user requests or customer requirements. One of the most critical issues concerning the efficient connection between ASC and software requirements is extracting accurate information from manually created documents without a unified format. Moreover, most of these documents are written in natural language, expressing human thoughts, making it challenging to utilize this unprocessed information. As a result, our goal is to design a precise and flexible automatic model capable of transforming human-written requirement descriptions into a unified tuple-based format.

In our previous work [4] [2], we provided a rule-based solution. Through the analysis of requirement descriptions, we observed specific rules related to the positioning of words. We discovered the relationship between keywords and target words using POS tags. Finally, we developed appropriate rules to extract keywords for different classifications. However, analyzing documents for rule extraction consumes considerable resources and lacks flexibility due to the need for repeating observations on different datasets. Therefore, we propose an efficient and flexible approach in this paper.

We employ Large Language Models (LLM) to recognize requirement descriptions and extract critical partitions. In recent years, LLM has demonstrated significant performance in text generation; by inputting relevant prompt information, users can receive beneficial responses from the model's output. We analyze the features of mainstream LLM and adjust our design to meet our objectives. In the following section, we introduce our approaches, including dataset construction, models, prompt design, fine-tuning, and experimental for implementing the extraction of tuple-based service demands through LLM.

With astounding experimental results, we demonstrate that the LLM-based methods we propose can address the challenge of establishing a standard format for human-written service descriptions. These approaches surpass our previous work in sentence accuracy and enable the rapid construction of prompts that are more understandable than those from our

This work was partially supported by the National Science and Technology Council, Taiwan, under Grant No. NSTC112-2221-E-001-008.

prior rule-based design. Furthermore, we have successfully implemented a partition of LLM on our machine, achieving performance that exceeds our previous efforts and approaches that of more resource-intensive LLM. Consequently, we confirm the approach can be applied in various hardware environments, even with limited resources.

The rest of the paper is organized as follows: In Section II, we define the problem addressed in the paper. Section III reviews the related work in the literature. Section IV explains our proposed approaches in detail. Section V presents our experimental results and includes a discussion of these findings. Finally, we conclude our work in Section VI.

II. Related Work

The primary concern and target of most surveyed ASC studies is to fulfill user requests or service requirements. User requests, expressed in natural language, typically do not conform to a specific format. Therefore, the most critical task is converting these requests into uniform service requirements. A service requirement is composed of two parts: the non-functional requirements and the functional expressions. Y. Fanjiang et al. [1] concluded that most existing ASC approaches and studies assume tuple-based service demands as the working input for subsequent service composition procedures. A generic tuple-based service description model (template) as defined in [1] is as follows:

$$<\text{I, O, P, E, NF, ON}>$$

where I, O, P, E, and NF represent the input, output, precondition, effect (postcondition), and nonfunctional property sets of a described service, respectively, and ON connects to a knowledge ontology that formally and semantically defines the elements contained in the other tuples [1].

The goal of composing disparate service requirements into a specific tuple-based format introduces several research challenges. First, it necessitates the identification of clear requirement types from a software engineering perspective. Second, these approaches must accurately recognize the textual requirements. Finally, these methods can efficiently extract target components from textual content. This section provides an overview of relevant studies in these research domains.

Rule-based knowledge extraction is viable as it only requires defining rules concerning the relationship between manual documents and tuple-based requirements. In our previous work [4] [2], through the analysis of requirement descriptions, we identified specific rules related to word positioning. Additionally, we employed the NLP tool, Stanford CoreNLP [5], to parse and tag our dataset, which consists of sentences collected from the Leetcode website. With this tool, we developed a part-of-speech (POS) tagger, software that reads text in a language and assigns parts of speech to each word (and other tokens), such as nouns, verbs, adjectives, etc. Achieving a unified format necessitates the extraction of target words from requirement descriptions. We identified an essential set that typically precedes the target words. Subsequently, we discovered the relationship between this crucial set and the target words using POS tags. Moreover, we utilized dependency parsing [6] to establish relationships between "head" words and words modifying those heads, analyzing the grammatical structure. This structure enables us to design reasonable rules for complex cases, such as sentences not in critical sets. As a result, we developed appropriate rules to extract target words for different classifications.

ML-based knowledge extraction is favored for its potential to achieve higher performance than traditional methods. Miwa et al. [7] proposed using LSTM-RNNs, enabling the model to perform entity and relation extraction simultaneously. Zhang et al. designed a holon ontology structure and chose RoBERTa-base-SQuAD2[3] [3], a pre-trained model utilizing the Robustly Optimized BERT Approach (RoBERTa) [8]. Using their chosen model, they converted raw data into a Holon description in JSON format. Subsequently, they converted this description into a Holon class. Finally, they obtained unified format information that their IoT device could easily recognize.

III. Problem

This section defines the investigated service composition problem and its motivation. ASC research necessitates significant resources for formalizing requirement texts, mainly since software engineering professionals are tasked with analyzing human-expressed content and transforming it into high-quality system design specifications. In this study, we utilize LeetCode [9], which offers numerous functional descriptions. Each description suggests the *Input* and *Output* of a desired software component, closely mirroring service requirements and serving as our primary source for natural language-based service demands. We provide one clear illustration of this with the following textual description example, collected in our new dataset (discussed in Subsection IV-A):

> *A problem from LeetCode is presented as follows: "You are given a string s and an integer k. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most k times. Return the length of the longest substring containing the same letter you can get after performing the above operations."*

Expanding on service requirements, we convert the above textual description into a tuple-based format for ASC-compliant formalization. As in our previous work [4], to facilitate performance comparison with standards, the remainder of this paper will focus on and report the handling of *Input* and *Output* tuples. The following example illustrates this process:

> *The nonfunctional requirement with ASC-compliant formalization:*
> *<I = (a string s; an integer k,*
> *O = (the length of the longest substring) >*

Based on the above discussion, a crucial step towards achieving ASC is the transformation of textual expressions by humans into a specific, uniform format. Approaches that enable precise extraction can effectively address this challenge. Thus, we can define and formally formulate the problem as follows:

$LLM(\bar{X} \rightarrow \bar{Y}) \rightarrow F, and \quad then \quad F : X \rightarrow Y$ Where LLM denotes an LLM used in this study to train and obtain an extraction model F for ASC; X and Y represent a set of human-expressed service demands and a set of corresponding tuple-based demands for X, respectively; $\bar{X}$ and $\bar{Y}$ denote sets of training cases that simulate text for service demands for LLM's fine-tuning and their corresponding tuple-based elements, respectively, where $\bar{X} \subseteq X$ and $\bar{Y} \subseteq Y$. In essence, our goal is to find a valid function F using $LLM()$ to map $\bar{X}$ to $\bar{Y}$, aiming for high-precision extraction. This study focuses on the $LLM()$ design, selecting and comparing mainstream LLM for precise extraction.

IV. Approach

In this study, we encounter several challenges, including the scarcity of service demand data sources, model performance, and the design of specific tasks for ASC. Therefore, we will first address the collection of suitable data. Secondly, we will analyze various models to select the most appropriate one. Thirdly, we discuss how to improve performance on ASC tasks through LLM and our prompt design. The overall workflow is depicted in Figure 1.

A. Dataset Construction

1) Dataset Source Selection and Data Acquisition: In our initial research phase, we sought human-written ASC requests from the existing literature. However, we found only a limited number of sentences that require more diversity and volume for our study. The challenge consists of many datasets containing undefined inputs and outputs, complex formatting symbols, and elements not aligning with our research requirements.

To address these issues, we use LeetCode as our dataset source.LeetCode provides abundant functional descriptions, each detailing software components' input and output requirements. These descriptions closely mirror service requirements, offering us a rich, natural language-based corpus for analysis. Compared to a previous dataset we examined, which only had 170 entries and needed to be more comprehensive, we collected a new dataset of 500 functional descriptions from LeetCode.

Each data entry in our dataset consists of several key components to facilitate thorough analysis:

- **ID:** A unique identifier for each problem allows us to easily reference and organize the dataset.
- **URL:** The direct link to the problem on LeetCode provides a pathway to view the original problem context and details.
- **Sentence (Original):** The problem description as it appears on LeetCode, including HTML tags. This version preserves the original formatting and structure, which is essential for understanding the context and specifics of each problem.
- **Sentence (Processed):** The problem description after removing HTML tags. This cleaned version focuses on the textual content, making it more accessible for analysis and processing.

2) Dataset Analysis and Data Cleaning: This methodology ensures we have a comprehensive, detailed dataset for analyzing and extracting tuple-based service demands from natural language descriptions. It also sets a robust foundation for further analysis and model development.

Our dataset, derived from LeetCode's extensive collection of 3,017 programming challenges, consists of a selected subset of 500 problems. This sample was randomly selected to balance manual annotation and analytical representativeness.

After a thorough examination, we found specific categories that did not align with our research objectives. Specifically, we excluded problems labeled as 'Database' due to their unique format, which frequently incorporates non-standard descriptions and SQL table schemas, elements that cannot be directly translated into the general service requirements essential for our study. Additionally, we removed entries featuring inconsistent HTML structures, which posed challenges for accurate parsing. This selection process ensured that our dataset remained focused and relevant to our aims, enhancing the quality and applicability of our research.

The initial dataset encompassed 88.56% of LeetCode's problem diversity. After excluding problems labeled as 'Database', which accounted for 8.19% of LeetCode's problem diversity due to their unique formats, the focus was narrowed to 57 categories. This adjustment resulted in covering 96.46% of the remaining diversity of LeetCode's problems.

This adjustment ensured the integrity and consistency of our dataset, maintaining a broad spectrum of programming and algorithmic challenges while aligning more closely with our study's focus on ASC.

3) Data Labeling: A pivotal step in refining our dataset for analysis involved manually annotating the I, O tuples, representing *Input* and *Output*, for each problem description. This meticulous process was essential in transforming raw problem descriptions into structured data, facilitating the training and evaluation of our models.

We ensured high accuracy and consistency in our dataset by assigning three annotators to independently label all 500 entries with *Input* and *Output* tuples. This comprehensive approach allowed for the examination of each data point from multiple perspectives. Following the initial annotation, an expert in software engineering reviewed the labels to identify and resolve discrepancies among the annotators, thereby achieving consensus on the most accurate labels for each entry.

This rigorous and collaborative annotation process guaranteed high reliability in our dataset, which is crucial for the effectiveness of our ASC study. This detailed attention in the data labeling stage provided a strong foundation for our research, facilitating the development and assessment of our models.

This dataset collection process is a cornerstone of our study, enabling the development and evaluation of models designed to extract tuple-based service demands. It reflects a systematic approach to overcoming the challenges of dataset scarcity and diversity, laying the groundwork for future advancements in ASC.

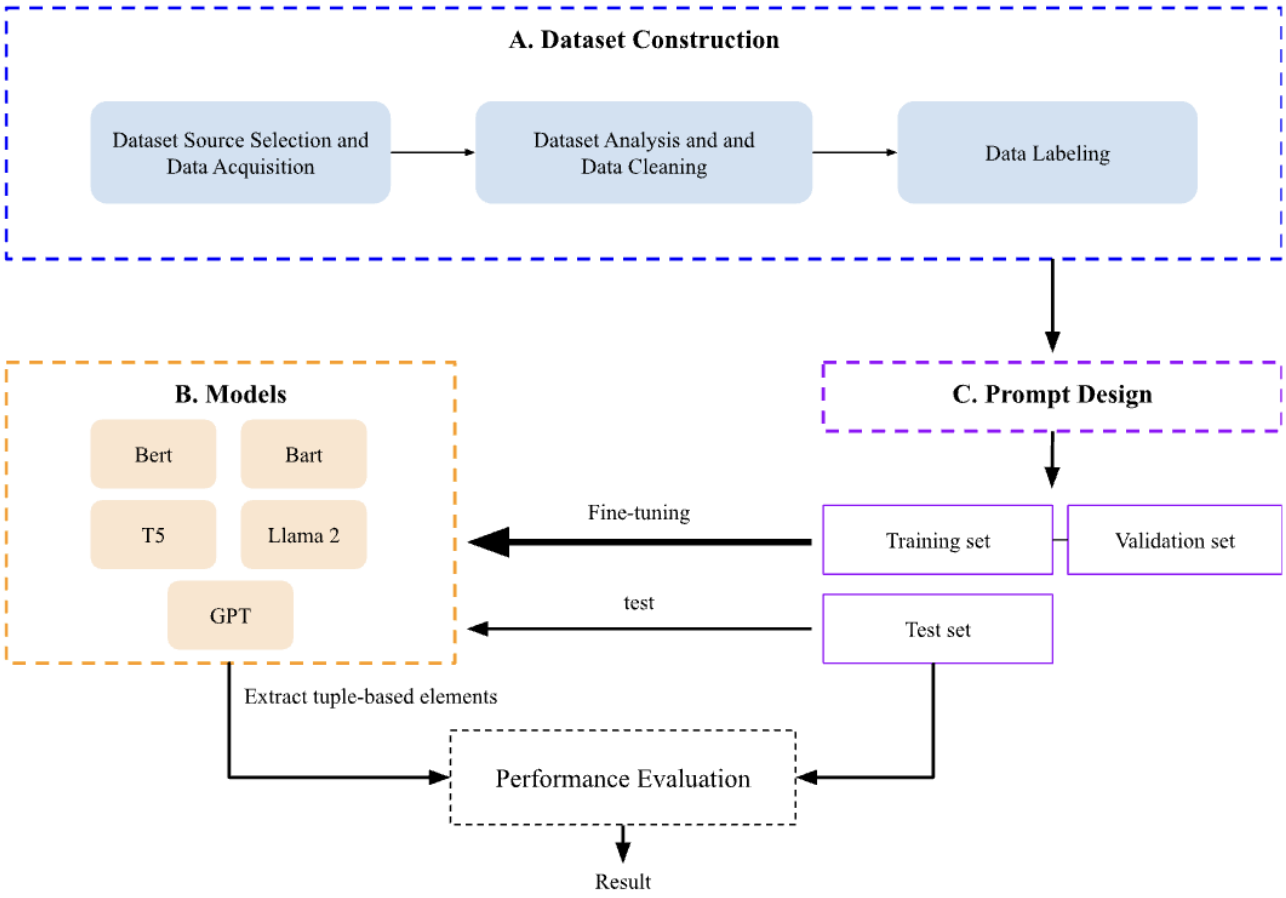

Fig. 1. Workflow of our approach

B. Models

LLM have become increasingly prevalent in recent years, with many focusing on diverse tasks. In our case, the task involves named entity recognition in a broad definition. Therefore, we selected suitable LLM from various options. We will introduce these LLM in the remaining subsection.

- Bidirectional Encoder Representations from Transformers (BERT) [10]: Developed by Google, this model introduced a revolutionary approach to NLP. BERT incorporates two key strategies: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly masks words in a sentence and trains the model to predict these masked words, thus enabling it to understand the context from both the left and right sides of a word. This training method is especially suited for our QA (Question-Answering) task. NSP further enhances BERT's understanding of the relationship between sentences. The model is pre-trained on a vast corpus, including the entire English Wikipedia and BookCorpus. BERT has become a popular benchmark in the field, serving as a foundational model for evaluating and developing NLP systems, which underscores our decision to incorporate it into our experiments.
- Bidirectional and Auto-Regressive Transformers (BART) [11]: Developed by Facebook (now Meta), this model combines the benefits of auto-encoding and auto-regressive models. It employs a standard Transformer-based neural machine translation architecture but distinguishes itself through its pre-training objective. BART is trained by corrupting text with an arbitrary noising function and then learning to reconstruct the original text. This training approach helps the model better understand sentence structure and context.
- Text-to-Text Transfer Transformer (T5) [12]: Developed by Google, this model extends the boundaries of NLP by treating all tasks as text-to-text problems. This Approach means that T5 treats both the input and output as text sequences, regardless of whether the task is translation, question answering, or classification. Its architecture leverages the Transformer model but introduces a novel pre-training strategy that combines unsupervised, supervised, and reinforcement learning techniques. T5 is pre-trained on the Colossal Clean Crawled Corpus (C4), a vast dataset compiled from the web, encompassing a wide range of language use cases.
- Llama 2 [13]: The latest model from Meta, Llama 2, marks a significant advancement in open-source AI and

dialogue systems. Building on the foundation laid by its predecessors, Llama 2 introduces key enhancements in language modeling and natural language understanding, making it particularly adept at facilitating coherent and contextually relevant interactions in conversational settings. This adaptability makes Llama 2 exceptionally suitable for applications requiring user interaction, such as chatbots and virtual assistants. It utilizes a comprehensive and diverse training dataset covering various conversational contexts. The training data for Llama 2 includes dialogues from social media, customer service exchanges, and curated conversational datasets, ensuring the model is well-versed in various human conversational patterns and languages. Enriching the dataset significantly enhances the model's capability to manage our tasks.

- Generative Pre-trained Transformer (GPT) [14], [15], [16]: A multi-layer Transformer decoder framework achieves natural solid language understanding by utilizing a single task-agnostic model through generative pre-training followed by discriminative fine-tuning. The training procedure [14] consists of two stages: the first involves learning a high-capacity language model on a large corpus of text. This is followed by a fine-tuning stage, where the model is adapted to a discriminative task using labeled data. In this study, we focus solely on the fine-tuning procedure.
  Supervised fine-tuning: we assume a labeled dataset C, where each instance consists of a sequence of input tokens, $x^1, ..., x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

$$P(y|x^1, ..., x^m) = softmax(h_l^m W_y). \quad (1)$$

  This gives us the following objective to maximize:

$$L(C) = \sum_{(x,y)} logP(y|x^1, ..., x^m). \quad (2)$$

C. Prompt Design

Creating a prompt is one of the most critical tasks in working with LLM. While simple prompts can achieve substantial results, the quality of the outcomes depends on the amount of information provided and the craftsmanship of the prompt. A prompt may include instructions or questions directed to the model and additional details such as context, inputs, or instances. By utilizing these elements, we can more effectively guide the model to improve the quality of the results.

We have designed an efficient and effective prompt by the principles outlined in [17]. The prompt is composed of three parts, as depicted in Figure 2: the first part is the task description, which includes providing reference text and clear instructions; the second part describes the context as expressed by humans; and the final part specifies the label that we aim to extract about the context.

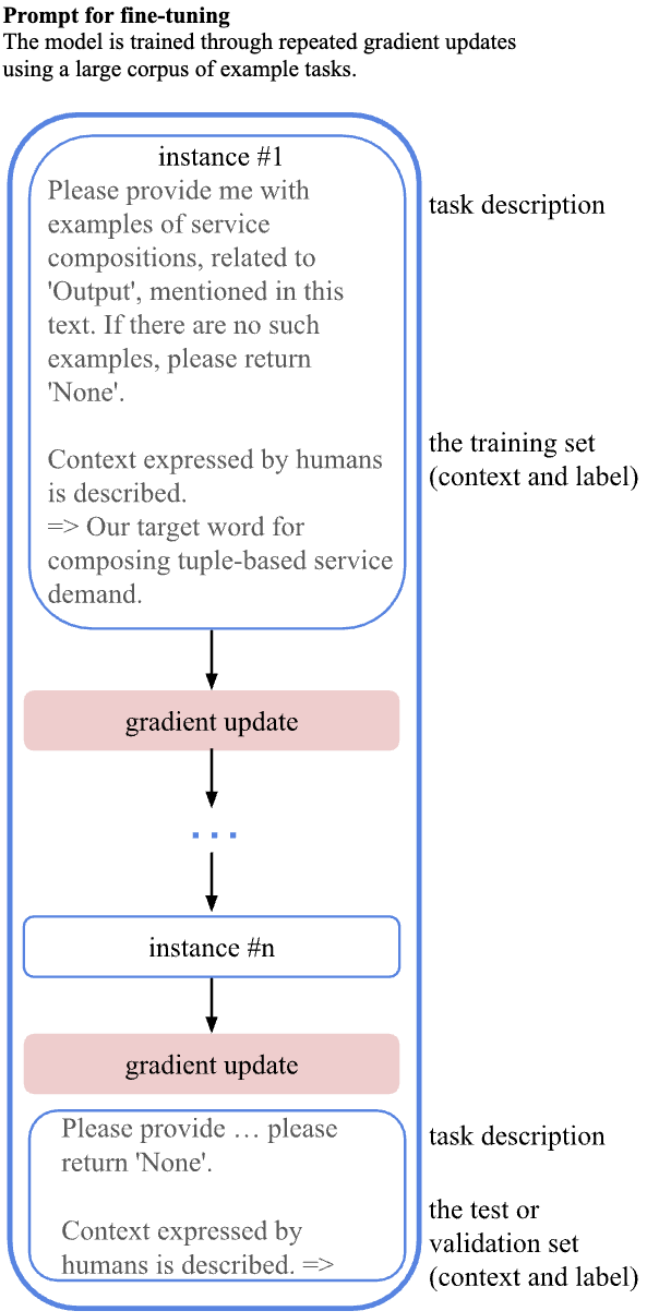

Fig. 2. Prompt design

During the fine-tuning procedure, we aim to minimize the difference between the instance's label and the LLM's extraction. In the validation and testing procedures, the process is composed only of the task description and the context for extraction.

V. Experiments

In this section, we first describe our experimental setup, including detailed descriptions of our dataset, hardware, and software configurations. Our goal is to ensure that the experiments conducted can be easily replicated by other researchers.

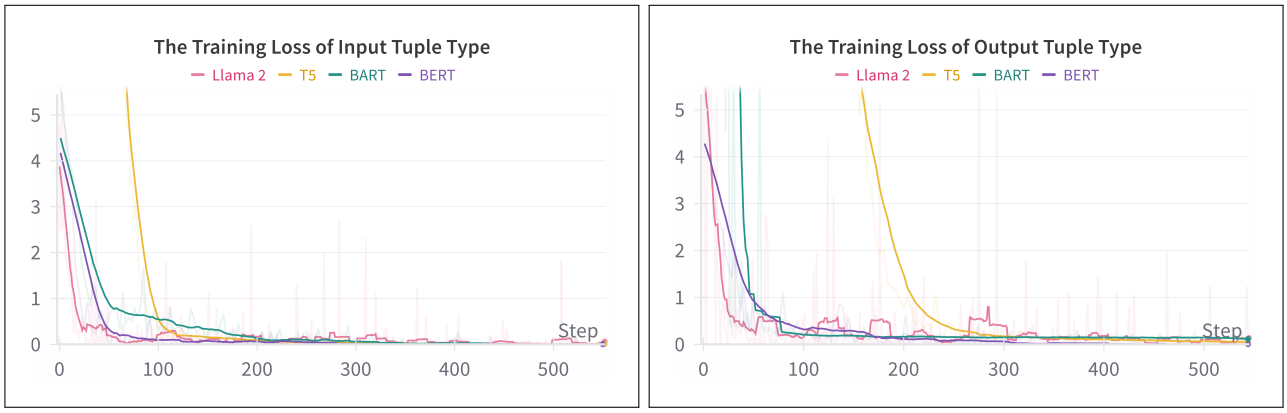

Fig. 3. Training Loss across steps for BERT, BART, T5, and Llama2

Secondly, we illustrate the evaluation metrics used for evaluating each model.

A. Experimental Setup

This subsection introduces our dataset preparation, environment setup, and the procedure for fine-tuning the LLM.

We have systematically prepared our experimental dataset to ensure our model's performance is robust and generalizable. The dataset consists of 500 entries dedicated to exploring the extraction of tuple-based service demands utilizing LLM for ASC. This dataset, primarily sourced from LeetCode, has been enriched with manual annotations of *Input* and *Output* tuples, as discussed in subsection IV-A. To eliminate any potential bias introduced by the original order of the data, we shuffled the entries to randomize the order. Subsequently, we divided the dataset into three segments: 64% for the training phase, 16% for the validation phase, and 20% for the testing phase.

Our environment setup is divided into two parts: firstly, we demonstrate that LLM can operate on a standalone computer, implying that tuple elements can be extracted using minimal resources; secondly, we aim to achieve optimal performance by running the LLM online. The related setup details are listed below:

- For the standalone computer environment: Operating System: Ubuntu 22.04 LTS; CPU: Intel Core i9-13900K; GPU: NVIDIA GeForce RTX 4090 24GB; Memory: 64GB DDR5. The models used for fine-tuning include bert-base-uncased, bart-large, t5-large, and Llama-2-7b-chat-hf.
- For the online environment: We utilize the OpenAI fine-tuning API with the model gpt-3.5-turbo-0613.

The fine-tuning procedure varied slightly due to each model's distinct capabilities and architectures. Using a classification approach, BERT was fine-tuned for a keyword extraction task within a QA framework. In contrast, BART, T5, and Llama 2 were treated as sequence-to-sequence (seq2seq) tasks, aligning with their capabilities for generating text outputs from input sequences. This differentiation underlines the versatility and adaptability of modern seq2seq models for complex NLP tasks such as keyword extraction from questions.

We employed Supervised Fine-Tuning (SFT) and Low-Rank Adaptation (LORA) strategies [13] regarding our implementation details. SFT tailored the models to our specific task through continued training on a domain-specific dataset, whereas LORA provided a parameter-efficient method for managing large models. Specifically, LORA introduces new matrices that interact with the existing model parameters to adjust the model without directly altering the original model weights. In this study, the sizes are 262M parameters and 7B parameters, respectively. This efficiency highlights the effectiveness of LORA and supports our claim that methodologies like ours can achieve outcomes comparable to those utilizing high-performance computing (HPC) resources, similar to the infrastructure supporting ChatGPT. Training parameters were meticulously optimized following initial validations, with a learning rate adjustment strategy based on gradient normalization to ensure thorough convergence. Coupled with a training span of 10 epochs, this methodology sustained high accuracy levels, further evidencing that cutting-edge computational research can be conducted on commercially available computing platforms. This advancement underscores the potential for employing locally accessible computing resources to execute resource-demanding AI models.

Utilizing online OpenAI GPT API for fine-tuning necessitates converting our dataset into the chat-completion JSON format with quality prompts, as discussed in subsection IV-C. Subsequently, we monitor various metrics, including training set loss, training set accuracy, validation set loss, and validation set accuracy, with the loss defined in Equation 2, to fine-tune hyperparameters. In our experiments, we achieved optimal results with n_epochs = 3.

B. Evaluation Metrics

This subsection describes the evaluation metrics used to assess the models' performance. Accurately extracting all correct fragments that correspond to the context is crucial for extracting the target content for a tuple-based request. Therefore, our primary focus is on evaluating the accuracy of the extracted complete content.

In addition to assessing the completeness of the extracted content, we aim to evaluate the models' performance across various scenarios. These scenarios include instances where a context is associated with multiple labels, lacks an existing label, or has a single label that the model accurately predicts. Consequently, we define common scenarios regarding extraction results corresponding to the classification in the confusion matrix as follows:

- True Positive: In this scenario, for a given description or sentence, the tuple elements extracted by the Approach must exactly correspond to the manually annotated tuple elements of the description (i.e., the correct answers). In this study, we classify a result in this category based on whether the extractions made by the Approach closely match the tuple elements we have labeled.
- False Positive This scenario is also referred to as a Type-1 Error. In such cases, for a given description, the Approach extracts tuple elements that do not appear in the corresponding ground truth annotation, adversely affecting its judgment of correctness. Additionally, this category encompasses situations where the Approach retrieves correct elements along with redundant segments or elements, impairing its judgment's accuracy.
- True Negative No elements should be extracted from a description in this scenario, and the Approach correctly retrieves nothing.
- False Negative This scenario is also referred to as a Type-2 Error. It occurs when the Approach fails to extract the intended tuple elements from a description, resulting in classification into this category. Furthermore, this category encompasses situations where the Approach extracts correct elements with insufficient details, rendering the retrieved tuple elements ambiguous or unclear.

We evaluate each tuple element we have labeled when assessing the set of elements predicted by the Approach. A service description set may be classified into different error types if extractions are incorrect (e.g., a service description with two labels: one label, which the Approach predicts correctly, is classified as True Positive; the other, which the Approach fails to predict, is classified as False Negative). This evaluation metric offers two advantages: first, it allows us to assess the accuracy of every label; second, it enables us to identify significant partitions from which we aim to extract tuple elements.

C. Result and Discussion

Model (Parameter size)	Accuracy
Rule-based [4]	0.56
BERT (336M)	0.63
BART (406M)	0.80
T5 (770M)	0.89
Llama 2 (7B)	0.91
GPT-3.5 (175B)	0.97

TABLE I
THE PERFORMANCE OF ALL MODELS IS EVALUATED BASED ON THE ACCURACY OF SENTENCE EXTRACTION CONCERNING THE *Input* TUPLE TYPE.

Model (Parameter size)	Accuracy
Rule-based [4]	0.48
BERT (336M)	0.67
BART (406M)	0.61
T5 (770M)	0.89
Llama 2 (7B)	0.93
GPT-3.5 (175B)	0.98

TABLE II
THE PERFORMANCE OF ALL MODELS IS EVALUATED BASED ON THE ACCURACY OF SENTENCE EXTRACTION CONCERNING THE *output* TUPLE TYPE.

Model	Accuracy	Precision	Recall	F1-score
Rule-based [4]	0.418	0.528	0.667	0.589
BERT	0.529	0.74	0.649	0.692
BART	0.702	0.844	0.807	0.825
T5	0.823	0.911	0.895	0.903
Llama 2	0.861	0.929	0.921	0.925
GPT-3.5	0.957	0.982	0.974	0.978

TABLE III
THE PERFORMANCE OF ALL MODELS IS EVALUATED BASED ON THE ACCURACY OF LABEL EXTRACTION CONCERNING THE *Input* TUPLE TYPE.

Through the evaluation metric based on a set of descriptions or a tuple from a set, we compare our previous work [4] (a rule-based approach), a method used in related work [16] (BERT), standalone approaches (BART, T5, and Llama 2), and a current outstanding approach (GPT-3.5).

In the analysis of accuracy for each candidate of service demand (see Tables I and II), we observed that the rule-based approach with initial rules is not as effective on larger-scale datasets as it is on the original dataset. To be more specific, the rule-based approach needs to set additional rules for larger-scale datasets to achieve higher performance. Consequently, the task of recognizing suitable rules requires a significant amount of human effort from ASC experts. On the other hand, the remaining LLM approaches achieve outstanding performance, surpassing the rule-based methods. Our experiments demonstrate that the parameter size of the model significantly affects performance across different models (see Tables III and IV). A similar trend was observed in evaluating each service demand candidate.

Our GPT-3.5 model, augmented with prompt design, demonstrates outstanding performance regardless of whether the evaluation metric is based on a set of descriptions or a tuple from a set, and irrespective of whether the tuple elements are *Input* or *Output*; it nearly achieved all correct answers. During the process of achieving these results, we experimented with different placeholders and official prompts multiple times.

Model	Accuracy	Precision	Recall	F1-score
Rule-based [4]	0.379	0.500	0.500	0.500
BERT	0.591	0.788	0.703	0.743
BART	0.522	0.691	0.631	0.66
T5	0.84	0.918	0.891	0.904
Llama 2	0.912	0.969	0.93	0.949
GPT-3.5	0.982	1.00	0.980	0.990

TABLE IV
THE PERFORMANCE OF ALL MODELS IS EVALUATED BASED ON THE ACCURACY OF LABEL EXTRACTION CONCERNING THE *output* TUPLE TYPE.

We discovered that prompt design is a crucial factor in GPT-3.5 usage. Among a multitude of prompt design experiments, we found that using common keywords, such as "*Extract*," which are typical for standard extraction tasks, does not yield high performance in ASC tasks. Consequently, avoiding these words and incorporating more ASC-related information is necessary.

This demonstrates that GPT-3.5, through fine-tuning with effectively designed prompts that leverage knowledge from our dataset, represents a highly flexible approach to addressing the problem defined in Section III. Additionally, an extra advantage of GPT-3.5 became evident during fine-tuning and the extraction of tuple elements: it can automatically correct some mistakes, such as grammatical errors, that were inadvertently introduced in our labels. This capability is not present in rule-based approaches. With this advantage, we can more accurately extract tuple-based service demands for ASC.

On the other hand, we observed that Llama 2, through LORA technology, achieves convergence at an early epoch (see Figure 3), despite its large parameter size of 7 billion, surpassing that of BERT, BART, and T5. Remarkably, this model demonstrates high performance while utilizing only limited computer resources.

Furthermore, all LLM approaches achieve high precision, enabling the extraction of correct elements and seldom producing non-existent ones. This precision is crucial to prevent the production of confusing elements that are difficult for humans to distinguish.

VI. Conclusion

ASC approaches typically require inputs in a specifically designated tuple-based format, presenting challenges. This study introduces feasible LLM strategies, appropriate prompt design, and a detailed fine-tuning process for ASC. Compared to our previous rule-based Approach, the LLM strategies accommodate greater complexity and scale more effectively to new datasets. We show that our approaches can efficiently and effectively extract tuple-based service demands for ASC.

Besides the *Input* and *Output* tuple types, there are other functional and nonfunctional property sets of a described service. In the future, we plan to enhance our Approach by including various tuple types. By achieving complete extraction with tuple-based approaches, a higher level of automation can reduce human intervention and improve the productivity of ASC.

Acknowledgment

We thank Ming-To Chuang, an intern research assistant at the National Taiwan University, Department of Electrical Engineering, for his assistance with our dataset construction.

References

[1] Y. Fanjiang, Y. Syu, S. Ma, and J. Kuo, "An overview and classification of service description approaches in automated service composition research," *IEEE Transactions on Services Computing*, vol. 10, no. 02, pp. 176–189, apr 2017.

[2] Y. Syu and C.-M. Wang, "A gap between automated service composition research and software engineering development practice: Service descriptions," in *Web Services – ICWS 2023*, Y. Zhang and L.-J. Zhang, Eds. Cham: Springer Nature Switzerland, 2023, pp. 18–31.

[3] Z. Zhang, Y. Elkhatib, and A. Elhabbash, "Nlp-based generation of ontological system descriptions for composition of smart home devices," in *2023 IEEE International Conference on Web Services (ICWS)*, 2023, pp. 360–370.

[4] Y. Syu, Y.-J. Tsao, and C.-M. Wang, "Rule-based extraction of tuple-based service demand from natural language-based software requirement for automated service composition," in *Services Computing – SCC 2021*, A. Katangur and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2022, pp. 1–17.

[5] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[6] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 740–750. [Online]. Available: https://aclanthology.org/D14-1082

[7] M. Miwa and M. Bansal, "End-to-end relation extraction using LSTMs on sequences and tree structures," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1105–1116. [Online]. Available: https://aclanthology.org/P16-1105

[8] 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 pretraining approach," 2019.

[9] [Online]. Available: https://leetcode.com/problemset/

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *arXiv preprint arXiv:1810.04805*. arXiv: arXiv, Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *arXiv preprint arXiv:1910.13461*. arXiv: arXiv, Oct. 2019. [Online]. Available: https://arxiv.org/abs/1910.13461

[12] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," in *arXiv preprint arXiv:1910.10683*. arXiv: arXiv, Oct. 2019. [Online]. Available: https://arxiv.org/abs/1910.10683

[13] H. Touvron, L. Martin, K. Stone *et al.*, "Llama 2: Open foundation and fine-tuned chat models," in *arXiv:2307.09288*. https://ar5iv.org/abs/2307.09288: arXiv, 2023. [Online]. Available: https://arxiv.org/abs/2307.09288

[14] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018, work in progress.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[17] [Online]. Available: https://platform.openai.com/docs/guides/