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

Chih-Jung Hsu[1], Yu-Xiang Luo[2], Yi-Chen Mao[2], Chien-Min Wang[1], and Yang Syu[3]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan
{hsu8591,cmwang}@iis.sinica.edu.tw
[2] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan {b10902037,b10902103}@csie.ntu.edu.tw
[3] 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 (LLMs)—where significant achievements have been made in real-time automatic text generation tasks—we propose leveraging LLMs 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 implemented an appropriate workflow that optimizes LLMs performance. Our experiments and results demonstrate that our proposed LLMs-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.

Keywords: Automatic Service Composition · Service Description · Large Language Model.

1 Introduction

Automated Service Composition (ASC) has been a widely studied research subject in academia for many years [3, 15, 18]. Research on ASC [15] focuses 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 [14] [15], 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 effective approach in this paper.

We employ large language models (LLMs) to recognize requirement descriptions and extract critical partitions. In recent years, LLMs 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 LLMs 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 LLMs.

With rigorous experimental results, we demonstrate that LLMs-based methods can effectively address the challenge of establishing a standard format for human-written service descriptions. This approach surpasses our previous work in sentence accuracy and enables the rapid construction of prompts that are more comprehensible than those generated by our prior rule-based design. Furthermore, we have conducted extensive experiments on extracting service demand through LLMs fine-tuning and have developed generalized, effective, and efficient prompts suitable for this process. Our approach outperforms the original LLMs version, achieving superior results.

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 approach in detail. Section V presents our experimental results and includes a discussion of these findings. Finally, we conclude our work in Section VI.

2 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. [3] 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 [3] is as follows:

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

where I, O, P, E, and NF represent the input, output, precondition, effect, 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 [3].

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 remains a viable approach due to its reliance on predefined rules that establish relationships between manual documents and tuple-based requirements. In previous work [14], [15], specific rules related to word positioning within requirement descriptions were identified through detailed analysis. Additionally, the NLP tool Stanford CoreNLP [8] was employed to parse and tag a dataset composed of sentences from the LeetCode website. This process involved the development of a part-of-speech (POS) tagger, which assigns parts of speech—such as nouns, verbs, and adjectives—to each word in the text. The grammatical structure revealed by these tags facilitated the creation of rules for more complex cases, particularly for sentences that did not belong to critical sets. Consequently, these structured rules enabled the extraction of target words across different classifications.

Machine learning-based knowledge extraction is favored for its potential to achieve higher performance than traditional methods. Miwa et al. [9] proposed using LSTM-RNNs to enable the simultaneous extraction of entities and relations. Zhang et al. designed a holon ontology structure and selected RoBERTa-base-SQuAD2[3] [18], a pre-trained model utilizing the Robustly Optimized BERT Approach (RoBERTa) [7]. Using this model, they converted raw data into a holon description in JSON format, which was then transformed into a holon class. This process resulted in unified format information that could be easily recognized by their IoT devices. Recently, LLMs, such as BERT [2], BART [5], T5 [13], Llama 2 [16], and GPT [11, 12, 1], have come to dominate the field of question answering due to their ability to generate effective responses. Consequently, we aim to leverage the capabilities of these LLMs to build an automated workflow for service demand extraction.

3 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 [4], 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 4.1):

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 [14], 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:

$$LLM(\bar{X} \rightarrow \bar{Y}) \rightarrow F, and \quad then \quad F : X \rightarrow Y \tag{1}$$

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.

4 Approach

In this study, we face several challenges, including the scarcity of service demand data sources and the accuracy of extracting tuple-based service demand information. To address these issues, we will first focus on collecting suitable data.

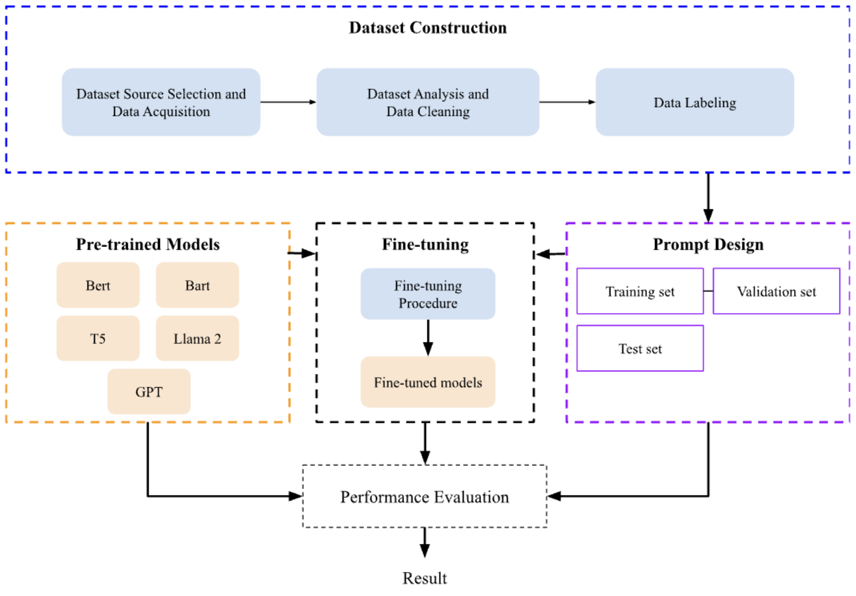

Fig. 1. Workflow of our approach

Subsequently, we will analyze various LLMs to select the most appropriate model and design for our objectives. The overall workflow is depicted in Figure 1.

4.1 Dataset Construction

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 functional descriptions from LeetCode.

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, preserving a broad spectrum of programming and algorithmic challenges while aligning more closely with our study's focus on accurate service demand information.

Data Labeling A pivotal step in refining our dataset for analysis involved manually annotating the I and 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 ensured high reliability in our dataset, which is crucial for the effectiveness of our study. The meticulous attention to detail during the data labeling stage provided a robust foundation for our research, facilitating the development and assessment of our models. This dataset collection process is a cornerstone of our study, enabling the creation and evaluation of models designed to extract service demands. The following entries are formatted according to the processes described above:

- **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 extraneous information such as HTML tags, focuses solely on the textual content. This cleaned version enhances accessibility for analysis and processing.
- **Label(Main):** The service demand pertains to the *Input* or *Output* in the processed content. It is important to note that there can be multiple instances of service demand information.

4.2 Pre-trained Models vs. Fine-tuning

In recent years, various LLMs have achieved high performance in natural language processing fields. To popularize our approach, we aim to select LLMs that can work effectively and efficiently with limited computational resources and data. Consequently, we chose indicative LLMs such as BERT, BART, T5, Llama 2, and GPT due to their outstanding pre-trained model performance.

Given the substantial resources required to retrain a new LLM, two mainstream approaches have been developed to achieve task-specific goals in natural language processing [6].

The first approach involves leveraging pre-trained models without additional training or fine-tuning on a specific dataset. This method is prevalent in LLMs, where the model is applied to a task without prior exposure to task-specific data, relying solely on its pre-existing knowledge—examples include zero-shot learning. Alternatively, the model can provide one or more examples of task-specific data to improve performance, as demonstrated in one-shot and few-shot learning. These approaches enable LLMs to generate accurate responses while conserving computational resources and minimizing the time required for fine-tuning. Additionally, they reduce the need for large amounts of labeled data.

The second approach entails fine-tuning pre-trained models on a specific dataset tailored to the task at hand. In this approach, the model undergoes additional training with a substantial amount of task-specific data, allowing its parameters to be refined and optimized for the target task. This method enables LLMs to achieve high accuracy in task-specific applications while also reducing the substantial resource requirements associated with training a new LLM from scratch.

Due to computational cost considerations, we initially employed zero-shot, one-shot, and few-shot learning methods, as these approaches can significantly reduce the time required for model tuning. However, our experiments revealed that existing LLMs exhibited limited performance on the specific service demand extraction task when relying solely on pre-trained models without fine-tuning. Consequently, we proceeded to fine-tune the pre-trained model using our training dataset. During the fine-tuning process, we aimed to minimize the discrepancy between the instance's label and the LLM's extracted output. In the testing phase, we concatenated the instruction designed for the service demand task with

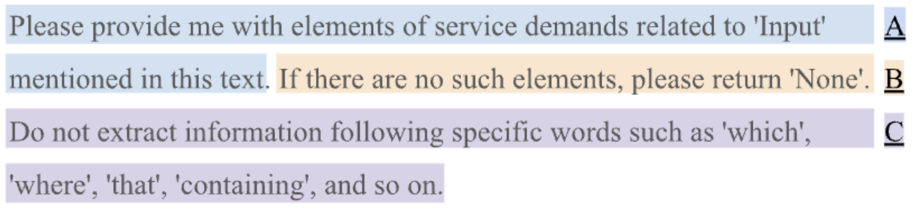
Please provide me with elements of service demands related to 'Input' mentioned in this text. If there are no such elements, please return 'None'. Do not extract information following specific words such as 'which', 'where', 'that', 'containing', and so on.

Fig. 2. Prompt Design

the relevant information and the human-expressed textual data. This integrated input is then fed into the LLMs to extract the service demand. Further details on this procedure will be discussed in the following section.

4.3 Prompt Design

Creating an appropriate prompt is one of the most critical tasks in leveraging LLMs. Mainstream LLMs are typically applied to general question-and-answer tasks rather than specific tasks such as service demand analysis. Therefore, the quality of this study heavily depends on the amount of information provided and the precision of the prompt design.

In our observation, we encountered three main issues affecting the accuracy of extracting service demand through GPT. First, the common keyword extraction tasks following the prompt principles outlined in [10] are not appropriate for specific tasks. Standard prompts for keyword extraction often fail to precisely capture service demand information, instead retrieving extraneous details. Second, people's descriptions of their needs do not always contain explicit service demand information regarding *Input* or *Output*. Therefore, it is a challenge to ensure the model returns correct information when no explicit service demand is present. Finally, individuals often include extraneous expressions surrounding the service demand information, which are not the target of our extraction process. Thus, it is crucial to develop prompts that accurately extract only the key information.

To address the aforementioned challenges, we have structured our prompt into three distinct parts to enhance the accuracy of extracting service demands. A prompt can include instructions or questions directed to the model, along with additional elements such as context, inputs, or instances. By incorporating these components, we can more effectively guide the model, thereby improving the quality of the results. Following this principle, the prompt design is divided into three components: Part A provides clear instructions for the service demand extraction task, Part B specifies the response format when the context does not contain any service demand, and Part C focuses the response on extracting specific segments of the text (see Figure 2).

In part A, we replace the standard prompt example [10], which typically extracts keywords from a block of text, with more precise instructions. Using the term *Extract* in the prompt could lead the LLM to extract general keywords rather than specific service demands. To mitigate this issue, we use alternative verbs such as "provide" and "show," and include a tip indicating that the task involves identifying a specific type of service demand.

In part B, we consider how the LLM should respond when no service demand is present in the textual requirements. Therefore, we design the prompt to account for these scenarios by specifying the output "None" when the text does not contain any service demand. This fixed-format simplifies subsequent data processing.

In part C, the presence of inessential information surrounding the correct service demand can complicate downstream data processing. We observed that such extraneous information often appears near specific words. To address this, we include related words in our prompt to ensure that only the necessary service demand information is extracted. Our experiments demonstrate that replacing the verb "extract" with other related words in this third part does not negatively impact performance.

5 Performance Evaluation

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

5.1 Experimental Setup

This subsection introduces our dataset preparation, environment setup, and the procedure for LLMs.

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 LLMs. This dataset, primarily sourced from LeetCode, has been enriched with manual annotations of *Input* and *Output* tuples, as discussed in subsection 4.1. 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 LLMs 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 LLMs 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 and gpt-4-turbo.

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.

Utilizing the online OpenAI GPT API for zero-shot, few-shot, and fine-tuning learning necessitates converting our dataset into the chat-completion JSON format with quality prompts, as discussed in subsection 4.2. During fine-tuning, we monitor various metrics, including training set loss and training set accuracy, to adjust hyperparameters. In our experiments, we achieved optimal results with $n\_epochs = 3$ to 5.

5.2 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.

5.3 Result and Discussion

Table 1. The performance of zero-shot, one-shot, and few-shot (10 cases) learning is evaluated based on the accuracy of sentence extraction for the *Input* and *Output* tuple types. These evaluations is conducted using the GPT-4 model.

Method	**Accuracy Input**	**Accuracy Output**
Zero-shot learning	0.00	0.11
One-shot learning	0.01	0.34
Few-shot learning	0.66	0.60

Table 2. The performance of few-shot learning (10 cases) and fine-tuning is evaluated based on the time required to extract both the *Input* and *Output* tuple types from the test cases. For these evaluations, we employed a few-shot learning approach using the GPT-4 model and a fine-tuned approach using the GPT-3.5 model.

Method	**Time(s) Input**	**Time(s) Output**
Zero-shot learning	153	172
One-shot learning	154	192
Few-shot learning	185	231
Fine-tuning	92	85

Performance of the Pre-trained Model In recent years, LLMs have begun to dominate many natural language processing tasks. It is widely recognized that a single pre-trained LLM can be applied to various natural language tasks without fine-tuning. Initially, we present the results of utilizing a pre-trained model without fine-tuning to achieve our goals (see Fig. 2).

We observed poor performance when directly extracting service demand through the pre-trained GPT-4 without any training set information (i.e., zero-shot learning) (see Table 1). Upon examining the extracted segments, we found that the original model did not adequately identify what constituted service demand information. Instead, it tended to display its inference procedure and the answer. Since the model lacked knowledge about service demand, it frequently inferred the answer as "None," as shown below:

> The text does not explicitly mention any elements related to 'Input' in terms of service demands. It only describes a problem scenario involving an array of strings and the process of deleting characters from specific indices to achieve a desired order. Therefore, the answer is 'None'.

However, for the purpose of ASC, we need to obtain clearer service demand segments to efficiently produce the subsequent service composition. A series of intermediate reasoning steps significantly improves the ability of large language models to perform complex reasoning [17]. We designed the reasoning instructions using our training set information to help the LLM understand which parts we want to extract as service demands (i.e., one-shot learning or few-shot learning). We observed improved performance compared to zero-shot learning (see Table 1)).

In our observations, performance improves significantly when the training set increases to 10 cases. The segments extracted by the LLM become clearer and more accurate for the service demand information we need. The redundant step-by-step inferences gradually disappear, resulting in impressive performance without fine-tuning the pre-trained GPT-4. However, increasing the amount of training data in the instructions is not an ideal approach, as it requires the model to scan more phases before answering each testing question, thereby wasting computational resources and increasing response time (see Table 2). Therefore, we employed the fine-tuning method in the subsequent phase.

Table 3. The performance of our designed prompt is compared to that of a standard extraction task prompt [10], with evaluations based on the accuracy of sentence extraction for both the *Input* and *Output* tuple types.

Model	**Prompt**	**Accuracy Input**	**Accuracy Output**
Few-shot learning	Standard	0.20	0.21
	Designed	0.66	0.60
Fine-tuning	Standard	0.94	0.96
	Designed	0.97	0.98

Comparison of Designed Prompt and Standard Prompt Our experiments revealed that prompt design plays a critical role in the performance of LLMs. In Section 4.3, we detail our approach to creating an effective prompt, developed through iterative testing with various placeholders and official prompts. We observed that a standard prompt typically used for extraction tasks is insufficient for specific tasks such as our service demand research, particularly when the pre-trained model has not been fine-tuned for this specific task. However, by employing a designed prompt, we achieved an accuracy improvement of over 40% in few-shot learning using the pre-trained model (see Table 3).

Even after fine-tuning the pre-trained model to incorporate knowledge specific to the service demand task, we observed further performance improvements when utilizing our designed prompt. Specifically, there was a notable increase in accuracy when avoiding generic keywords such as "Extract," which are typically employed in standard extraction tasks. Remarkably, we achieved 100% accuracy in cases where no service demand was present, and the extracted service information consistently excluded any redundant segments following the words specified in the prompt. These findings highlight the critical importance of excluding generic terms and incorporating task-specific information into the prompt design.

Table 4. The performance of all models was evaluated based on the accuracy of sentence extraction related to both the *Input* and *Output* tuple types.

Model (Parameter size)	**Accuracy Input**	**Accuracy Output**
Rule-based [14]	0.56	0.48
BERT (336M)	0.63	0.67
BART (406M)	0.80	0.61
T5 (770M)	0.89	0.89
Llama 2 (7B)	0.91	0.93
GPT-3.5 (175B)	0.97	0.98

Table 5. The performance of all models is evaluated based on the accuracy of label extraction concerning the *Input* tuple type.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Rule-based [14]	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 6. The performance of all models is evaluated based on the accuracy of label extraction concerning the *Output* tuple type.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Rule-based [14]	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

Performance of Fine-tuning Using evaluation metrics based on a set of descriptions or tuples, we employed the related work [14], which utilizes a rule-based approach, as the baseline model for comparison. We then assessed the performance of various fine-tuned LLMs, including BERT, BART, T5, Llama 2, and GPT-3.5, against this baseline.

In the analysis of accuracy for each service demand candidate (see Table 4), a score is assigned only when the fine-tuned model correctly predicts all elements of the service demand for each description. This stringent criterion ensures that the reported accuracy reflects the model's ability to fully capture the necessary details of the service demand in each case. We observed that the rule-based approach, when applied with initial rules, is less effective on larger-scale datasets compared to its performance on the original dataset. Specifically, the rule-based method requires additional rule-setting to achieve higher performance on larger datasets. This task of identifying suitable rules demands substantial human effort from ASC experts.

Furthermore, in our analysis of the accuracy for each service demand candidate, a score is given when the fine-tuned model correctly predicts every element of the service demand for each description (see Tables 5 and 6. We observed that fine-tuned LLMs achieve high precision, enabling the extraction of correct elements while rarely producing non-existent ones. This precision is crucial in preventing the generation of misleading elements that could be difficult for humans to distinguish.

In contrast, the other LLM approaches consistently outperformed the rule-based methods. Our experiments demonstrate that the model's parameter size significantly impacts performance across different models (see Table 4). A similar trend was observed in the evaluation of each service demand candidate (see Tables 5 and 6.

Our GPT-3.5 fine-tuning model, augmented with prompt design (see subsection 4.2), 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. This demonstrates that GPT-3.5, when fine-tuned with effectively designed prompts leveraging knowledge from our dataset, represents a highly flexible approach to addressing the problem defined in Section 3. Moreover, it is more efficient compared to GPT-4 with few-shot learning during the extraction of service demands from the testing set, saving half the cost and about 2 times faster while achieving greater accuracy (see Table 2). An additional advantage of GPT-3.5 became evident during fine-tuning and the extraction of tuple elements: it can automatically correct 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.

6 Conclusion

ASC approaches typically require inputs in a specifically designated tuple-based format, presenting several challenges. Our study introduces a representative dataset we collected, along with reasoning designs and strategies for LLMs, such as appropriate prompt design, and a detailed few-shot learning and fine-tuning process. Compared to our previous rule-based approach, the LLM strategies accommodate greater complexity and scale more effectively to new datasets. We demonstrate that our approach can efficiently and effectively extract accurate tuple-based service demands for ASC.

In addition to 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 such as precondition and postcondition. By achieving complete extraction with tuple-based approaches, a higher level of automation can be attained, reducing human intervention and improving the productivity of ASC.

Acknowledgments. This work was partially supported by the National Science and Technology Council, Taiwan, under Grant No. NSTC112-2221-E-001-008. We thank Ming-To Chuang, an intern research assistant at the National Taiwan University, Department of Electrical Engineering, for his valuable assistance with our dataset construction.

References

1. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: arXiv preprint arXiv:1810.04805. arXiv, arXiv (Oct 2018), https://arxiv.org/abs/1810.04805

3. Fanjiang, Y., Syu, Y., Ma, S., Kuo, J.: An overview and classification of service description approaches in automated service composition research. IEEE Transactions on Services Computing **10**(02), 176–189 (apr 2017). https://doi.org/10.1109/TSC.2015.2461538
4. https://leetcode.com/problemset/, online
5. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: arXiv preprint arXiv:1910.13461. arXiv, arXiv (Oct 2019), https://arxiv.org/abs/1910.13461
6. Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., Raffel, C.A.: Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 1950–1965. Curran Associates, Inc. (2022), https://proceedings.neurips.cc/paper_files/paper/2022/file/0cde695b83bd186c1fd456302888454c-Paper-Conference.pdf
7. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach (2019)
8. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), http://www.aclweb.org/anthology/P/P14/P14-5010
9. Miwa, M., Bansal, M.: End-to-end relation extraction using LSTMs on sequences and tree structures. In: Erk, K., Smith, N.A. (eds.) Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1105–1116. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/P16-1105, https://aclanthology.org/P16-1105
10. https://platform.openai.com/docs/guides/, online
11. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018), work in progress
12. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
13. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. In: arXiv preprint arXiv:1910.10683. arXiv, arXiv (Oct 2019), https://arxiv.org/abs/1910.10683
14. Syu, Y., Tsao, Y.J., Wang, C.M.: Rule-based extraction of tuple-based service demand from natural language-based software requirement for automated service composition. In: Katangur, A., Zhang, L.J. (eds.) Services Computing – SCC 2021. pp. 1–17. Springer International Publishing, Cham (2022)
15. Syu, Y., Wang, C.M.: A gap between automated service composition research and software engineering development practice: Service descriptions. In: Zhang, Y., Zhang, L.J. (eds.) Web Services – ICWS 2023. pp. 18–31. Springer Nature Switzerland, Cham (2023)
16. Touvron, H., Martin, L., Stone, K., et al.: Llama 2: Open foundation and fine-tuned chat models. In: arXiv:2307.09288. arXiv, https://ar5iv.org/abs/2307.09288 (2023), https://arxiv.org/abs/2307.09288
17. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language

models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2024)

18. Zhang, Z., Elkhatib, Y., Elhabbash, A.: Nlp-based generation of ontological system descriptions for composition of smart home devices. In: 2023 IEEE International Conference on Web Services (ICWS). pp. 360–370 (2023). https://doi.org/10.1109/ICWS60048.2023.00055