A Gap Between Automated Service Composition Research and Software Engineering Development Practice: Service Descriptions

Yang Syu[1,2](✉) and Chien-Min Wang[2]

[1] Department of Information Science, National Taipei University of Education, Taipei City, Taiwan (R.O.C.)
yangsyu@mail.ntue.edu.tw

[2] Institute of Information Science, Academia Sinica, Taipei City, Taiwan (R.O.C.)
cmwang@iis.sinica.edu.tw

Abstract. In research, automatic service composition (ASC) has been a widely studied academic subject for many years. However, this field still contains topics and issues that remain unidentified or uninvestigated. In this paper, we focus on one such unsolved problem of the ASC, elaborating on a current effort and future plan.

This recognized problem is caused by the difference between the formation of service descriptions used by human composers and that used by ASC approaches. In practice, engineers are used to write various development-related documents in natural language, such as their requirement specifications and software component descriptions. However, an exhaustive survey found that most existing ASC studies are assumed to take their required service descriptions in a tuple-based format. Although this difference and problem in theory can be sufficiently addressed using manual processing techniques (e.g., human transformation or extraction), we consider such human intervention to be inefficient, costly, and, most importantly, harmful to the level of automation of ASC. Thus, this study develops an automated solution to this problem.

We first introduce some necessary background knowledge and foundation so that the targeted problem can be fully understood and motivated. Then, the specific problem to be studied is clearly defined and exemplified along with a detailed explanation. Finally, the three key components to explore and address this research problem (dataset, approach, and evaluation) are discussed in detail, including the current work of the authors and proposals for future research.

Keywords: automatic service composition · service description · natural language processing · rule-based extraction · conditional random field · deep learning

1 Introduction

In academia, automated (Web) service composition (ASC/AWSC) has been a hotly and widely studied research subject for many years, and there are already plenty of ASC approaches and related tools, theories, and (reference) standards that have been published and are available. However, there remain issues and topics that have never been identified or studied in ASC research. To improve the applicability and successful implementation of ASC, these unsolved problems must be clearly defined, thoroughly investigated, and properly addressed. This paper first identifies one such ASC issue – the tuple-based service descriptions used and assumed in most existing ASC studies – and then discusses its remedy. Because the proposed and targeted issue is in ASC, to ensure clear understanding, we introduce service composition (SC) and its automation (i.e., ASC/AWSC) in the remainder of this section. Then, the problem of interest is formulated and explained in detail in the next section.

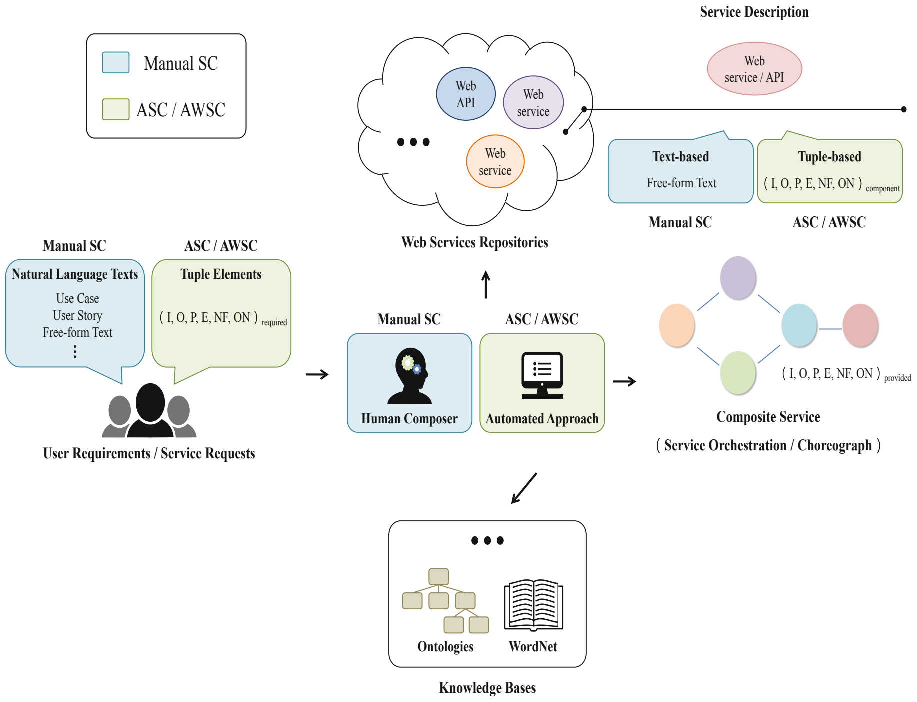

Fig. 1. Graphical process and comparison of design-time manual SC and ASC/AWSC.

Overall, the lifecycle of an SC can be divided into two stages: the design-time phase and runtime execution. A more complex model could exist, but for simplicity, in this study we introduce SC based on this basic model. More specifically, the discussion and introduction below exclusively focus on the first stage of SC, which concentrates on the entire and complete generation of a composite service (CS) from scratch to meet a received user request or customer requirement and, according to the proposed investigation, is also the primary concern and target of most surveyed ASC studies. As shown in Fig. 1, on its left-hand side, the creation of a CS (i.e., its design-time composition process) begins with a software requirement (service request), which is identical to most development process models proposed and used in software engineering [1]. Typically, such a requirement or request would be expressed using at least one of the formations widely used and developed in software requirement engineering, such as free-form texts, use case descriptions defined in UML, and user stories proposed in agile methods. The commonality between these different forms of demand expression is that they all consist of natural language statements to describe their writers' intentions: *natural language is used prevalently for expressing systems and software requirements* [2]. After having this stimulative information about what is required, a human composer, who is represented at the center of Fig. 1, takes time to read and analyze the message; assess and consider the available component services; and finally select and integrate a set of appropriate component services to design and produce a service orchestration or choreograph, which forms the desired CS, as shown on the right-hand side of Fig. 1. Regarding component services, the composer must understand and comprehend their specifications because programmers must carefully read the documents of their used program APIs before loading and calling them (e.g. functionality and nonfunctional properties) by inspecting their text-based service descriptions (on top of Fig. 1) because these available component services are often developed, offered, and owned externally by others.

Instead of manually composing services, researchers work on its automation, ASC, which is also shown in Fig. 1. Although ASC is a broad research field that includes and is associated with many different topics and aspects in detail, such as service discovery, matchmaking, and QoS prediction, overall, the basic elements involved in this automatic procedure resemble its manual version that we have explained in the previous paragraph. However, compared with regular/manual SC, some of these elements are different in their forms in ASC research, as shown in the figure and explained below. First, the most significant difference is that an ASC is performed automatically rather than by a human service composer, as shown at the center of Fig. 1. Second, the service descriptions involved in this automated process, which are the primary inspiration of this paper and this study work, including both the information of an intended/required service (i.e., a service requirement or composition request on the left-hand side of Fig. 1) and the descriptions for the available component services (on the top), are in forms of tuples, not in any of those natural language sentence-based formations introduced previously for software requirement engineering. In addition, the outcome and result of an instance of ASC (i.e., a CS) could be presented in a tuple-based way as well. Finally, for semantic ASC/AWSC, which has been widely considered a more sophisticated and advanced manner of automatic aggregation of services, at least one knowledge base, such as a self-defined domain ontology or a general WordNet, is required to have a reference or source of common/domain knowledge, as shown at the bottom of Fig. 1. As an exemplification, the above three types of service (description) correspond to the tuple-based required, component, and provided service types defined and used in an ASC study [3]. This process identified important differences between the formation of the service descriptions of ASC and the one used in the software development practice motivates this study.

Certainly, this problem, the processing (e.g., transformation or extraction) of natural language texts into their corresponding tuple-based elements, which is defined and explained in detail in Sect. 2, can be solved and addressed by human labor. However, this manual solution is inefficient and even ineffective. First, the biggest shortcoming of the involvement and intervention of humans is that it markedly decreases ASC's extent of automation and removes the goal and appeal of ASC, which is to reduce (or even entirely eliminate), as much as possible, manual tasks. Second, manually performing the handling process defined in Sect. 2 for the problem is both costly and time-consuming (and even impractical) because, as a set of materials/resources for composition, there could be numerous component services. For example, on ProgrammableWeb, over 15,000 WSs and Web APIs are registered and available. Similarly, the many incoming human-written, natural language sentence-based service requests and software requirements poses the same dilemma. Thus, a human labor-free solution for the problem is required to reach and have a fully automated composition of services. To our knowledge, this problem has never been identified, considered, or studied before; therefore, in Sect. 3, we elaborate on the authors' current effort and future research plan.

2 Problem

This section explains the identified and targeted problem in detail. One of the major findings of the proposed previous survey [4] of the service description approaches in ASC indicates the extensive adoption of a tuple-based formation for different types of service descriptions in this field of research. As reviewed and listed in [4], the set of tuple elements assumed and considered in an ASC study depends on its concern and coverage (awareness), and after a thorough investigation of existing ASC papers, as a universal service description model for ASC research, a generic, tuple-based paradigm proposed and used in the survey is:

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

where *I, O, P, E,* and *NF* are a described service's input, output, precondition, effect (postcondition), and nonfunctional property sets, respectively; and *ON connects to a knowledge ontology (or base) that formally and semantically defines the elements contained in other tuples* [4] (i.e., the component at the bottom of Fig. 1). Overall, as mentioned, which tuple elements are considered and used in an ASC study is determined by relevant aspects, and because different ASC considerations and their combinations have been studied in the past, a number of disparate groupings of these tuple elements have been used, such as *<I, O, P, E>*, *<I, O, QoS>* and *<I, O, P, E, QoS>*, as comprehensively described in Table 1 of [4]. Due to limited space, we now provide an explanation with an assumption, where the most basic functional service description model is considered and used: *<I and O>*.

As an example taken from an ASC study, a request that is acceptable to existing ASC approaches would be in the following format:

I = *(Book title, Book author, Credit card information, Address that the book will be shipped to),*

O = *(Payment from a credit card for the purchase, Shipping dates, Customs cost for the specific item)*

However, as explained in the Introduction, human demanders are used and preferable to, as below, express their intention as text in natural language:

"The user wishes to provide as inputs a book title and author, credit card information and the address that the book will be shipped to. The outputs of the desired composite service are a payment from the credit card for the purchase, as well as shipping dates and customs cost for the specific item."

Similarly, also sampled from an ASC research paper, the natural language sentence, which is *PatientByIDService takes as input a patient ID to return the patient's description*, describes the function of a component service. However, as mentioned before, to be usable in and compatible with most current ASC approaches, a tuple-based version of this service description, such as $I =$ *(Patient ID)* and $O =$ *(Patient's description)*, is required.

More specifically and formally, this study's goal is to seek an effective and accurate automation (F) that takes a received natural language text-based service/software description (X) as its input and working foundation and then produces a set of corresponding tuple elements (Y) of the description as its processing result for subsequent ASC operations:

$$\boldsymbol{F : X \rightarrow Y}$$

3 Solution

To solve this research problem, we consider three major issues that must be properly addressed: a dataset of ample natural language sentence-based service descriptions and their corresponding tuple-based labels; an automated process that can perform the task defined in the previous section or, in a more advanced and handy way, a technique or approach that is capable of finding such an automation automatically; and a quantitative standard for performance assessment for the developed automations upon an established dataset. Below, we discuss each of these issues in detail in dedicated sections, including this study's plan and current ongoing effort.

3.1 Dataset

First, before being able to develop an approach and measure its performance, an essential foundation is a collection comprising a sufficient number of problem instances and their corresponding labeling (i.e., answers). In this case, the instances and labels are natural language service descriptions and their tuple-based elements, respectively. Because we are the first to identify and investigate this research problem, to our knowledge, an available and valid dataset matching this study's goal does not exist before us; thus, we must build one. Regarding its construction, there are several considerations for the effectiveness, usability, and representativeness of such a dataset, including its scale, diversity, type of description, and labeling.

The first two concerns, each of which can also be viewed and used as a (quantitative) criterion of dataset, are critical to the development of an automation for the defined problem. As discussed in the next section, if such an automation is manually created, which we consider is a relatively primitive and inefficient way of solving the problem, then a small- or middle-scale dataset with at most hundreds of instances may be sufficient and more appropriate because manually observing and inducting from more cases probably would overload human developers. Thus, diversity, such as the variation of the structures and phrasing of natural language sentences, would be more critical and important for an automation's generality rather than quantity. However, if a meta-automation is used for the generation of automations, such as a machine learning-based technique or method, then it is likely that both the scale and diversity of the dataset matter. For example, due to the enormous number of neurons (weight values) comprised in their neural networks, deep learning (DL) techniques usually require tens of thousands, even millions, training cases (learning samples) to work. Overall, for both approach development and performance evaluation, we consider these two properties of the dataset as the higher the better.

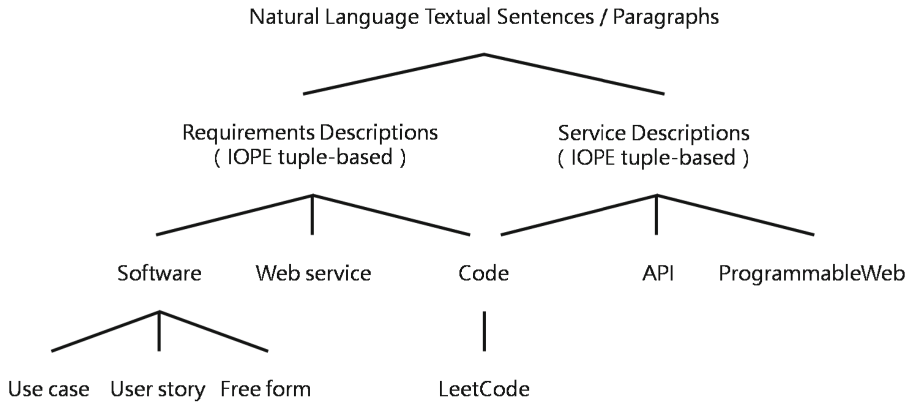
Natural Language Textual Sentences / Paragraphs
Requirements Descriptions (IOPE tuple-based)
Service Descriptions (IOPE tuple-based)
Software
Web service
Code
API
ProgrammableWeb
Use case
User story
Free form
LeetCode

Fig. 2. Alternative sources of the planned under-construction service description dataset.

As mentioned in the Introduction, ASC involves three types of service descriptions. However, only two of them likely must be processed for ASC approaches and are therefore studied with this research problem. Also, we consider the descriptions of service/composition requests (requirements) and component services because these two types of information are indispensable when an ASC approach works to compose services. For the gathering and collection of such service descriptions, existing ASC papers and their illustrating composition examples can be a source, including both their demonstrated composition requests and component service specifications. However, insufficient quantities have been found to be a problem [5] (i.e., a threat to both diversity and, primarily, scale) because most ASC studies only present a few (typically only one) illustrations of composition upon a small set of component services. Thus, certainly, valid alternatives must be sought and used. Specifically, considering the two concerned properties, for service demands (i.e., composition requests), we plan to use existing software requirement datasets as an additional supplement because they are similar in essence (service/composition requests are also a type of software requirement). Regarding software requirements, a review of the literature and this study's data collection showed that such datasets can be written down and expressed in different formations in software engineering, including use case scenarios in UML, user stories in agile methods, and free-form texts (i.e., texts with specific restrictions/templates) [2], as introduced in Sect. 1. For both dataset diversity and approach generality, we consider that requirements in these paradigms (and the others that are also commonly used in software engineering practice but not mentioned in this study) should all be considered, included, and addressed properly, as has been done in some previous studies for the automated extraction and generation of graphical software (e.g., domain [2]) models from various forms of (i.e., unrestricted) natural language software requirements. When using a developed approach, no assumptions about the syntax and structure of its processable descriptions must be made. Finally, we discuss one more provider (alternative) of service demands in the next paragraph, *Leetcode*, which has been seen and used in a study as a massive collection of intended software functionalities. Conversely, regarding the descriptions of component services, current service repositories (e.g., *ProgrammableWeb*) and the textual explanations of their registered services are a straightforward and ample origin of this type of information. In addition, we consider that program API documents may be another abundant source for the proposed dataset construction because, as human-written descriptions of component services, these documents are also natural language declarations of (program) components in nature. The entire idea and big picture of the proposed planned under-construction service description dataset are shown in Fig. 2. Eventually, with a dataset built in this manner with the above roots, we believe that an adequate scale, diversity, and types of service descriptions for future research can be reached.

In a preliminary study [5], an extraction of the *Leetcode* problem set consisting of approximately 160 functionality statements and 10 composition requests from the ASC literature are used for both approach development and performance evaluation. However, apparently, this aggregation is deficient in all the aspects discussed previously (i.e., diversity, scale, and type). Thus, as a remedy and enhancement, the establishment of a complete service description dataset obeying prior ideas is ongoing. While the tagging of the input and output elements (IO tuples) for the functionality demands contained in the aggregation has been manually performed, we consider that two disparate perspectives regarding data labeling are worth discussing in more detail, including the detailedness of data labeling and untagged data. First, in the above data aggregation, the tokens (words and symbols) of sentences are marked in binary, and thus, a token is either an extraction target (i.e., part of an intended input/output element) or not. As an example, *this/0 service/0 generates/0 the/1 location/1 of/1 customer/1 (or this service generates (the location of customer)*$_{TargetOutput}$*)*, for which the result of extraction should be like $O =$ *(the location of customer).* However, this method lacks specifics and distinctions, which we consider might be a disadvantage or even an obstacle, particularly when using machine learning-based solutions, such as those using conditional random fields (CRFs), which we discuss and explain in the next section, because they provide users with insufficient details and poor variation. For example, for CRF, this leads to fewer distinguishable and definable features that likely would deteriorate the extraction result and performance. To investigate and solve the proposed problem more completely, more detailed and advanced remarking of problem instances could be helpful and useful, such as *this service generates* $((\textit{the location})_{CoreOutput}\ (\textit{of customer})_{AdjectivePhrase})_{EntireOutput}$. However, this process certainly increases the cost of dataset establishment and tagging that we discuss below for the second perspective. As anticipated, manual labeling of a dataset is time-consuming, tedious, and error-prone, particularly when its scale is large, as is required in the proposed study. However, unfortunately, carefully reviewing and tagging data is necessary when referred to as answers and supervised learning. To solve this issue, cutting-edge machine/deep learning technologies, such as pretrained models, which required training with large untagged data; and fine-tuning, which adjusts a pretrained model with a small amount of mission-specific labeled data, may be worth trying and investigating for this study and application.

3.2 Approach

Regarding the resolution of this research problem, because it is basically a case and application of natural language processing (NLP) involving information extraction (e.g., IO tuple elements) from human-written (service) descriptions, no matter what specific (paradigm of) solution is used, a reliable NLP tool is indispensable for parsing and providing essential linguistic knowledge for a subsequent solution (either an automation or meta-automation) to perform as expected. Thus far, *Stanford CoreNLP* [6] has been widely recognized as the best performing of such NLP tools. Therefore, this study uses this tool as the foundation and provider of linguistic information for the proposed planned and developed approaches, as shown at the top of Fig. 3.

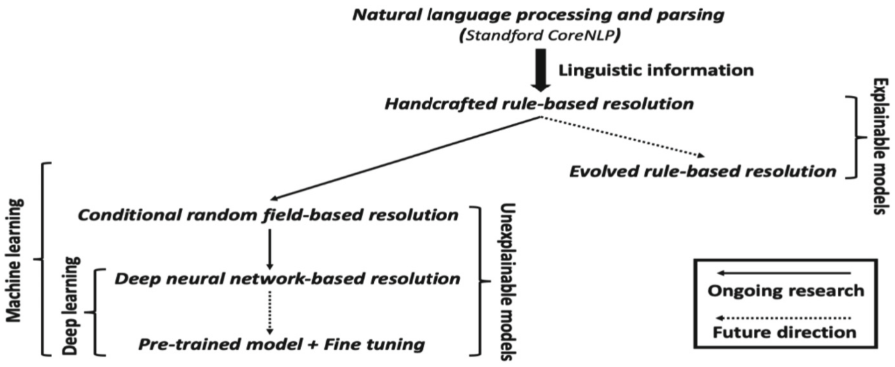
Natural language processing and parsing
(Standford CoreNLP)
Linguistic information
Handcrafted rule-based resolution
Evolved rule-based resolution
Explainable models
Machine learning
Conditional random field-based resolution
Deep learning
Deep neural network-based resolution
Pre-trained model + Fine tuning
Unexplainable models
Ongoing research
Future direction

Fig. 3. Proposed research plan with an increasing level of automation.

The remainder of Fig. 3 shows a path that visualizes the proposed study plan. As explained, the proposed ultimate goal aims to eliminate, as much as possible, human intervention in the creation of a method or model for the problem defined in Sect. 2 so that a true automation (truly automated solution) with a minimized manual effort can be obtained. However, such a goal is quite challenging and not instantly reachable due to the complexity and difficulty of the targeted problem. Thus, when developing the proposed method, we plan to explore the proposed problem step by step along a route of increasing level of automation to have an insightful understanding and comprehensive solving (studying) of the problem, as shown in Fig. 3 and as described in detail below. The proposed research plan can be discussed and viewed as having several disparate dimensions, which are the method of production (manual, semiautomatic, and automatic), form (representation, such as *if-then-else*-based rules or neural networks), and understandability (explanability and/or interpretability) of a generated model.

For this targeted problem, [5] is the proposed first published research work, which can be characterized as, in terms of the three considered dimensions, a manually developed, NLP extraction rule-based, and human-understandable-model approach. In this research, as an initial study, the most straightforward and basic (primitive) paradigm for the resolution and addressing of such NLP applications has been used to preliminarily investigate this problem. More specifically, after an iterative and exhaustive human observation and analysis of the dataset established in [5], a set of NLP extraction rules were manually and incrementally developed and verified by the authors of [5]. Based on the experimental results of this preliminary study, the accuracy and performance of transforming functionality descriptions into their corresponding tuple-based elements is acceptable. However, the biggest weakness (also the deficiency of this type of solution) of this process is that the observation, analysis, enactment, and implementation of these NLP extraction rules is both time-consuming and labor-demanding, which we consider can and should be improved due to its inefficiency and heavy cost. To address this issue and research direction, we decide to take advantage of machine learning techniques, as shown in the two branches of Fig. 3.

Before diving into the left branch of Fig. 3, we first discuss the figure's smaller right branch. Considering the third concerned dimension (i.e., the understandability of a model), although the machine learning-based approaches on the left branch could significantly decrease (CRF) or even entirely eliminate (DL technology) human labor and intervention in the production of an intended model, unlike handcrafted NLP extraction rules, such as those proposed in [5], common machine learning models are hard to understand and explain by humans. To address this concern, as indicated in the right branch of Fig. 3, it is possible to automatically (genetically) evolve and generate a set of NLP processing rules for a problem or application, as has been tried and demonstrated in [7], where the advantages, benefits, usefulness, and applications of highly explainable and human-readable rule-based models can be found and seen. Thus, with a well-labeled dataset, theoretically human-understandable NLP extraction rules for the problem can be generated and obtained without manual effort, which we consider the best case on the two proposed dimensions (i.e., the first and last ones) and, thus, is worth trying and investigating (i.e., one of the proposed future research directions).

Conversely, along the left branch of Fig. 3, with a rising extent of automation, a planned track of solutions is shown and includes three separate stages/approaches. However, instead of the conditional random field (CRF), which appeared first on the branch, after [5], we first used the neural network models of DL to address the problem due to its powerful capability and many successful applications reported. Following the order of the left branch, we consider CRF to have a lower level of automation compared with DL because it requires its users to intervene to design and define a set of suitable feature functions before learning and running. However, with the small dataset built and used in [5], the evaluation of the generated neural network models discloses that they are nearly perfect in training performance (nearly 100 percent accurate) but poor in testing with unseen cases, which suggests that overfitting has occurred. We believe that this overfitting phenomenon is probably caused by the two contrary circumstances, the large number of variables (i.e., the weight values of neurons, usually hundreds of thousands of them at least) contained in a deep neural network model and the small dataset consisting of fewer than 200 problem instances for learning and fitting (i.e., an underdetermined system). Thus, before a large-scale dataset described in Section III A is available for the development and investigation of a DL-based solution, we decide to take one step back in terms of the level of automation and study the application of CRF to the proposed problem, as described in the next paragraph.

As with most techniques and models, CRF has advantages and disadvantages. Compared with DL networks, one of the advantages of CRF is that depending on the designed and used feature functions (the number of considered feature instances), its number of (weight) parameters that must be fitted and adjusted with training materials is typically much lower. Thus, in theory, unlike DL techniques and models that frequently require at least tens of thousands of instances and their tags to work due to their enormous neurons, CRF takes fewer cases to learn and fit, which matches the proposed current research circumstance and limitation that only a small dataset is available. However, this advantage of CRF also comes with a cost of poorer automation because for each specific problem, a set of appropriate feature functions must be manually identified and formulated in advance. Thus far, the ongoing and unpublished research of the authors that uses CRF can reach the same level of extraction performance as [5]. However, in [5], we spend dozens, even a few hundreds, of hours on developing its NLP extraction rules and approach, while by contrast, it takes a shorter time to identify a set of valid feature functions of CRF. More specifically, compared to the handcrafted rules-based approach in [5], CRF performs better in both the efficiency of approach development and the extent of automation, but one of its disadvantages is that even with much fewer model parameters, a CRF model is still not easy to explain and understand by humans. Overall, we consider that the model understandability of CRF is worse than that of handcrafted rules and better than that of DL networks. Regarding the level of automation, however, CRF is higher than handcrafted rules and lower than deep learning networks. Finally, regarding CRF, we are still working on the identification and formulation of its more efficient and important feature functions for the proposed application problem so that better performance and further insight can be obtained.

As shown in Fig. 3, the proposed next stage of approach development goes into deep learning. As discussed before, we have already tried and applied a DL neural network to address the proposed problem, although this preliminary trial presents a problematic result that suggests that more effort and deeper investigation must be performed. Regarding its technical details, in this initial study of DL for the proposed problem, we use the long short-term memory (LSTM) model, which is a type of recurrent neural network (RNN) in DL, because it can process both single data points and entire sequences of data (e.g., the dependencies across the disparate points of a data sequence), which is an important property that is naturally and intensively required in the proposed application and research because human sentences and descriptions, in essence, are sequences of words and phrases (units) that, except for the individual units, their entire context and the relationships between them must also be considered and handled as a whole. In addition, LSTM has also been successfully applied in many other NLP tasks and sequence labeling/learning applications, which is another reason for its adoption. Regarding the improvement and correction of this current failure of LSTM on the problem, as mentioned before, we consider that this result is probably caused by an overfitting LSTM model, for which three different solutions and potential directions have been planned or studied: (1) a larger dataset offering a greater number of training/learning cases, as discussed in Sect. 3. A; (2) decreasing the complexity of the model or adopting a model with a lower complexity (e.g., the proposed employment and trial with CRF); and (3) reducing the noise among data (i.e., denoising). Concerning the third point, because the currently available dataset comprises merely a small number of problem instances for learning and training, any type of noise likely causes severe overfitting and negatively affects the proposed approach and the resulting DL models. Thus, the identification and handling of the noises within the dataset would probably be necessary. For example, the replacement of a set of semantically or functionally equivalent words and phrases with a general term during the preprocessing stage could be helpful in avoiding overfitting.

Last, as an advanced study of the application of DL to solve the proposed problem, due to its recent popularity and success in various NLP tasks and applications [8], we plan to take advantage of pretrained models and their fine-tuning. More specifically, instead of training and fitting a DL model from scratch for a specific mission, such as what we have done before with LSTM, pretrained large-scale NLP neural models, such as BERT and GPT, could be used for the proposed problem. In this case, if applicable and useful, we no longer must work to generate a mission/task-specific DL model from scratch, which is both costly and time-consuming. Also, the gathering and labeling of a large-scale dataset for the proposed problem, which is also expensive and typically requires long computation times, such as that described in Sect. 3.1, likely would become unnecessary and avoidable. Regarding the fine-tuning of a pretrained model for the proposed purpose and application, because this training (transferring) takes only a relatively smaller mission/task-specific dataset, the proposed current data collection established and used in [5] might be instantly and directly applicable and triable. Overall, we consider that this study and application will likely markedly benefit from these cutting-edge DL technologies, as analyzed above. Thus, as drawn in the last stage of the left branch of Fig. 3, they are worth trying and investigating.

3.3 Evaluation

For this research subject, the criteria of the proposed assessment incorporate both time and accuracy. Regarding time, two different stages must be considered and measured separately, how long it takes to automate the problem (e.g., developing a set of extraction rules or fitting a CRF/DL model) and to generate a set of adequate tuple elements using a specific automation (a set of developed rules or a fitted CRF/DL model), respectively. At the first stage, the current observation is that the higher the level of automation is, the shorter the time taken to achieve automation. As mentioned previously, it takes a lot of time (e.g., dozens of hours) to observe and develop the rule-based approach presented in [5], while the training and generation of a CRF/DL model takes a much shorter duration.

Considering the pure model generation (training) time of ML, CRF is actually faster than LSTM (i.e., DL model) according to the proposed experiments because the former approach has fewer parameters (weight values) to adjust and fit. As explained before, however, CRF requires additional time for human involvement because it requires its human developers/users to manually identify and design a set of feature functions before running, which is typically time-consuming. Applying a pretrained model is expected to be more efficient in time because rather than creating a new DL model from scratch, only fine-tuning is required, which explains why a manner with superior automation is preferably wanted in this study. Regarding the second stage, we consider it unimportant because, regardless of which solution is used, it would not take too long to produce a result (i.e., much faster than the speed of human processing and extraction, at least), even with a DL model containing millions of neurons (or a pretrained model reaching a billion-level).

The second criterion, which is likely the primary concern of the performance evaluation of this research problem, measures both the correctness and completeness of the produced tuple elements with their corresponding ground truth data (i.e., labeled answers). Overall, this type of evaluation, which is common and regularly used in information retrieval and extraction, consists of two consecutive phases. First, a confusion/error matrix including four separate categories – true positive (TP), false-positive (FP), true negative (TN), and false-negative (FN) – must be counted and acquired. Then, based on the values of these four cases, we can easily calculate and obtain ultimate performance metrics (indicators), such as accuracy (TP + TN/TP + TN + FP + FN), precision (TP/TP + FP), recall (TP/TP + FN), and F1 score, as done in [5]. However, there is a potential problem found during the proposed evaluation. As an example, for a service description *with a customer ID, this service generates the public information of this customer, and the correct and complete element of the output tuple should be the public information of this customer*. However, in reality, different approaches may give different results with different levels of completeness (or correctness), such as *the information*, *the public information*, and *the information of this customer*. Without a common standard or consensus, different studies may treat and consider these incomplete outcomes differently (e.g., how to punish and count an incomplete element when making a confusion/error matrix and to what extent for each possible case), which would produce inconsistent scores, makes different studies incomparable with each other and is why we want to develop a utility providing, with the planned dataset, an impartial and handy performance evaluation for this research subject, as explained in detail below.

The last thing about the performance evaluation of this research topic is that as a public, large-scale dataset described in Sect. 3.1, a utility for making and facilitating a consistent, convenient, and quick performance assessment for researchers working on this problem is an important future goal. With the proposed planned dataset and this utility, the developers of different approaches can quickly and accurately calculate and understand the quantitative performance of their work, making precise and fair comparisons with others.

4 Conclusion

In this paper, we first introduce manual service composition and its automation, automatic service composition. Then, based on this introduction, we compare and describe the major differences between manual and automatic service composition, which may be a threat to automation due to the necessity of human intervention and processing, the formations of their used descriptions of services. After the targeted research issue has been defined and explained in detail, the proposed current achievement and future plan on the issues are presented exhaustively, including the dataset, approach, and evaluation of the issue.

During the proposed collection and observation of the current service descriptions and their alternatives, an important issue is identified: the desired targets, such as the input elements of a service, do not always explicitly appear in the text of its depiction; they can be considered and called implicit elements. More specifically, a software component (e.g., a Web API/service) can be explained and described from disparate perspectives, in different ways, and at unidentical levels of detail. For example, both the sentence *this hotel searching service takes a specific location as its input and generates a list of candidate hotels as output for its user and the statement this service provides a hotel searching functionality* could be describing the same service. However, the approaches introduced and planned in this paper can only process and obtain the targeted tuple elements from the former description and cannot address the second statement because the targets are not literally and explicitly contained in the statement. With sufficient domain knowledge and development experience, we believe that human engineers and developers can comprehend and manage both types of component/service descriptions adequately, manually determining what material is required (input) and what would be produced (output) with a service offering a hotel searching functionality. However, with the automations discussed in Section III B, it is impossible to extract and identify elements from a text that contains none of them. Such limitations and the inability to implicit elements, which can be processed, generated, and added by human engineers and developers with proper domain experience and knowledge, can be found in many past studies, such as [9], where only the concepts and their relationships that explicitly appear in the stated requirements will be extracted for the automatic construction and creation of a corresponding conceptual model. Because a large proportion of the surveyed and collected service descriptions and functionality statements are in this style (the majority of them, actually), we consider it a more challenging problem and advanced issue that must be deeply considered and properly addressed in the future for a better applicability and usefulness of the proposed solution.

Acknowledgment. This paper was partially sponsored by the Ministry of Science and Technology (Taiwan) under Grants Most111-2222-E-152-001-MY3 and Most110-2221-E-001-006-MY2.

References

1. Sommerville, I.: Software Engineering, 9/E. Addison-Wesley, Boston (2010)
2. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting domain models from natural-language requirements: approach and industrial evaluation. In: Presented at the Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France (2016). https://doi.org/10.1145/2976767.2976769
3. FanJiang, Y.-Y., Syu, Y.: Semantic-based automatic service composition with functional and non-functional requirements in design time: a genetic algorithm approach. Inf. Softw. Technol. **56**(3), 352–373 (2014). https://doi.org/10.1016/j.infsof.2013.12.001
4. Fanjiang, Y.-Y., Syu, Y., Ma, S.-P., Kuo, J.-Y.: An overview and classification of service description approaches in automated service composition research. IEEE Trans. Serv. Comput. **10**(2), 176–189 (2017)
5. 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.) SCC 2021. LNCS, vol. 12995, pp. 1–17. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-96566-2_1
6. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60 (2014)
7. Shahrzad, H., Hodjat, B., Miikkulainen, R.: Evolving explainable rule sets. Presented at the Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, Massachusetts (2022). https://doi.org/10.1145/3520304.3534023
8. Han, X., et al.: Pre-trained models: past, present and future. AI Open **2**, 225–250 (2021). https://doi.org/10.1016/j.aiopen.2021.08.002
9. Vidya Sagar, V.B.R., Abirami, S.: Conceptual modeling of natural language functional requirements. J. Syst. Softw. **88**, 25–41 (2014). https://doi.org/10.1016/j.jss.2013.08.036