electronics

MDPI

Article

Extensible Chatbot Architecture Using Metamodels of Natural Language Understanding

Rade Matic [1,†], Milos Kabiljo [1,2,†], Miodrag Zivkovic [2,*,†] and Milan Cabarkapa [3,*,†]

[1] Department for Information Systems and Technologies, Belgrade Academy for Business and Arts Applied Studies, Kraljice Marije 73, 11000 Belgrade, Serbia; rade.matic@bpa.edu.rs (R.M.); milos.kabiljo@bpa.edu.rs (M.K.)
[2] Faculty of Informatics and Computing, Singidunum University, Danijelova 32, 11000 Belgrade, Serbia
[3] School of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, 11000 Belgrade, Serbia
* Correspondence: mzivkovic@singidunum.ac.rs (M.Z.); cabmilan@etf.bg.ac.rs (M.C.)
† These authors contributed equally to this work.

Abstract: In recent years, gradual improvements in communication and connectivity technologies have enabled new technical possibilities for the adoption of chatbots across diverse sectors such as customer services, trade, and marketing. The chatbot is a platform that uses natural language processing, a subset of artificial intelligence, to find the right answer to all users' questions and solve their problems. Advanced chatbot architecture that is extensible, scalable, and supports different services for natural language understanding (NLU) and communication channels for interactions of users has been proposed. The paper describes overall chatbot architecture and provides corresponding metamodels as well as rules for mapping between the proposed and two commonly used NLU metamodels. The proposed architecture could be easily extended with new NLU services and communication channels. Finally, two implementations of the proposed chatbot architecture are briefly demonstrated in the case study of "ADA" and "COVID-19 Info Serbia".

Keywords: chatbot; extensible architecture; metamodel; natural language understanding; framework; COVID-19

check for updates

Citation: Matic, R.; Kabiljo, M.; Zivkovic, M.; Cabarkapa, M. Extensible Chatbot Architecture Using Metamodels of Natural Language Understanding. *Electronics* **2021**, *10*, 2300. https://doi.org/10.3390/electronics10182300

Academic Editors: Cataldo Musto and George A. Papakostas

Received: 22 August 2021
Accepted: 16 September 2021
Published: 18 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Social, Mobile, Analytics, Cloud, and Internet of Things (SMACIT) technologies and advances in the field of artificial intelligence (AI) have fundamentally challenged the way people are working, doing business, or communicating with each other. A typical example of an AI system and one of the most elementary and widespread examples of intelligent Human–Computer Interaction (HCI) is a chatbot [1]. The chatbot is a computer program that simulates human conversation or chat, using or not using artificial intelligence, and it participates in a dialog with a human using natural language [2]. Chatbots have many advantages for users and developers, and their popularity is rising. Most implementations are instantly available to users without needing installations. Chatbots can provide users with quick and convenient support responding specifically to their questions, and this is the reason why the most frequent motivation for chatbot use productivity, while other motives are entertaining, is social factors, good and up-to-date information and contact with novelty. Rather than creating a human-like smart machine application, it is about creating effective digital assistants who can provide information, answer questions, discuss a specific topic, or perform a task [3].

Today, a chatbot can perform many functions of mobile applications or websites, all within conversation via communication applications, without requiring the user to install or download new applications. Chatbots are now incorporated in popular digital assistants such as Siri, Cortana, Alexa, Assistant, etc. There is no longer a need to download, install, or open applications only for performing operations such as ordering a product. It is just another aspect of an already-connected multitasking world.

The use of chatbots evolved rapidly in numerous fields in recent years, including customer services, e-commerce, marketing, supporting systems, education, healthcare, cultural heritage, and entertainment. In all of these fields, chatbots have proven useful in various contexts to automate tasks and improve the user experience. Additional predictions say that by 2022, 80% of companies will use chatbots, and banks will be able to automate up to 90% of their customer interaction with them [4]. The global chatbot market is projected to reach 2 billion USD by 2024, growing at a CAGR (compound annual growth rate) of 29.7% [5].

A chatbot platform must have the following three parts that really add conversational experience:

- Natural Language Processing (NLU): understanding user input and extracting relevant information.
- Conversation flow: including managing the context of the conversation.
- Action Fulfilments: used to represent simple responses, as well as advanced features such as database querying, Application Programming Interface (API) requests, or a custom logic trigger.

The NLU is emphasized in this paper as the heart of every chatbot. Extensibility is the capacity of software architectures to glue external architectures (leaves) to their core structure, creating a synergy between these dissimilar architectures [6]. In this paper, the advanced extensible microservice chatbot architecture has been proposed, allowing easy switching from one NLU provider to another (e.g., because of better support for the Serbian language or due to cost issues). The proposed object-oriented framework called Weaver [7] is designed to be agile to tackle an incredibly variable environment of modern communication channels for interactions of users. The novelties of our approach are:

- The proposed chatbot architecture is extensible, supporting new natural language understanding services and communication channels for user interactions. Thanks to this, we resolve the independence of chatbot frameworks from all market vendors dealing with natural language understanding and communication channels.
- The solution was implemented in two case studies on the Serbian language. As far as we know, no one has dealt with the Serbian language using different NLUs.
- It relies on so-called metamodels as the main extensible mechanism in Weaver. For this mechanism to succeed, we provide:
  - A general NLU metamodel (model of a model).
  - Metamodels for the two specific NLU services (Dialogflow and RASA).
  - Corresponding metamodels as well as rules for a mapping between generic and specific NLU metamodels.

Due to everything specified above, we think that extensible and scalable architecture is required for natural language understanding that solves some of the problems defined above. It is much easier to make metamodels of each NLU service and map them to our general metamodel, and thus go beyond manual programming for each new NLU service. Designers define all these metamodels and corresponding mapping rules in the database at design time. Using mapping rules, we automatically create, maintain, and forward objects with all the necessary data to the desired NLU metamodels (NLU services) or communication metamodels (channels).

This paper is organized as follows. Section 2 brings motivation for this paper as well as related work relevant to the topic. Section 3 presents the NLU support. In general, logical architecture has been presented with the detailed specification of the components of the proposed architecture. Metamodels and their mapping rules to ensure independence from one NLU service are presented in Section 5. The details exposed on the NLU Metamodels and corresponding mapping rules are explained with a specific example in Section 6. Section 7 explains the use of the proposed framework in two case studies: ADA and COVID-19 Info Serbia. Finally, the conclusion is derived in Section 8.

2. Related Work and Motivation

The introduction of chatbot technology started in 1966 with the computer program known as ELIZA [8]. ELIZA was able to mimic human conversation by trying to answer a user question by matching scripted answers, i.e., by performing simple sample matching. Developed by Joseph Weizenbaum in 1956 [9], it was designed to emulate a psychotherapist and had a knowledge base in this domain. In 1995, the chatbot ALICE (Artificial Linguistic Internet Computer Entity) was developed. ALICE relies on a simple pattern-matching algorithm with the underlying intelligence based on the Artificial Intelligence Markup Language (AIML) [10], which makes it possible for developers to define the building blocks of the chatbot knowledge [11]. In scientific literature, academic articles, and conference proceedings, different aspects of chatbot technology and applications have been analyzed [12,13].

Within the study of chatbot technologies, a special direction of interest is related to the chatbot modeling framework. Frameworks of the existing chatbots provide specific chatbot architectures and tools. Some are very simple, and others use web services' massive linguistic datasets, which have very precise models obtained owing to many years of experience and vast amounts of data. The most popular ones include Dialogflow, RASA, Microsoft Bot Framework, Botkit, Pandorabot, and WIT.ai. The chatbot framework has been mostly accomplished by defining entities, intentions, and responses within a specific platform, providing interface and great natural language understanding capacities. Authors Mu and Sarkar [14] discuss some drawbacks of NLU and find that restricted natural language systems might perform better than full natural language. According to this, the proposed model in this paper is based on the closed domain.

A great study about kinds of chatbot architectures was written in [13]. In this and other similar papers [3,15–25], no one explains how to overcome the problem of integration with different NLU services or with different communication platforms. Several approaches have been proposed to simplify the process of chatbot development and to improve their response mechanism [15,26,27]. It is very difficult to integrate chatbot architecture with other external services without manual coding, and designers with new communication platforms or NLU services cannot easily extend it [15]. Most chatbot architectures just offer an API to query the results of the intent and integrate with the NLU service. Similarly, although each of them supports different communication channels, they usually do not offer any extension options. Weaver is designed to solve these problems.

Xatkit [26] (formerly known as Jarvis [15]) and Conga [27] are partly similar works as ours. They use a model-driven architecture (MDA) for developing chatbots. Although our approaches and chatbot architectures are different from these two papers, they show realization of the idea of developing chatbots that are independent of NLU services. However, we do not use MDA to develop chatbots like them, but we use one of the basic concepts of the MDA approach (models). To avoid vendor dependency, our chatbot architecture consists of metamodels for NLU services, and they map to our general NLU metamodel-making mapping rules. You will notice that we only use NLU, which is one part of Natural Language Processing (NLP), and we have a different logic of the context usage and execution action. The metamodels of Xatkit and Conga are used for different purposes than ours, because they include concepts such as context, action, platforms, etc. We use a simpler technique to provide extensible NLU-independent chatbot architecture. In our work, all the logic related to the context affecting the flow conversation or the execution of a certain action is found in the components that will be explained in Section 4.2. However, except for a detailed explanation of microservice chatbot architecture, the main contribution of this paper is easy integration with NLU external services because these are probably the most important parts of chatbot architecture that are difficult to implement on your own. Therefore, it is good to use external services. However, this does not rule out the possibility of creating our own NLU service. It can be integrated in the same easy way as shown here in the paper without manual programming, thanks to metamodels and proposed mapping rules.

Xatkit introduces a multi-platform chatbot modeling framework using metamodels and textual Domain Specific Language (DSL). This framework decouples the chatbot modeling part of the platform-specific aspects, increasing the reusability. DSL provides primitives to design the user intentions, execution logic, and deployment platform. They currently only support Dialogflow as an NLU service. Our implementation has shown independence from Dialogflow and Rasa NLU services. In our approach, we have conditional branching, and we support generic events (webhooks) that allow us to make a reactive chatbot that can actively listen and respond. Our framework enhances extensibility and customizability by providing explicit webhook methods and architecture elements that allow it to extend its stable interface. Differently from us, their metamodel of the Intent Package contains a collection context. On the other hand, we do not see on their metamodel how they annotate entity roles and entity groupings. With entity roles, we can define entities with specific roles in utterance. Entity groupings allow entities to be grouped together with a specific group label defining different orders. Conga presents a model-driven solution for forward and backward chatbot engineering, featuring a recommender system that assists in selecting the most suitable chatbot development tools. It comprises a neutral metamodel and a DSL for code generation and parsers for several chatbot platforms. The main and most important difference is that our defined chatbots are executable by providing an execution engine. Conga also does not support platform-specific concepts, such as buttons action. Although their entire metamodel also lacks role and group elements, it is difficult to compare the rest of the metamodel because the purposes of their metamodels are different from ours. Although MDA has many advantages in software development, such development is often limited by the kind of tool they use. Conga already has 15 generation tools. They are only flexible in the parts of the framework covered by the used DSL. Using the metamodel frameworks proposed in our model is extensible and adaptable enough to include new NLU services and communication channels. Weaver provides interactive learning that makes it easier for developers when it comes to developing conversational flows, but this is out of the scope of this paper.

3. NLU Support

NLU is a subset of NLP and conversational AI. It helps computers to understand human language by understanding, analyzing, and interpreting basic message or speech parts. NLU is trained with natural user utterances tagged with entities and outspread with the use of synonyms. The concept of utilizing the AI and human language to perform the communication with machines is not new; however, researchers have underestimated the complexity of human languages for years, even decades. Even nowadays, after recent advances in AI and NLP, several studies have been conducted that show that users cannot be tricked and they always will know that they are communicating with a computer rather than another human being [28]. Recently, AI-driven chatbots have been employed for various tasks from different application domains, such as medicine and self-diagnostics [29] and university student service support [30,31].

In the first version of our Weaver platform, we chose Dialogflow as the NLU service. When we saw some shortcomings, we introduced RASA as the most trustworthy open-source NLU service in terms of confidence score [32]. After that, we realized the problem of NLU dependency and then came up with the idea we have implemented in our architecture. Dialogflow and RASA are two different NLU services with the same purpose. Every new NLU service would be similarly applied in our overall proposed solution. We will explain their differences and the reasons for choosing them as example for our architecture.

Both NLU services use ML-based NLU, and they are trained with natural user utterances tagged with entities. Dialogflow has a validation, indicating that there is a specific intent that needs more training data. It also has been provided with system entities such as numerics, date, time etc. RASA, on the other hand, has a custom pipeline selection based on the amount of training data. We can use the pre-trained model and using more data, we can train from scratch by choosing a pipeline. RASA provides functionality to

evaluate intents and entities. RASA also has a training data importer option to import data in different formats and from different sources. In Dialogflow, we do not have the option to import or use training data, but it accepts one line per utterance as training data. Both platforms provide multi-language support. To support the Serbian language in Dialogflow, we must add Google Translator to our architecture. RASA allows us to add this support with the help of spaCy as inbuilt components. For languages which does not have pre-trained word embeddings, RASA suggests a pipeline without the spaCy library. Pre-trained word embeddings are helpful as they already encode linguistic knowledge. For most used languages, it is a very helpful library because it can "understand" large volumes of text used to build natural language understanding systems. Unfortunately, for Serbian, there are no pre-trained word embeddings, and this is reason why we did not use the spaCy library. To support Serbian, we used an adjustable NLU pipeline. The pipeline consists of different components working sequentially to process user input into a structured output. There are components for intent classification, for entity recognition, for tokenization, pre-processing, etc. Each component processes an input and/or creates an output. The order of the components is specified in a configuration file. We tried multiple configuration files with the RASA test. To get the most out of our training data, we trained and evaluated our model on different pipelines and different amounts of data. Repeating this process with different percentages of training data, we tried to understand how each pipeline behaved by increasing the amount of training data. The whole process was performed five times for each specified configuration.

RASA is open source, where you can use it without any costs. RASA can be deployed on-prem and on the cloud. Dialogflow stores and deploy models in Google cloud. RASA has added Bidirectional Encoder Representation from Transformer into the pipeline, helping to create better models. RASA also provides an additional advantage of adding your own custom model into the pipeline for any task. Dialogflow is a great platform with good models and pre-trained entities, but it does not support custom models. RASA also has added support for Tensorboard 2, used to visualize training metrics. It helps to understand if the model has been trained properly, and we can make changes to hyperparameters based on the metrics to improve model.

4. Proposed Chatbot Architecture

The logical architecture of the proposed solution, called Weaver, is based on the reference chatbot model proposed by the authors E. Adamopoulou and L. Moussiades [33]. This architecture can be used for developing a broad range of chatbots. It enables the understanding, implementation, maintenance, and further development of a platform-independent chatbot.

In this paper, we describe our approach of building a chatbot based on AI. Generally, there are two types of models prevailing in chatbot development based on AI [13]:

- Retrieval-Based Models: these use a repository of predefined responses or use some type of heuristics to choose the adequate answer based on the inquiry and context. They can provide more reliable and more grammatically accurate answers. They are easier to teach as they require less information, but they are not able to respond to questions beyond their knowledge base.
- Generation models: these do not rely on predefined answers. They generate new responses from the start as they can respond to ambiguous questions. These chatbots become smarter over time and they can learn from previous questions, responses, and conversations. However, they are hard to train as they demand large amount of data, and retrieval models often enjoy better control over response quality than generative models.

Weaver use retrieval-based conversation. Our retrieval-based model uses NLU techniques to predict the most accurate intent. After that, Weaver takes responses from a closed set of responses using an output ranked list of possible answers. The NLU-independent chatbot architecture is defined through three main parts: (1) a generic metamodel defining the general concepts of NLU and their relationships, (2) a generic metamodel of specific NLU

service (e.g., Dialogflow, RASA), and (3) a set of mapping rules which maps every concept of specific NLU service to concept of generic metamodel. The Weaver platform just follows these mapping rules and then executes the overall chatbot logic.

Here proposed, the conversational AI platform provides a set of analytical features that assist in knowing the number of sessions, intents, number of users, returning users, number of answered and unanswered questions, and the flow of questions. The Weaver administration tool allows us to make chatbot scenarios on our own in easy steps, from defining new scenarios to adding intention and definite answers. We realize efficiencies in the development of chatbot architecture by exploiting the framework tools and modeling facilities. The logical architecture and component structure of the implemented Weaver BotFramework is given next, with detailed explanations of each component.

4.1. Architecture

The logical architecture of the model proposed in this paper is shown in Figure 1. As can be seen, the architecture consists of four main building blocks:

- User.
- Communication platforms (channels) through which users communicate.
- Weaver—Conversational AI platform.
- External services among which NLU is the most important.

The chatbot platform involves BotFramework and two connectors: API messenger connectors and External mapping connectors. Among other things, this platform also serves for connection, construction, testing, and deployment of the intelligent chatbot. Communication between the BotFramework and both connectors (messenger and external) is performed via Hypertext Transfer Protocol Secure (HTTPS). The entire platform is based on microservices. Validation and mapping of messages being received from a communication platform are performed in the API Messenger Connector. This aims to prepare the message, which will be forwarded to the next component depending on the type of message received from the messenger. Each communication platform has its own metamodel and set of mapping rules which are defined in the database and are used by the connector to fill the object with all necessary data, helping it to be independent from just one communication platform.

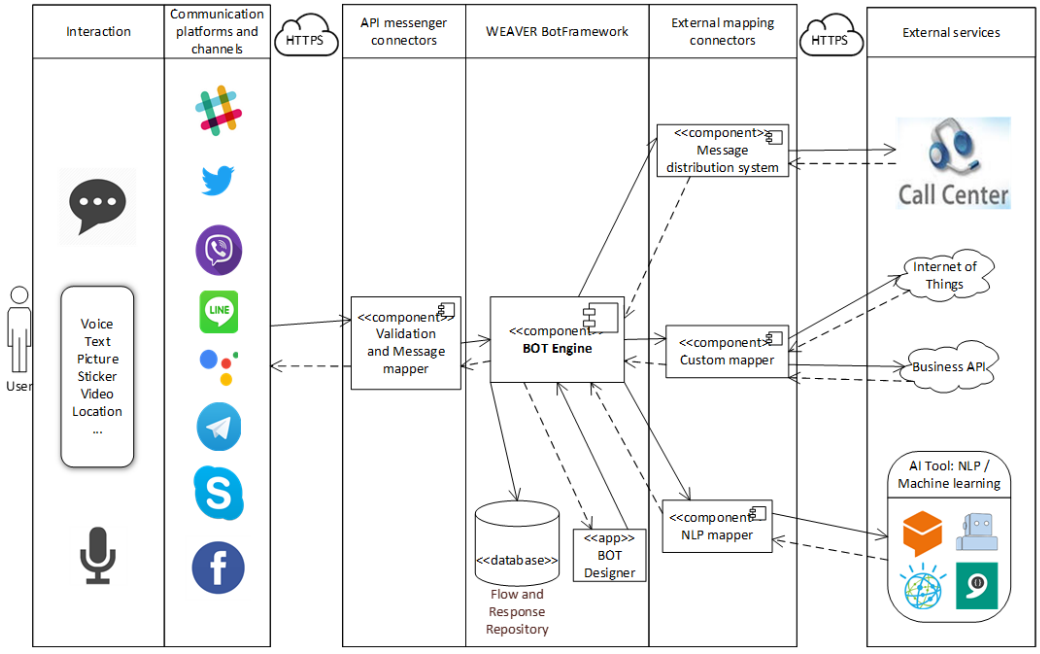

Figure 1. The logical architecture of the proposed solution.

The lifecycle of the message starts and finishes at the connector, since its role is to receive the message and return the message from the chatbot to the channel from which the message was sent. For the message to be received from the communication platform to our system, we must configure the webhook method. We use the messenger connector for that purpose, which contains the defined methods that accept callback by the communication platform. Our messenger connector must first be subscribed to the messaging events of the approved communication platform and set up a callback uniform resource locator (URL). The chatbot platform will be able to receive webhooks sent from the communication platforms. The first function of the connected component is to recognize where the message was sent from, i.e., from which communication platform the event occurred, causing the system to receive the message. This component recognizes it simply through the URL. Based on the URL data, the component recognizes from which communication platform the message was received and retrieves rules for validation and mapping of context that arrived at the request. Based on the rules, the message validity is checked, i.e., whether the message was sent by the communication platform to which we have subscribed or not. If the message is valid, it can be forwarded to the mapping module. In this part, the message is mapped into the object with which the BotFrameworks operates using the metamodel and defined mapping rules. Messenger connectors must ensure proper mapping of all message types for a particular communication platform. Objects forwarded to the BotFramework also contain data of the communication platform and the user, so that the message can be properly forwarded to the right user. The said mapping also works in the opposite direction, i.e., when the BotFramework should return the message towards a certain communication platform. Metamodels and mapping rules of communication platforms are not part of this work; we want to focus on metamodels and mapping rules of NLU services explained in Section 5. The BotFramework consists of many components making this system scalable, and it supports most of the criteria for building a good enterprise framework, suggested in [34]. This framework is the central and main part of the architecture containing the entire logic for processing received messages, works with contexts, chat flow, logical decision-making, integration with predefined scenarios and responses, and various other required rules. It contains several important components such as bot engine, bot designer, language processing, conversation engine, etc. Each received message is processed separately and passes through a series of stages. Each user has their own chat session (conversation) that keeps a record of all messages between users and chatbot. All initiated, suspended, and completed instances of scenarios following their definition of scenarios are also seen here. Therefore, each chat session represents one or more scenario between users and chatbots that contains contexts affecting the current and future chat flow. The bot engine is a complex component that constitutes the central part of the BotFramework and it consists of several minor components that will be detailed in Section 4.2.

External mapping connectors serve for connection with external services, such as NLU services. These connectors also have separate predefined mapping rules for each specific type and implementation. In addition to predefined connectors working with the above-mentioned services, there are also custom connectors for external data necessary in conversation, in which mappings can be changed depending on the need to successfully realize any conversation. External mapping connectors also contain components that expose services to other systems participating in the communication process with the user, e.g., a contact center that could take over user conversation. The BotFramework can be integrated with the existing contact center application with the help of the component Message Distribution Connector. In the case of complex questions, the user is redirected and chats with the operator from the corresponding contact center. The extensibility of the proposed framework simplifies the integration of other useful external services. The BotDesigner is a visual tool equipped with the Software Development Kit (SDK) for the corresponding implementation platform. The SDK offers possibilities that facilitate chatbot–user interactions. Using SDK, it is possible to directly teach the NLU to avoid dependency

on the NLU interface of the vendor. In addition, BotDesigner enables the visual generation of the scenario, i.e., its flow. The flow of conversation maps all potential directions in which the discussion may develop, with many branches for all possibilities. The conversation flow is responsible for predicting all possible input messages and chatbot reactions. This helps users to quickly achieve their goals and to obtain information easier and give priority to business needs. The BotDesigner is intuitive and easy to understand and use. All data necessary for scenario flow management and context-based responses, identified intents, and entities are in the Flow and Response Repository. This database is used for storing all predefined scenarios with their actions and conditions, responses with the necessary segmentation, conversations, users, and logs. It accommodates all settings related to connectors, components, and general system settings. All metamodels and mapping rules are also stored in this database.

4.2. The Component Structure of the Bot Engine

A detailed representation of the Bot engine is shown in Figure 2. It consists of a series of components that are interconnected. The Core Engine (CE) is a separate component that is the central part of this engine. The core engine is a component that manages other components within the BotFramework. It must know the correct sequence of execution of the action, track the execution of actions and errors, send notifications, etc. Depending on the configuration, each action passes through debug and info log to follow the flow of such a complex component.

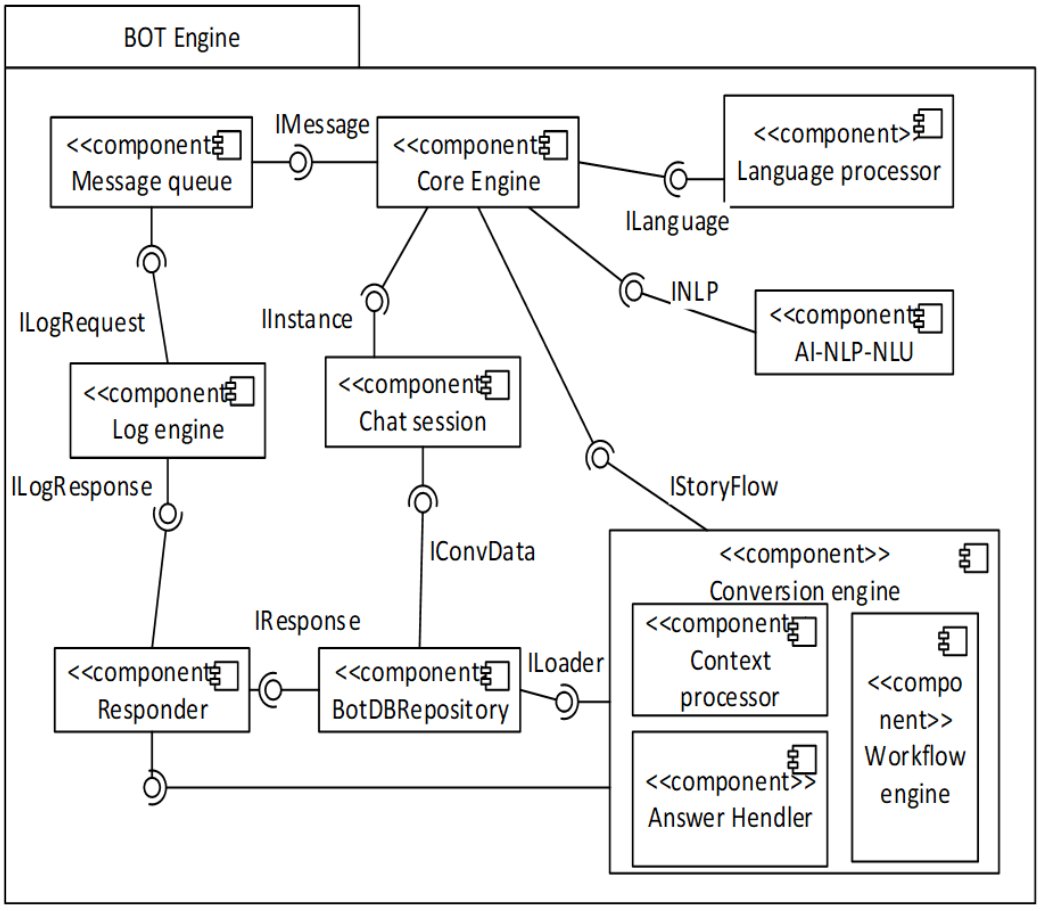

Figure 2. Bot engine component diagram.

In the component Message queue, attention is paid to the sequence of messages arriving at the channel and their release to processing. All messages entering and exiting the Bot engine are monitored in the log system using this component. Each chat session has its life

cycle that follows at least one definition of a scenario created via BotDesigner. When the user initiates a scenario, the Chat Session component creates one instance of the scenario for that user. It has the possibility to offer recognized contexts from the previous conversation as the context for current conversation (context filling) and has the option to recognize whether the user wanted to switch any context value used in the previous conversation (context switching). The user can change the language in which it addresses the chatbot at any moment, and the component Language Processor is responsible for understanding the changing language. If it is necessary, this component must ensure the best possible translation into the adequate language so that the AI-NLP-NLU component can process the text. Based on intent and context, the task of the Conversation engine is to decide which scenario it should process and whether it is a new scenario or a continuation of an existing scenario. The processing of context performed by the Context Processor is also performed here. Workflow mechanisms have been incorporated into the suggested framework. Workflow engine has the task of taking care of every scenario that follows its scenario definition, i.e., the sequence of action of the ongoing scenario instance. Each action has contexts, which are supposed to be met for it to be executed. In addition, each action has the option of built-in or custom validation for validating the user entry and skipping the processing of the AI-NLP-NLU component. An example of this is when an action in some scenarios requires a number, email, and telephone that can be validated skipping the components for translation and AI-NLP-NLU. The Answer Handler component has the task of providing adequate answers for actions within the scenario. Answers can be of different types: text, picture, URL, location, i.e., union of types defined by the communication platform. Multiple responses of the same or different types may be defined for a certain action in the proper sequence in which they will be delivered. The Responder component prepares the message for sending back to the messenger connector to forward it to the user. Since the connector is presented as a microservice, this component addresses the connector through Hypertext Transfer Protocol (HTTP) RESTful service and forwards the prepared message. Many components in a bot framework rely on storage and database access. For example, the conversation engine needs to know about scenario definitions and their instances. The chat session handles all users and their communications. It is also very important to remember all messages, entities, and intents. All metamodels and corresponding mapping rules are also important and need to be stored in the database. The task of the BotDBRepository component is responsible for all necessary interfaces for communication with the database.

NLU have challenging problems that demand significant expertise. This is one of the reasons why companies frequently choose not to build their own solution but use other service platforms. Microsoft LUIS.ai [35], Facebook Wit.ai [36], Google Dialogflow [37], RASA [38] and IBM Watson [39] are some of the many popular NLU service platforms. They accept natural language and return structured data (data divided into a simple and organized format for simple processing, e.g., JSON—JavaScript Object Notation). These companies have turned many years of experience into platforms that provide good solutions to smaller or larger companies for machine learning. Our architecture precisely supports this approach via our AI-NLP-NLU component. An utterance comes to enter this component so that the user intent could be sent to processing and detected with all necessary parameters. Each external NLU service has its own rules and mappings applied when creating requests towards service and response serialization. All these mappings are set up when adding services for NLU. The main task of the component is to forward the detected intents and entities from the external services in the proper form understood by the Bot engine so that it can continue with the process. It has the possibility to have N services that will be in single/multi-mode, i.e., active/passive mode, that can help in better intent detection. If there are two NLU services in the multi-mode, the processing request is forwarded in parallel to both services. Two responses are taken into consideration and the response which best meets the set criteria continues the processing. One of the main roles of this component is intent and entity detection. In addition to this role, this

component has other roles administered in the tool for administration of the external NLU service. Each NLU service has its own metamodel and mapping rules, which are used by the AI-NLP-NLU component to fill the object with all necessary data and help to be independent from just one vendor of NLU services.

5. NLU Metamodels and Corresponding Mapping Rules

NLU is a Natural Language Understanding engine classifying utterance by intents and extracting relevant information from utterances called entities. The main role of NLU is to try to understand the user input and the intent of the conversation and extract entities that are needed to perform a user action or business logic. We can think of NLU services as a set of high-level APIs for building our own language parser using existing NLU and Machine Learning libraries. To be independent from just one NLU external service, we develop our own NLU metamodel, called the DORIUS metamodel, excerpt shown in Figure 3. The main reason for developing our own metamodel version is to allow easier correspondences between concepts of two very popular NLU services (Dialogflow and RASA). We show how our NLU metamodel is mapped to metamodels of Dialogflow and RASA. The defined mappings must follow rules and constraints, which are defined by the mapping metamodel [40]. The excerpt of Dialogflow and RASA metamodels are shown in Figures 4 and 5. These are also original metamodels developed by the authors of this paper. The versions of Dialogflow and RASA we use in our approach to develop metamodels are officially described in [37,38]. All these metamodels and corresponding mapping rules are defined in the database at design time. The designer enters training data into NLU concepts of DORIUS metamodel for each utterance. NLU training data consists of example user utterances categorized by intent. When the clients of the chatbot decide which NLU services they want to use, the corresponding NLU metamodel is automatically loaded from the DORIUS metamodel. Using mapping rules, we automatically create, maintain, and forward objects with all the necessary training data to the corresponding NLU metamodels. After that, training data for the learning chatbot is released. The results of these actions are NLU services which are trained with natural user utterances tagged with entities. During the runtime, NLU services parse the text (utterance) based on machine learning techniques, compares, and checks for a match. NLU services return intent and entities back to the AI-NLP-NLU component with the percentage of accuracy of understanding the intention. As we have already said, the platform ensures that multiple NLU services can be used, but the Core Engine component chooses the one that has a higher probability of accuracy. Using these metamodels proposed in our approach, it is also easy to include new NLU services. All you need is to create a metamodel and the appropriate mapping rules to the DORIUS metamodel, which will automatically extend the choice of NLU services. Furthermore, the DORIUS metamodel can help the old chatbot to switch to another NLU service. Assuming that there is a chatbot that only uses the Dialogflow service, using the DORIUS metamodel it is possible to easily switch the chatbot from Dialogflow to RASA service and vice versa. In this usage, the DORIUS concepts are loaded from Dialogflow metamodel using mapping rules, and then its concepts are mapped to the RASA metamodel.

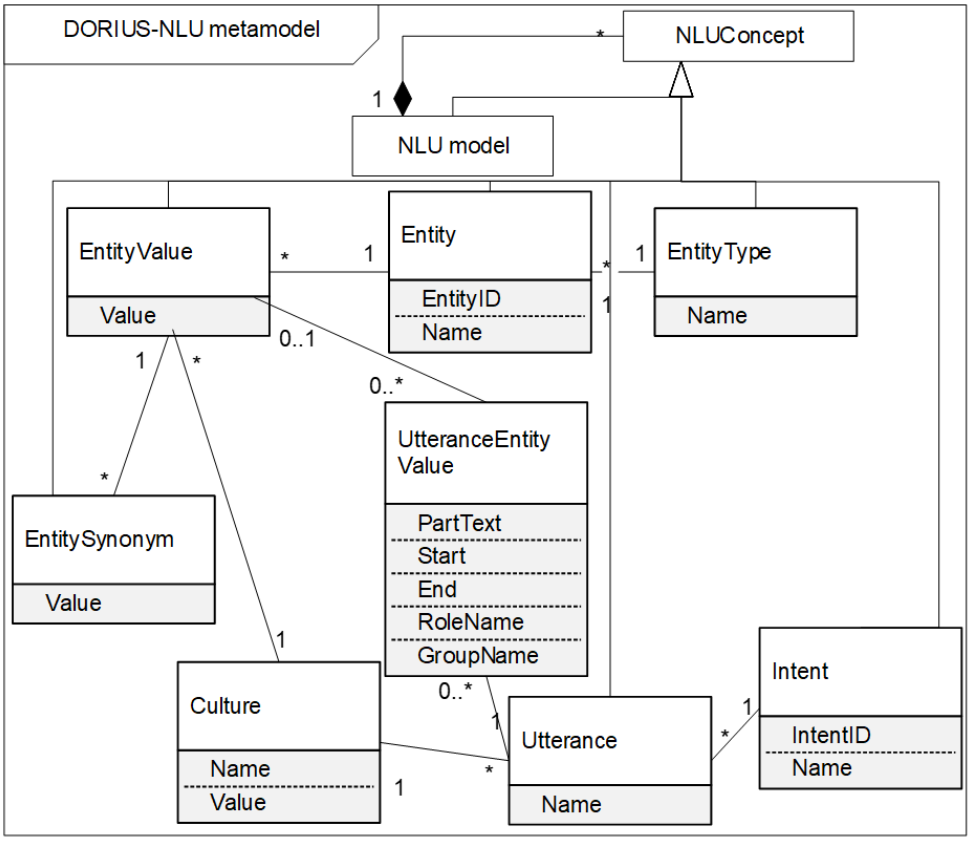

Figure 3. DORIUS metamodel.

The DORIUS NLU Concept represents the most abstract concept in the NLU data model. It is specialized using NLU concepts that are more concrete:

- Intent categorizes a user intention in one conversation. An intent is a group of utterances with similar meaning. Intent refers to the goal the customer has in mind when typing in a question or comment. Attribute IntentID represents a unique identifier of intent. Attribute Name is the name of intent. For example, if the Intent Name is ProfessorConsultation, the IntentID could be Guid123.
- EntityType represents a type of entity. Attribute Name can be systemic or custom. For example, EntityType can be custom.
- Entity represents the entity entry for an associated EntityType. The Entity can be a systemic or custom EntityType. Entities are annotated in training examples with the name of the entity. Entities are a mechanism for identifying and extracting useful data from natural-language inputs. While intents allow understanding the motivation behind a particular user input, entities are structured pieces of information that can be extracted from an utterance. In addition to the entity name, we can annotate an entity with values and synonyms. TeacherName can be an example of the Entity that is a custom EntityType. EntityID could be 3en-22t.
- EntityValue represents the primary value associated with entity entry. The Entity can have one or more Value. For example, if the entity is TeacherName, the EntityValue could be: Rade Matić, Miloš Kabiljo, etc.
- EntitySynonym represents a synonym for EntityValue. You can use synonyms when there are multiple ways users are referring to the same thing. EntityValue can have one or more EntitySynonym. For example, if the EntityValue is Rade Matić, the EntitySynonym could be: R.M., Radetom Matićem, Radu Matiću.
- Utterance is an example (training) phrase for what users might type or say. For each intent, we can have many utterances. When a user message looks like (corresponds to) one of these utterances, NLU matches the intent. Examples of utterances could be:

  - When does Professor Rade Matić have a consultation? (serb. Kada profesor Rade Matić ima konsultacije)?
  - When can I consult with Professor Rade Matić (serb. Kad mogu da se konsultujem sa profesorom Radetom Matićem)?
  - When can I visit professor R.M. (serb. Kad mogu da dođem kod profesora R.M.)?
- UtteranceEntityValue represents entity value in one or more utterances. Utterance can have zero or more values of entity. Attribute start and end positions are important because this is how the model knows which characters to extract and use to train the model. PartText represents the ordered list of utterance parts. The PartText is concatenated to form the utterance. With RoleName we can define entities with specific roles in an utterance. GroupName allows entities to be grouped together with a specific group label. The group label can be used to define different orders. For example, in the utterance: "When does Professor Rade Matić have a consultation", the entity value Rade Matić is PartText that starts at position 20 and ends at position 30.
- Culture provides the possibility of multi-language support. Utterances related to an intent can be defined for each language individually as well as the entities used within them. Each culture can have its own set of utterances and entities.

The Dialogflow metamodel is shown in Figure 4. NLU Concept is further specialized into concepts that are more concrete:

- Intent.
- Kind.
- EntityType.
- Entity.
- Synonym.
- TrainingPhrase.
- Part.

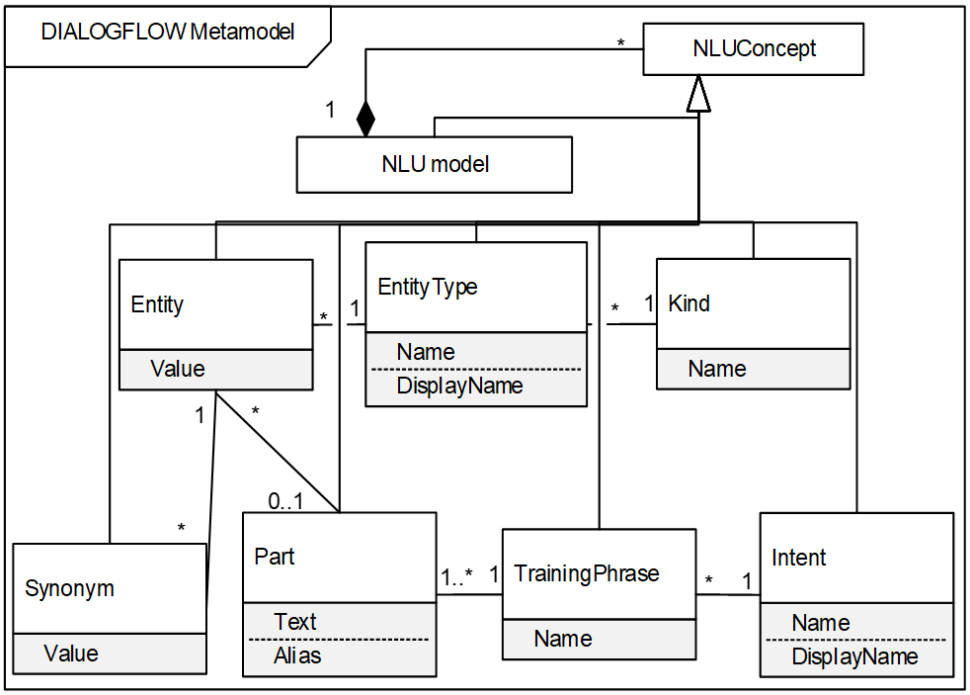

Figure 4. Dialogflow metamodel.

The mapping between DORIUS NLU and Dialogflow NLU concepts is determined by the following rules:

- Dialogflow2DORIUS rule: Each Dialogflow model maps to a DORIUS model.
- I.N-I.ID rule: Each Intent.Name maps to Intent.IntentID.

- I.DN-I.N rule: Each Intent.DisplayName maps to Intent.Name.
- K.N-ET.N rule: Each Kind.Name maps to EntityType.Name.
- ET.N-E.ID rule: Each EntityType.Name maps to Entity.EntityID.
- ET.DN-E.N: Each EntityType.DisplayName maps to Entity.Name.
- E.V-EV.V rule: Each Entity.Value maps to EntityValue.Value.
- S.V-ES.V rule: Each Synonym.Value maps to EntitySynonym.Value.
- TP.N-U.N rule: Each TrainingPhrase.Name maps to Utterance.Name.
- P.T-UEV.PT rule: Each Part.Text maps to UtteranceEntityValue.PartText.
- P.A-UEV.RN rule: Each Part.Alias maps to UtteranceEntityValue.RoleName.

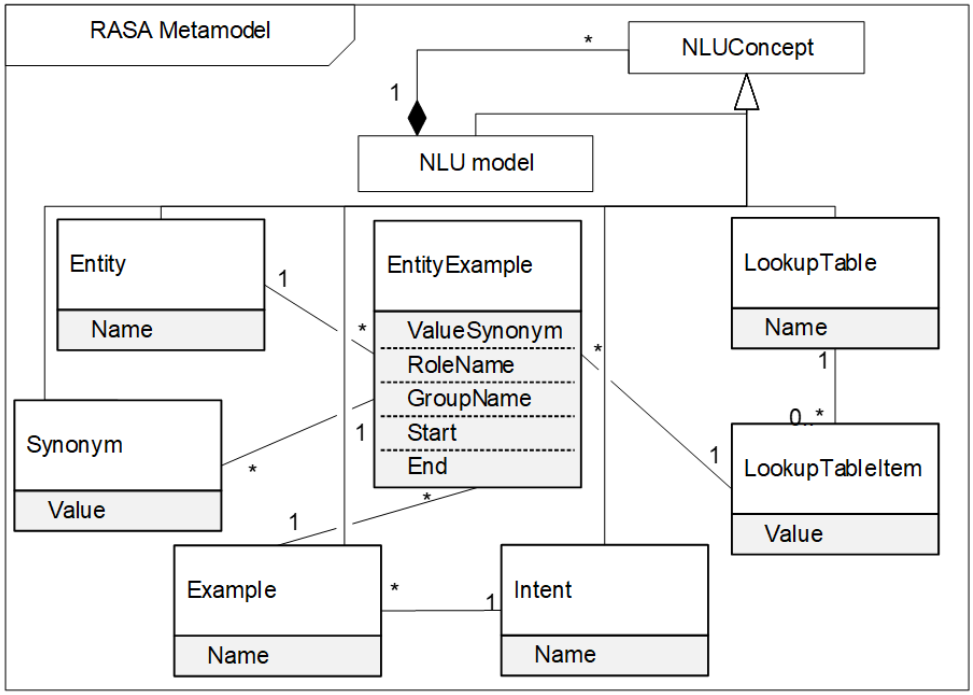

Figure 5. RASA metamodel.

The Mapping metamodel, shown in Figure 6, defines allowed correspondences between Dialogflow NLU concepts and DORIUS NLU concepts based on the rules. A mapping between two concrete Dialogflow and DORIUS models is represented by Dialogflow2DORIUS class, which encompasses all correspondences between concrete elements of the Dialogflow and DORIUS models made using the mapping rules. The class NLUElement represents an instance of such correspondences. For each mapping rule, there are appropriate subclasses of NLUElement, which are named after the rule. As is obvious from the metamodel, mappings between concepts of Dialogflow model and DORIUS model are unique and non-ambiguous.

The excerpt of the RASA metamodel with the most abstract RASA NLU concept is shown in Figure 5. It is specialized using more concrete NLU concepts:

- Intent.
- Entity.
- EntitySynonym.
- Example.
- EntityExample.
- LookupTable.
- LookupTableItem.

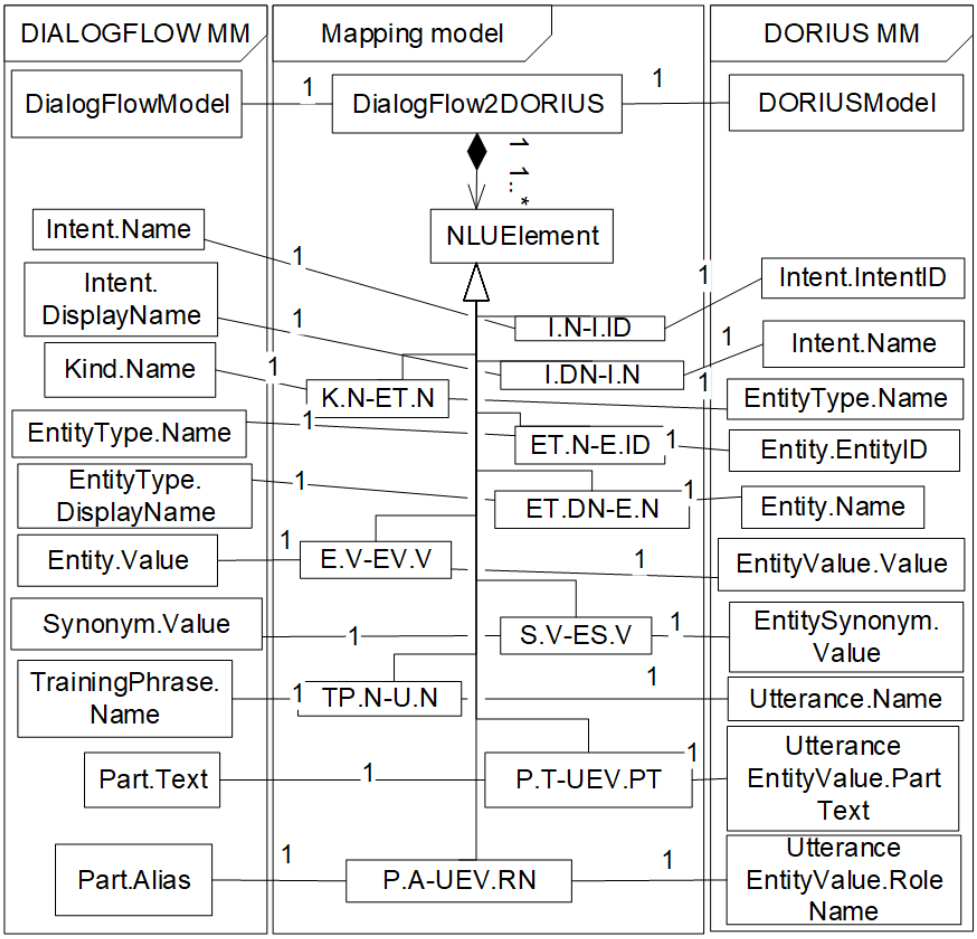

Figure 6. Dialogflow—DORIUS Mapping metamodel.

The mapping between RASA NLU and DORIUS NLU concepts is determined by the following rules:

- RASA2DORIUS rule: Each RASA model maps to a DORIUS model.
- I.N-I.ID rule: Each Intent.Name maps to Intent.IntentID.
- I.N-I.N rule: Each Intent.Name maps to Intent.Name.
- E.N-E.N rule: Each Entity.Name maps to Entity.EntityID.
- LT.N-E.N rule: Each LookupTable.Name maps to Entity.EntityID.
- LTI.V-EV.V rule: Each LookupTableItem.Value maps to EntityValue.Value.
- S.V-ES.V rule: Each Synonym.Value maps to EntitySynonym.Value.
- E.N-U.N rule: Each Example.Name maps to Utterance.Name.
- EE.VS-EV.V rule: Each EntityExample.ValueSynonym maps to EntityValue.Value.
- EE.Start-UEV.Start rule: Each EntityExample.Start maps to UtteranceEntityValue.Start.
- EE.End-UEV.End rule: Each EntityExample.End maps to UtteranceEntityValue.End.
- EE.RN-UEV.RN rule: Each EntityExample.RoleName maps to UtteranceEntityValue. RoleName.
- EE.GN-UEV.GN rule: Each EntityExample.GroupName maps to UtteranceEntity-Value.GroupName.

The Mapping metamodel, shown in Figure 7, defines allowed correspondences between RASA NLU concepts and DORIUS NLU concepts based on the rules. A mapping between two concrete RASA and DORIUS models is represented by RASA2DORIUS class, which encompasses all correspondences between concrete elements of the RASA and DORIUS models made using the mapping rules. The class NLUElement represents an instance of such correspondences. For each mapping rule, there are appropriate subclasses of NLUElement, which are named after the rule. The same as in the previous mapping

metamodel, mappings between concepts of the RASA model and DORIUS model are unique and non-ambiguous.

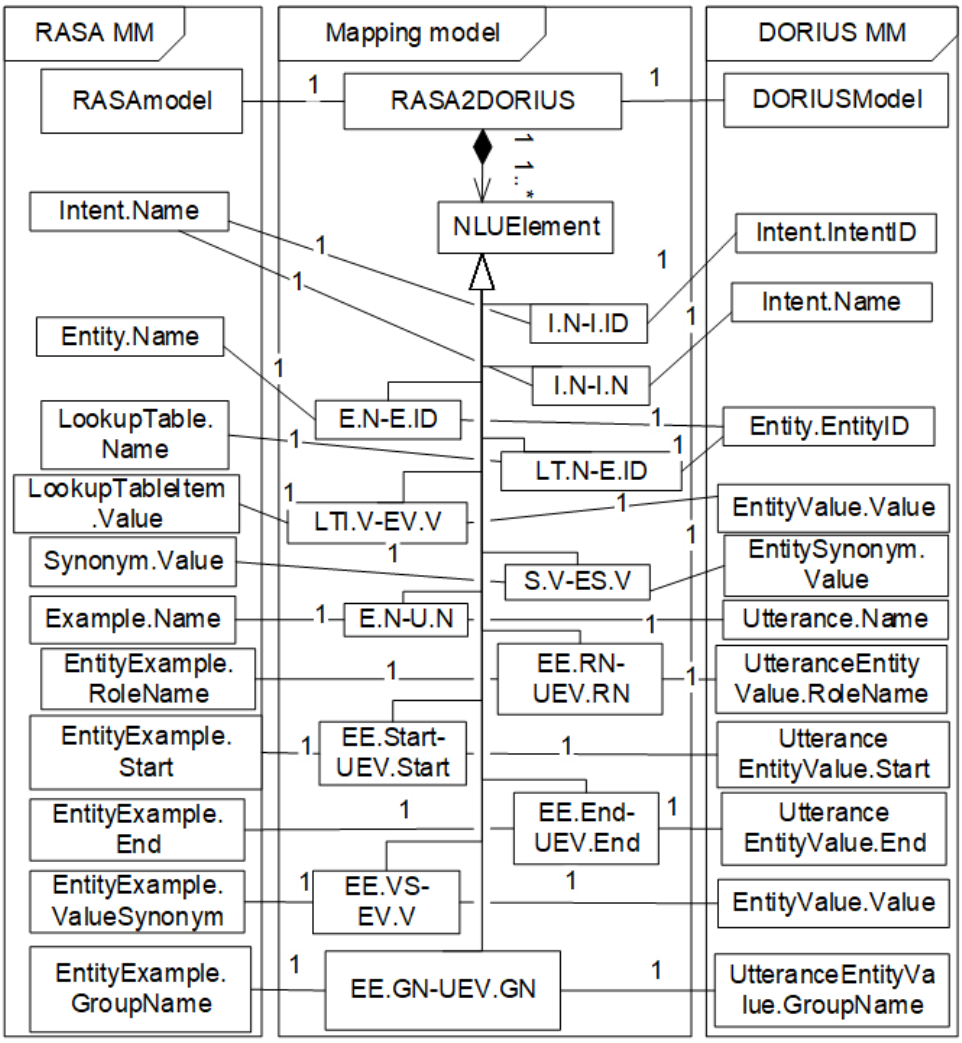

Figure 7. RASA—DORIUS Mapping metamodel.

6. Example of Mapping Model

A part of the corresponding mapping model between the Dialogflow model and the DORIUS model is given in the object diagram in Figure 8. The Dialogflow model package contains instances of the Dialogflow metamodel from Figure 4, while the package DORIUS model contains instances of the DORIUS metamodel from Figure 3. The Package Mapping model contains instances of the corresponding mapping rules by which concepts of the Dialogflow are mapped from the concept of the DORIUS. For example, intent i1 (ProfessorConsultation) is mapped from corresponding intent of the same name using the I.DN-I.N rule. Kind k1 is mapped from et1:EntityType of the same name using the K.N-ET.N rule. On the other hand, attribute Name (3en-22t) of et1:EntityType, is mapped from attribute EntityID (3en-22t) of e1:Entity by rule ET.N-E.ID. Furthermore, attribute DisplayName (TeacherName) of et1:EntityType, is mapped from attribute Name (TeacherName) of e1:Entity by rule ET.DN-E.N. Thus, entities e1 (Rade Matić) and e2 (Miloš Kabiljo) are mapped from corresponding entity values ev1 and ev2 of the same value using the E.V-E.VV rule. Furthermore, synonyms s1 (R.M.) and s2 (Radu Matiću) are mapped to corresponding entity synonyms es1 and es2 of the same value using the S.V-E.SV rule. Training phrases tp1 and tp2 are mapped from corresponding utterances u1 and u2 of the same name using the TP.N-U.N rule. Attribute Text of p1:Part is mapped from at-

tribute PartText of uev1:UtteranceEntityValue by rule P.T-UEV.RN. UtteranceEntityValue represents entity value in one or more utterances. For example, in the utterance: "When does Professor Rade Matić have a consultation?", entity value Rade Matić is PartText. An example of a mapping model with only one NLU metamodel is given here. A similar example of mapping can be shown for the RASA metamodel, but due to work limitations, the details of mapping are not part of this work.

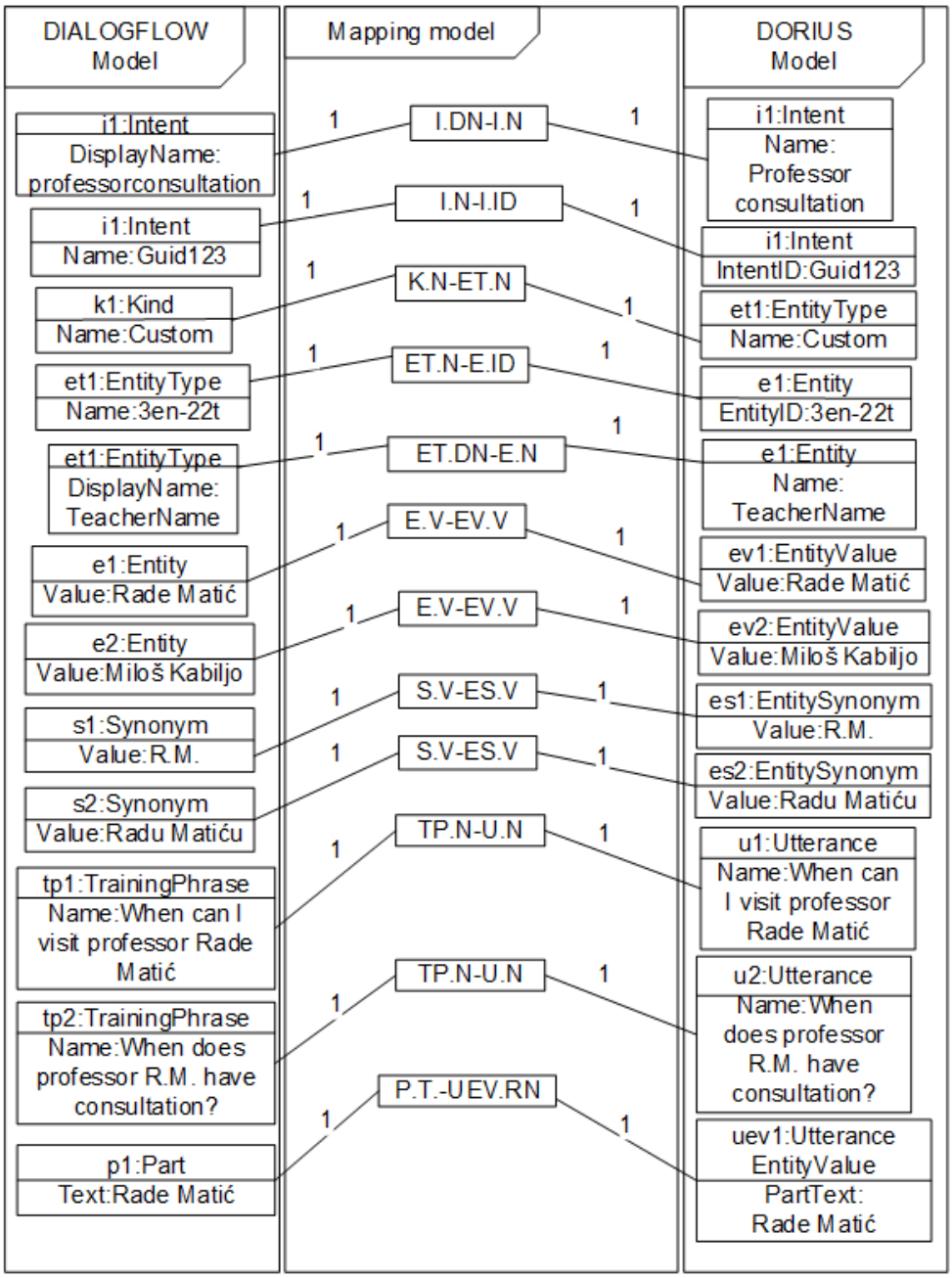

Figure 8. A part of mapping model between Dialogflow and DORIUS model.

7. Two Case Studies and Their Implementations

Some experiences in the implementation of the described chatbot architecture exist in two case studies. The first one is an ADA chatbot of Belgrade Business and Art Academy of Applied Studies and the second is "COVID-19 Info Serbia". In the conditions of the COVID-19 pandemic and due to the closure of the education system, Belgrade Business and Arts Academy of Applied Studies [41] is required to switch to the online delivery of interaction with their students and educational content [42]. To improve, modernize,

and digitalize education services, the Academy developed an ADA (Academic Digital Assistant) [43] chatbot using the Weaver platform to provide its students with improved service and necessary information during their studies, as shown in Figure 9.

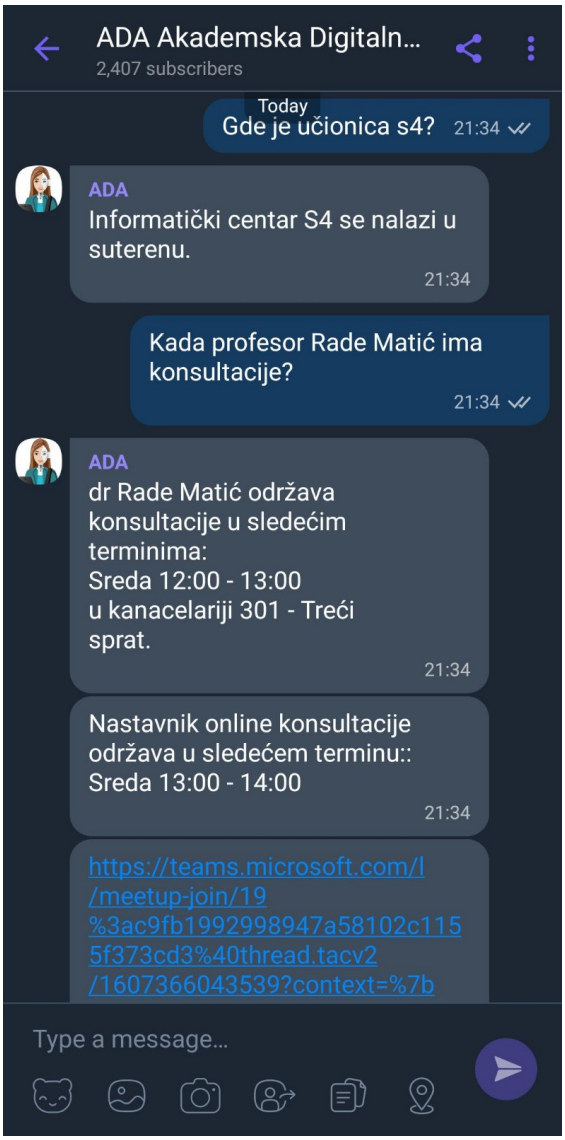

Figure 9. ADA (Academic Digital assistant) chatbot, implemented in the Belgrade Business and Arts Academy of Applied Studies.

ADA is currently using two NLU external services: RASA and Dialogflow. Thanks to the solution proposed here, it was easy to switch from one NLU service to another. Testing was performed in three phases. The first phase involved internal testing of the intents due to the specifics of the Serbian language. We confirmed the shortcomings of Dialogflow from the paper [32] and switched to RASA. The second phase of testing was carried out in a controlled environment with a group of 50 students using RASA. We told these students what the ADA knows, but we did not tell them how to ask. The goal was to achieve 75% accuracy. Students entered the appropriate utterances, but we obtained an average accuracy of recognizing intents around 55%. Our learning model was not as good as we expected, but utterances that the ADA did not recognize helped us to boost training. The model became much richer with new utterances. The third and final phase of testing involved all students of an IT major at the Academy. They provided us with many utterances that we constantly added to the training until we managed to get the ADA to answer 85% of the questions. Issues that were outside of the learned model were excluded

from training and confidence score because some utterance was out of the scope for this part of the test. To get the most out of our training data, we trained and evaluated our model on different pipelines and different amounts of training data. To support Serbian, we did not use the spaCy library. The same recommendation is made for other languages that do not have support in spaCy. For English and other more popular languages, spaCy is very useful to include. Regardless of which language we use, it is very important to get the most out of training data. We had to train and evaluate our model on different pipelines and different amounts of training data.

A snapshot of one scenario definition and NLU validation is shown in Figures 10 and 11, respectively. Figure 10 shows how the administrator can choose which NLU service to use. Figure 11 shows the NLU validation of utterances for the two intents asked by the students, as well as their confidence core on the RASA NLU service. Using the NLU validation it is possible to directly teach the NLU service to avoid dependency on the NLU interface of the vendor.

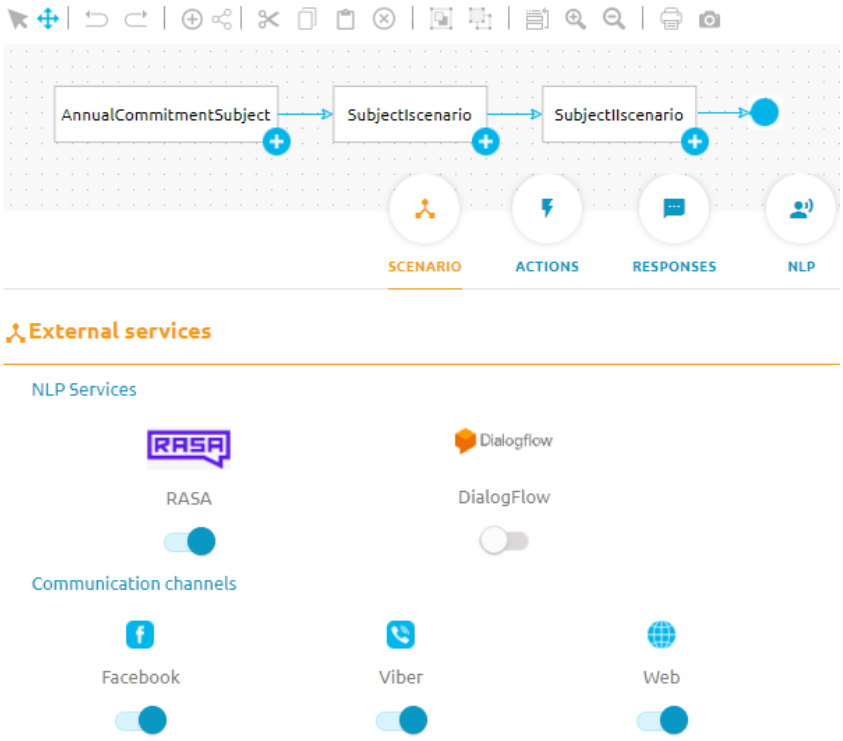

Figure 10. Bot Designer.

Figure 11. NLU Validation.

ADA helps prospective students learn more about the Academy. ADA is trained to provide students with the highest quality educational service with a wide range of information about the teachers, classrooms, curriculum, literature, working hours, library, subjects, notifications, education content, lecture schedule, and price list in an easy-to-use, conversational manner. Students ask ADA for various information such as: the total cost of tuition, which documents are required for enrolment in the new school year, the schedule of lectures, or the date of admission of a particular professor, as shown in Figure 9. ADA also can perform some processes such as exam registration or cancellation, change of teacher on the exam, reset password, or buy a book. In the analyzed months, the interest of students was in accordance with the school calendar, that is, in January, the month of taking the exams, students were most interested in when the teacher teaches, examines, or holds exercises in a certain subject, as shown in Table 1. In February, the month of enrolment in the new school year, at the top of the list of inquiries was information related to the cost of tuition and the type of documents required for enrolment, as shown in Table 2. When both Tables 1 and 2 are considered, it is also possible to observe the significant increase in the total amount of inquiries.

Table 1. Statistics of student inquiries in January 2021.

The Most Frequently Asked Questions with no. of Conversations	
Which professor teaches and examines a particular subject	178
The amount of any tuition fee	176
What is the schedule of lectures	140
When lectures start in a given semester	137
When does the student enrolment begin	134

Table 2. Statistics of student inquiries in February 2021.

The Most Frequently Asked Questions with no. of Conversations	
Required documentation for enrolment for a specific school year	513
The amount of any tuition fee	472
In which cabinet and at what time professor holds consultations	470
Location, working hours, books offered in the Academy's script shop	459
Budget ranking information	298

Thanks to this, we managed to get almost 60% of students to use ADA as their primary communication channel with the Academy. It also provides links to additional resources for those who wish to learn more about the Academy. The ADA chatbot project is the first such project in the Balkans in the field of education that provides time saving, easier and better communication, and faster and more efficient implementation of educational services. Chatbot technology has the potential to significantly influence the way students experience educational institutions and the way they interact with them. The aim of the platform, in addition to being available to many students for all necessary information related to the Academy, is to obtain official information without waiting, as well as to relieve contact center personnel and teachers, allowing them to pay attention to more creative tasks. The Chatbot is freely available and launches in Serbian via Viber and FB messenger.

Because the Serbian language is very difficult, a very large training set with various utterances was needed for the model to be as efficient as possible. In addition to utterances, we take care of the entities, their values, as well as their synonyms due to the complexity of the Serbian language, which can be a problem during the recognition of intent. ADA, unlike COVID-19, has a lot of entities that have the same value but different meanings. In that case, we use the concept of entity role so that the same entity in a sentence can have two meanings. An example of this entity is "management", which can be the name of subject in one study program and the name of another study program. Table 3 presents the

number of intents, utterances, entities to make ADA and COVID-19 functional in Serbian. These numbers are growing because we are constantly working on learning and improving the model.

Table 3. The number of intents, utterances, entities to make ADA and COVID-19 functional in Serbian.

	ADA	COVID-19
Intent	202	107
Utterance	7549	3770
Entity	20	10
Entity values	525	120
Entity synonyms	3524	150

To facilitate the process of efficient and timely information dissemination, a proposed solution is participating in developing "COVID-19 Info Serbia" [44]. This includes a wide range information about coronavirus and its symptoms. It also provides links to additional resources for those who wish to learn more about the disease. The chatbot is freely available and launches in Serbian and English via Viber [45]. "COVID-19 Info Serbia" is a beta version chatbot developed on the Weaver platform. This is a retrieval-based model connected with NLU, which is classified into closed domains and it provides easier and quicker information regarding the COVID-19 virus epidemic in Serbia [46]. The Virtual Assistant answers all potential citizen questions regarding COVID-19 without waiting, and it is available 24/7 at the official website of the government related to the COVID-19 virus situation. The "COVID-19 Info Serbia" chatbot has the potential to create individual learning experiences about COVID-19. The chatbot responds with official answers to most common questions of citizens regarding COVID-19, including details on the state of emergency, local emergency numbers, symptoms, the latest Ministry of Health updates, and other useful information. It also helps reduce the pressure on healthcare hotlines and keeping the phone lines open for people who really need to speak to a doctor. "COVID-19 Info Serbia" is obviously a simplification of what a chatbot could do. It can be improved and updated easily by teaching the chatbot to provide rich experiences, but this mostly depends on the trainers of the chatbot.

COVID-19 had accelerated testing due to short deadlines and a pandemic situation. This chatbot is therefore different because it has simple scenarios and there are also button-based scenarios. The pilot production started very quickly, where a lot of citizens participated, who strengthened the training with various issues and expanded the domain of model knowledge. We also implemented a scenario that provides an opportunity for citizens to register for vaccination. About 30,000 citizens applied for vaccination in the first round via the COVID-19 chatbot. This chatbot currently has about 300,000 subscribers on the Viber channel, and the success rate for correct answers was 80%.

The Weaver platform is built on trained NLU engines, but each chatbot that has been built has a different set of conversation designs and solves different industry problems. To achieve the best outcome, we need a decent amount of good quality data. Under the condition to provide the best conversation flow, the platform will be able to work at its best.

8. Conclusions

Driven by the promise of intelligent digital assistants that will always be at disposal for fast and consistent resolution of client requests, chatbots are becoming increasingly useful. One of the main goals was to make an agile and extensible chatbot architecture to handle a highly variable environment of modern platforms and emerging technologies. The advantages offered by proposed chatbot architecture are the following:

- It is extensible, supporting new natural language understanding and communication channels for user interactions.
- It relies on metamodels as the main extensible mechanism. For this we provide:

  - General NLU metamodel.
  - Metamodels for the two specific NLU services (Dialogflow and RASA).
  - Corresponding metamodels as well as rules for a mapping between generic and specific NLU metamodels.
- It is tailorable and customizable, referring to the ability of the architecture to be managed and customized by a modeler.
- It is inherently scalable by means of microservices.
- The solution was implemented in two case studies on the Serbian language.
- It provides a mature runtime functionality within the domain of chatbots.
- It provides a workflow representing detailed and dynamic business processes (scenarios) that are concerned with the step-by-step sequence of activities used to complete the intents.

In this paper, we show software architectures that can be expanded, adapted, managed, and customized to meet future changes and chatbot technologies. Platform independence is closely related to the concern for open APIs and support for distributed objects. We have presented the advanced architecture of the chatbot framework built on microservices, i.e., API service architecture. We also presented a solution for creating a chatbot that is independent of one external NLU service. Lessons learned from both cases proved that Web API was a good choice among other things because:

- The service is based on the HTTP protocol-enabling RESTful service and JSON objects.
- It can be hosted within the application or IIS (Internet Information Services).
- It can be used by any client who understands JSON or extensible markup language (XML).
- It has a simple architecture.

When developing and designing chatbot architecture, we use several techniques for scalability. Thanks to microservices, our architecture can replace a component with more powerful and faster ones as requirements grow and technology develops. The chatbot platform does not care about the front end of communication platforms. Maintenance is easier because the front end is maintained by the communication service. For the first phase, we only use the local cache to accelerate request retrieval. We use index tables to solve a quick search. We regulate concurrent requests and handle many of them. We perform queries asynchronously and queue them. Asynchronous processing removes some of the bottlenecks that affect performance. We increase the scalability choosing Angular as one of the modern frameworks for single page application enhancing the overall performance. The user interface for NLU learning is defined in one place, regardless of different NLU external services. Scenario definitions are also defined in one place regardless of different communication platforms. Switching from one to more NLU or introducing new NLU is very easy to accomplish in a couple of clicks due to the metamodels and low coupling of the basic parts of the architecture. This reduces the learning curve for the user and helps in better understanding of user message. The solution was implemented in two case studies: "ADA" and "COVID-19 Info Serbia". ADA Chatbot is a platform that uses NLU to find the right answer to all student questions and solve their problems. In that way, the satisfaction of students increases, and they achieve a better interaction with the Academy using more than one communication platform and NLU services. COVID-19 has accelerated the need for chatbot solutions. Chatbots may not be the solution to all student problems, but it is a powerful tool that can increase the efficiency of the Academy and support the realization of the educational process. After the pandemic, the use of chatbots for information and education applications will continue to grow. What says the most about the success of this implementation is that the Academy dedicated the whole brand image campaign to ADA. To better inform the citizens of Serbia about COVID-19, the Government of Serbia has launched a chatbot. The goal of "COVID-19 Info Serbia" is to reach as many people as possible with reliable health information using AI-powered contact center messaging [47]. The proposed platform covers general scenarios and topics related to the virus, and focuses on information on symptoms, prevention measures, important phones, current data, and

decisions of the Government of the Republic of Serbia. However, this training procedure requires more patience, time, and domain knowledge. There is always a need for human intervention, even where chatbots are deployed. This research work can be continued in the direction of full automation and extension of our prototype with new possibilities. For instance, an automatic extraction of user sentiment from communication could be added. We need to create a distributed cache to provide the data piece distribution throughout all the nodes. The support service should then be enabled for voice messages and additional performance measurements to identify bottlenecks, using evaluation techniques such as [17,18]. The authors claim that the proposed chatbot architecture presented in this paper is an excellent starting point for these research directions.

Author Contributions: Conceptualization, R.M. and M.K.; methodology, M.Z. and M.C.; software, R.M. and M.K.; validation, R.M., M.Z. and M.C.; formal analysis, M.K. and R.M.; investigation, R.M. and M.K.; resources, R.M., M.K., M.Z. and M.C.; writing—original draft preparation, R.M. and M.K.; writing—review and editing, M.Z. and M.C.; visualization, R.M.; supervision, R.M. and M.K.; project administration, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bansal, H.; Khan, R. A review paper on human computer interaction. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2018**, *8*, 53. [CrossRef]
2. Dale, R. The return of the chatbots. *Nat. Lang. Eng.* **2016**, *22*, 811–817. [CrossRef]
3. Smutny, P.; Schreiberova, P. Chatbots for learning: A review of educational chatbots for the Facebook Messenger. *Comput. Educ.* **2020**, *151*, 103862. [CrossRef]
4. Chatbot Report 2019: Global Trends and Analysis | by BRAIN [BRN.AI] CODE FOR EQUITY | Chatbots Magazine. Available online: https://chatbotsmagazine.com/chatbot-report-2019-global-trends-and-analysis-a487afec05b (accessed on 5 June 2021).
5. Chatbot Market Size, Share | Industry Trends and Analysis by 2027. Available online: https://www.alliedmarketresearch.com/chatbot-market (accessed on 5 June 2021).
6. Fayad, M.E.; Hamza, H.S.; Sanchez, H.A. Towards scalable and adaptable software architectures. In Proceedings of the IRI-2005 IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, USA, 15–17 August 2005; pp. 102–107.
7. Weaver. Available online: https://weaverbot.ai/ (accessed on 5 June 2021).
8. Weizenbaum, J. ELIZA—A computer program for the study of natural language communication between man and machine. *Commun. ACM* **1966**, *9*, 36–45. [CrossRef]
9. Weizenbaum, J. Eliza—A computer program for the study of natural language communication between man and machine. *Commun. ACM* **1983**, *26*, 23–28. [CrossRef]
10. Marietto, M.D.G.B.; de Aguiar, R.V.; Barbosa, G.D.O.; Botelho, W.T.; Pimentel, E.; França, R.D.S.; da Silva, V.L. Artificial intelligence markup language: A brief tutorial. *arXiv* **2013**, arXiv:1307.3091.
11. Wallace, R.S. The anatomy of ALICE. In *Parsing the Turing Test*; Springer: Dordrecht, The Netherlands, 2009; pp. 181–210.
12. Rodríguez Cardona, D.; Werth, O.; Schönborn, S.; Breitner, M.H. A mixed methods analysis of the adoption and diffusion of Chatbot Technology in the German insurance sector. In Proceedings of the AMCIS, Cancun, Mexico, 15–17 August 2019.
13. Adamopoulou, E.; Moussiades, L. Chatbots: History, technology, and applications. *Mach. Learn. Appl.* **2020**, *2*, 100006.
14. Mu, J.; Sarkar, A. Do we need natural language? Exploring restricted language interfaces for complex domains. In Proceedings of the Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; pp. 1–6.
15. Daniel, G.; Cabot, J.; Deruelle, L.; Derras, M. Multi-platform chatbot modeling and deployment with the Jarvis framework. In Proceedings of the International Conference on Advanced Information Systems Engineering, Rome, Italy, 3–7 June 2019; Springer: Basel, Switzerland, 2019; pp. 177–193.
16. Abdul-Kader, S.A.; Woods, J. Survey on chatbot design techniques in speech conversation systems. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*. [CrossRef]
17. Radziwill, N.M.; Benton, M.C. Evaluating quality of chatbots and intelligent conversational agents. *arXiv* **2017**, arXiv:1704.04579.
18. Pereira, J.; Díaz, O. A quality analysis of facebook messenger's most popular chatbots. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, Pau, France, 9–13 April 2018; pp. 2144–2150.
19. Nimavat, K.; Champaneria, T. Chatbots: An overview. Types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev.* **2017**, *5*, 1019–1024.
20. Jwala, K.; Sirisha, G.; Raju, G.P. Developing a Chatbot using Machine Learning. *Int. J. Recent Technol. Eng. (IJRTE)* **2019**, *8*, 89–92.
21. Swanson, K.; Yu, L.; Fox, C.; Wohlwend, J.; Lei, T. Building a production model for retrieval-based Chatbots. *arXiv* **2019**, arXiv:1906.03209.

22. Peng, Z.; Ma, X. A survey on construction and enhancement methods in service chatbots design. *CCF Trans. Pervasive Comput. Interact.* **2019**, *1*, 204–223. [CrossRef]
23. Hussain, S.; Sianaki, O.A.; Ababneh, N. A survey on conversational agents/chatbots classification and design techniques. In Proceedings of the Workshops of the International Conference on Advanced Information Networking and Applications, Matsue, Japan, 27–29 March 2019; Springer: Basel, Switzerland, 2019; pp. 946–956.
24. Khan, R. Standardized architecture for conversational agents aka chatbots. *Int. J. Comput. Trends Technol.* **2017**, *50*, 114–121. [CrossRef]
25. Motger, Q.; Franch, X.; Marco, J. Conversational Agents in Software Engineering: Survey, Taxonomy and Challenges. *arXiv* **2021**, arXiv:2106.10901.
26. Daniel, G.; Cabot, J.; Deruelle, L.; Derras, M. Xatkit: A multimodal low-code chatbot development framework. *IEEE Access* **2020**, *8*, 15332–15346. [CrossRef]
27. Pérez-Soler, S.; Guerra, E.; de Lara, J. Model-driven chatbot development. In Proceedings of the International Conference on Conceptual Modeling, Vienna, Austria, 3–6 November 2020; Springer: Basel, Switzerland, 2020; pp. 207–222.
28. Hill, J.; Ford, W.R.; Farreras, I.G. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Comput. Hum. Behav.* **2015**, *49*, 245–250. [CrossRef]
29. Divya, S.; Indumathi, V.; Ishwarya, S.; Priyasankari, M.; Devi, S.K. A self-diagnosis medical chatbot using artificial intelligence. *J. Web Dev. Web Des.* **2018**, *3*, 1–7.
30. Petrovic, A.; Zivkovic, M.; Bacanin, N. Singibot-A Student Services Chatbot. In Proceedings of the Sinteza 2020-International Scientific Conference on Information Technology and Data Related Research, Belgrade, Serbia, 17 October 2020; Singidunum University: Belgrade, Serbia, 2020; pp. 318–323.
31. Ranoliya, B.R.; Raghuwanshi, N.; Singh, S. Chatbot for university related FAQs. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1525–1530.
32. Abdellatif, A.; Badran, K.; Costa, D.; Shihab, E. A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering. *IEEE Trans. Softw. Eng.* **2021**. [CrossRef]
33. Adamopoulou, E.; Moussiades, L. An overview of chatbot technology. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Halkidiki, Greece, 5–7 June 2020; Springer: Basel, Switzerland, 2020; pp. 373–383.
34. Fayad, M.E.; Hamu, D.S.; Brugali, D. Enterprise frameworks characteristics, criteria, and challenges. *Commun. ACM* **2000**, *43*, 39–46. [CrossRef]
35. LUIS (Language Understanding)—Cognitive Services—Microsoft. Available online: https://www.luis.ai/ (accessed on 5 July 2021).
36. Wit.ai. Available online: https://wit.ai/ (accessed on 5 July 2021).
37. Dialogflow, Natural Language Understanding Platform. Available online: https://cloud.google.com/dialogflow/docs/ (accessed on 5 June 2021).
38. Open Source Conversational AI | Rasa. Available online: https://rasa.com/ (accessed on 5 June 2021).
39. IBM Watson | IBM. Available online: https://www.ibm.com/watson (accessed on 5 July 2021).
40. Nešković, S.; Matić, R. Context modeling based on feature models expressed as views on ontologies via mappings. *Comput. Sci. Inf. Syst.* **2015**, *12*, 961–977. [CrossRef]
41. BAPUSS—Beogradska Akademija Poslovnih i Umetničkih Strukovnih Studija. Available online: https://www.bpa.edu.rs/ (accessed on 5 July 2021).
42. Kabiljo, M.; Vidas-Bubanja, M.; Matić, R.; Zivković, M. Education system in the republic of serbia under COVID-19 conditions: Chatbot-acadimic digital assistant of the belgrade business and arts academy of applied studies. *Knowl. Int. J.* **2020**, *43*, 25–30.
43. ADA Chatbot. Available online: https://chatbot.bpa.edu.rs/en/index.html (accessed on 5 July 2021).
44. Ministarstvo Zdravlja Republike Srbije—COVID-19. Available online: https://covid19.rs/homepage-english/ (accessed on 5 July 2021).
45. COVID-19 Info Srbija on Viber. Available online: https://chats.viber.com/covid19info (accessed on 5 July 2021).
46. COVID-19 Info Serbia | Saga—New Frontier Group. Available online: https://saga.rs/news/COVID-19-info-serbia/?lang=en (accessed on 5 July 2021).
47. Nguyen, T.T.; Nguyen, Q.V.H.; Nguyen, D.T.; Hsu, E.B.; Yang, S.; Eklund, P. Artificial intelligence in the battle against coronavirus (COVID-19): A survey and future research directions. *arXiv* **2020**, arXiv:2008.07343.