electrónica

MDPI

Artículo

Arquitectura de chatbot extensible usando metamodelos de comprensión del lenguaje natural

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

1 Departamento de Sistemas y Tecnologías de la Información, Academia de Belgrado de Estudios Aplicados a las Artes y los Negocios, Kraljice Marije 73, 11000 Belgrado, Serbia; rade.matic@bpa.edu.rs (RM); milos.kabiljo@bpa.edu.rs (MK)

2 Facultad de Informática y Computación, Universidad Singidunum, Danijelova 32, 11000 Belgrado, Escuela de

3 Ingeniería Eléctrica de Serbia, Universidad de Belgrado, Bulevar Kralja Aleksandra 73, 11000 Belgrado, Serbia

* Correspondencia: mzivkovic@singidunum.ac.rs (MZ); cabmilan@etf.bg.ac.rs (MC) † Estos autores contribuyeron igualmente a este trabajo.

Abstracto: En los últimos años, las mejoras graduales en las tecnologías de comunicación y conectividad han permitido nuevas posibilidades técnicas para la adopción de chatbots en diversos sectores, como servicios al cliente, comercio y marketing. El chatbot es una plataforma que utiliza el procesamiento del lenguaje natural, un subconjunto de la inteligencia artificial, para encontrar la respuesta correcta a las preguntas de todos los usuarios y resolver sus problemas. Se ha propuesto una arquitectura de chatbot avanzada que es extensible, escalable y admite diferentes servicios para la comprensión del lenguaje natural (NLU) y canales de comunicación para las interacciones de los usuarios. El documento describe la arquitectura general del chatbot y proporciona los metamodelos correspondientes, así como las reglas para el mapeo entre los metamodelos NLU propuestos y dos de uso común. La arquitectura propuesta podría ampliarse fácilmente con nuevos servicios y canales de comunicación de NLU. Finalmente, dos implementaciones de la arquitectura de chatbot propuesta se demuestran brevemente en el estudio de caso de "ADA" y "COVID-19 Info Serbia".

Palabras clave: chatbot; arquitectura extensible; metamodelo; comprensión del lenguaje natural; estructura; COVID-19

Citación: Matic, R .; Kabiljo, M .; Zivkovic, M .; Cabarkapa, M. Arquitectura de chatbot extensible usando metamodelos de comprensión del lenguaje natural. *Electrónica* **2021**, *10*, 2300. https:// doi.org/ 10.3390 / electronics10182300

Editores académicos: Cataldo Musto y George A. Papakostas

Recibido: 22 de agosto de 2021
Aceptado: 16 de septiembre de 2021
Publicado: 18 de septiembre de 2021

1. Introducción

Las tecnologías sociales, móviles, analíticas, en la nube e Internet de las cosas (SMACIT) y los avances en el campo de la inteligencia artificial (IA) han desafiado fundamentalmente la forma en que las personas trabajan, hacen negocios o se comunican entre sí. Un ejemplo típico de un sistema de IA y uno de los ejemplos más elementales y generalizados de Interacción inteligente entre humanos y computadoras (HCI) es un chatbot [1]. El chatbot es un programa informático que simula una conversación o chat humano, utilizando o no inteligencia artificial, y participa en un diálogo con un humano utilizando lenguaje natural [2]. Los chatbots tienen muchas ventajas para los usuarios y desarrolladores, y su popularidad va en aumento. La mayoría de las implementaciones están disponibles instantáneamente para los usuarios sin necesidad de instalaciones. Los chatbots pueden brindar a los usuarios un soporte rápido y conveniente respondiendo específicamente a sus preguntas, y esta es la razón por la cual la motivación más frecuente para la productividad del uso de los chatbots, mientras que otros motivos son entretenidos, son los factores sociales, la información buena y actualizada y el contacto. con novedad. En lugar de crear una aplicación de máquina inteligente similar a la humana, se trata de crear asistentes digitales efectivos que puedan proporcionar información, responder preguntas, discutir un tema específico o realizar una tarea [3].

Hoy en día, un chatbot puede realizar muchas funciones de aplicaciones móviles o sitios web, todo dentro de la conversación a través de aplicaciones de comunicación, sin necesidad de que el usuario instale o descargue nuevas aplicaciones. Los chatbots ahora están incorporados en asistentes digitales populares como Siri, Cortana, Alexa, Assistant, etc. Ya no es necesario descargar, instalar o abrir aplicaciones solo para realizar operaciones como pedir un producto. Es solo otro aspecto de un mundo multitarea ya conectado.

El uso de chatbots evolucionó rápidamente en numerosos campos en los últimos años, incluidos servicios al cliente, comercio electrónico, marketing, sistemas de apoyo, educación, atención médica, patrimonio cultural y entretenimiento. En todos estos campos, los chatbots han demostrado ser útiles en varios contextos para automatizar tareas y mejorar la experiencia del usuario. Las predicciones adicionales dicen que para 2022, el 80% de las empresas utilizarán chatbots y los bancos podrán automatizar hasta el 90% de la interacción de sus clientes con ellos [4]. Se proyecta que el mercado global de chatbots alcance los 2 mil millones de dólares en 2024, creciendo a una CAGR (tasa de crecimiento anual compuesta) del 29,7% [5].

Una plataforma de chatbot debe tener las siguientes tres partes que realmente agregan experiencia de conversación:

- Procesamiento del lenguaje natural (NLU): comprender la entrada del usuario y extraer información relevante.
- Flujo de conversación: incluida la gestión del contexto de la conversación.
- Cumplimiento de acciones: se utiliza para representar respuestas simples, así como funciones avanzadas como consultas de bases de datos, solicitudes de interfaz de programación de aplicaciones (API) o un disparador lógico personalizado.

La NLU se enfatiza en este documento como el corazón de cada chatbot. La extensibilidad es la capacidad de las arquitecturas de software para pegar arquitecturas externas (hojas) a su estructura central, creando una sinergia entre estas arquitecturas diferentes [6]. En este documento, se ha propuesto la arquitectura de chatbot de microservicio extensible avanzado, que permite cambiar fácilmente de un proveedor de NLU a otro (por ejemplo, debido a un mejor soporte para el idioma serbio o debido a problemas de costos). El marco propuesto orientado a objetos llamado Weaver [7] está diseñado para ser ágil para abordar un entorno increíblemente variable de canales de comunicación modernos para las interacciones de los usuarios. Las novedades de nuestro enfoque son:

- La arquitectura de chatbot propuesta es extensible y admite nuevos servicios de comprensión del lenguaje natural y canales de comunicación para las interacciones del usuario. Gracias a esto, resolvemos la independencia de los marcos de chatbot de todos los proveedores del mercado que se ocupan de la comprensión del lenguaje natural y los canales de comunicación.
- La solución se implementó en dos estudios de caso sobre el idioma serbio. Hasta donde sabemos, nadie se ha ocupado del idioma serbio utilizando diferentes NLU.
- Se basa en los denominados metamodelos como principal mecanismo extensible de Weaver. Para que este mecanismo tenga éxito, proporcionamos:
  - Un metamodelo general de NLU (modelo de un modelo).
  - Metamodelos para los dos servicios NLU específicos (Dialogflow y RASA).
  - Metamodelos correspondientes, así como reglas para un mapeo entre metamodelos NLU genéricos y específicos.

Debido a todo lo especificado anteriormente, creemos que se requiere una arquitectura extensible y escalable para la comprensión del lenguaje natural que resuelva algunos de los problemas definidos anteriormente. Es mucho más fácil hacer metamodelos de cada servicio NLU y mapearlos a nuestro metamodelo general, y así ir más allá de la programación manual para cada nuevo servicio NLU. Los diseñadores definen todos estos metamodelos y las reglas de mapeo correspondientes en la base de datos en el momento del diseño. Usando reglas de mapeo, automáticamente creamos, mantenemos y reenviamos objetos con todos los datos necesarios a los metamodelos NLU (servicios NLU) o metamodelos de comunicación (canales) deseados.

Este artículo está organizado de la siguiente forma: Sección2 aporta motivación para este artículo, así como trabajos relacionados relacionados con el tema. Sección3 presenta el soporte de NLU. En general, la arquitectura lógica se ha presentado con la especificación detallada de los componentes de la arquitectura propuesta. Los metamodelos y sus reglas de mapeo para garantizar la independencia de un servicio de NLU se presentan en la Sección5. Los detalles expuestos en los metamodelos de NLU y las reglas de mapeo correspondientes se explican con un ejemplo específico en la Sección6. Sección7 explica el uso del marco propuesto en dos estudios de caso: ADA y COVID-19 Info Serbia. Finalmente, la conclusión se deriva en la Sección8.

2. Trabajo y motivación relacionados

La introducción de la tecnología chatbot comenzó en 1966 con el programa informático conocido como ELIZA [8]. ELIZA pudo imitar la conversación humana al intentar responder la pregunta de un usuario haciendo coincidir respuestas escritas, es decir, realizando una simple comparación de muestras. Desarrollado por Joseph Weizenbaum en 1956 [9], fue diseñado para emular a un psicoterapeuta y tenía una base de conocimientos en este dominio. En 1995, se desarrolló el chatbot ALICE (Entidad informática de Internet lingüística artificial). ALICE se basa en un algoritmo simple de coincidencia de patrones con la inteligencia subyacente basada en el Lenguaje de marcado de inteligencia artificial (AIML) [10], lo que permite a los desarrolladores definir los componentes básicos del conocimiento del chatbot [11]. En la literatura científica, artículos académicos y actas de congresos, se han analizado diferentes aspectos de la tecnología y aplicaciones de chatbot [12,13].

Dentro del estudio de las tecnologías de chatbot, una dirección especial de interés está relacionada con el marco de modelado de chatbot. Los marcos de los chatbots existentes proporcionan arquitecturas y herramientas específicas de chatbots. Algunos son muy simples y otros utilizan conjuntos de datos lingüísticos masivos de servicios web, que tienen modelos muy precisos obtenidos gracias a muchos años de experiencia y grandes cantidades de datos. Los más populares incluyen Dialogflow, RASA, Microsoft Bot Framework, Botkit, Pandorabot y WIT.ai. El marco de chatbot se ha logrado principalmente mediante la definición de entidades, intenciones y respuestas dentro de una plataforma específica, proporcionando una interfaz y una gran capacidad de comprensión del lenguaje natural. Autores Mu y Sarkar [14] discuten algunos inconvenientes de NLU y encuentran que los sistemas de lenguaje natural restringidos pueden funcionar mejor que el lenguaje natural completo. De acuerdo con esto, el modelo propuesto en este trabajo se basa en el dominio cerrado.

Se escribió un gran estudio sobre los tipos de arquitecturas de chatbot en [13]. En este y otros artículos similares [3,15-25], nadie explica cómo superar el problema de la integración con diferentes servicios de NLU o con diferentes plataformas de comunicación. Se han propuesto varios enfoques para simplificar el proceso de desarrollo de chatbot y mejorar su mecanismo de respuesta [15,26,27]. Es muy difícil integrar la arquitectura de chatbot con otros servicios externos sin codificación manual, y los diseñadores con nuevas plataformas de comunicación o servicios de NLU no pueden extenderlo fácilmente [15]. La mayoría de las arquitecturas de chatbots solo ofrecen una API para consultar los resultados de la intención e integrarse con el servicio NLU. Del mismo modo, aunque cada uno de ellos admite diferentes canales de comunicación, no suelen ofrecer opciones de extensión. Weaver está diseñado para resolver estos problemas.

Xatkit [26] (anteriormente conocido como Jarvis [15]) y Conga [27] son obras en parte similares a las nuestras. Usan una arquitectura basada en modelos (MDA) para desarrollar chatbots. Aunque nuestros enfoques y arquitecturas de chatbots son diferentes de estos dos artículos, muestran la realización de la idea de desarrollar chatbots que sean independientes de los servicios de NLU. Sin embargo, no usamos MDA para desarrollar chatbots como ellos, sino que usamos uno de los conceptos básicos del enfoque MDA (modelos). Para evitar la dependencia del proveedor, nuestra arquitectura de chatbot consta de metamodelos para los servicios de NLU, y se asignan a nuestras reglas generales de asignación de metamodelos de NLU. Notará que solo usamos NLU, que es una parte del procesamiento del lenguaje natural (NLP), y tenemos una lógica diferente del uso del contexto y la acción de ejecución. Los metamodelos de Xatkit y Conga se utilizan para fines diferentes a los nuestros, porque incluyen conceptos como contexto, acción, plataformas, etc. Usamos una técnica más simple para proporcionar una arquitectura de chatbot independiente de NLU extensible. En nuestro trabajo, toda la lógica relacionada con el contexto que afecta la conversación de flujo o la ejecución de una determinada acción se encuentra en los componentes que se explicarán en el apartado4.2. Sin embargo, a excepción de una explicación detallada de la arquitectura del chatbot de microservicio, la principal contribución de este documento es la fácil integración con los servicios externos de NLU porque estas son probablemente las partes más importantes de la arquitectura del chatbot que son difíciles de implementar por su cuenta. Por lo tanto, es bueno utilizar servicios externos. Sin embargo, esto no descarta la posibilidad de crear nuestro propio servicio NLU. Se puede integrar de la misma manera fácil que se muestra aquí en el documento sin programación manual, gracias a los metamodelos y las reglas de mapeo propuestas.

Xatkit presenta un marco de modelado de chatbot multiplataforma que utiliza metamodelos y lenguaje específico de dominio textual (DSL). Este marco desacopla la parte de modelado de chatbot de los aspectos específicos de la plataforma, aumentando la reutilización. DSL proporciona primitivas para diseñar las intenciones del usuario, la lógica de ejecución y la plataforma de implementación. Actualmente solo admiten Dialogflow como un servicio de NLU. Nuestra implementación ha demostrado ser independiente de los servicios de Dialogflow y Rasa NLU. En nuestro enfoque, tenemos bifurcaciones condicionales y admitimos eventos genéricos (webhooks) que nos permiten crear un chatbot reactivo que puede escuchar y responder activamente. Nuestro marco mejora la extensibilidad y la personalización al proporcionar métodos de webhook explícitos y elementos de arquitectura que le permiten extender su interfaz estable. A diferencia de nosotros, su metamodelo del Intent Package contiene un contexto de colección. Por otro lado, no vemos en su metamodelo cómo anotan roles de entidad y agrupaciones de entidades. Con los roles de entidad, podemos definir entidades con roles específicos en el enunciado. Las agrupaciones de entidades permiten que las entidades se agrupen con una etiqueta de grupo específica que define diferentes órdenes. Conga presenta una solución basada en modelos para la ingeniería de chatbot hacia adelante y hacia atrás, con un sistema de recomendación que ayuda a seleccionar las herramientas de desarrollo de chatbot más adecuadas. Comprende un metamodelo neutral y un DSL para la generación de código y analizadores para varias plataformas de chatbot. La principal y más importante diferencia es que nuestros chatbots definidos son ejecutables al proporcionar un motor de ejecución. Conga tampoco admite conceptos específicos de la plataforma, como la acción de los botones. Aunque todo su metamodelo también carece de elementos de rol y grupo, es difícil comparar el resto del metamodelo porque los propósitos de sus metamodelos son diferentes a los nuestros. Aunque MDA tiene muchas ventajas en el desarrollo de software, dicho desarrollo a menudo está limitado por el tipo de herramienta que utilizan. Conga ya cuenta con 15 herramientas de generación. Solo son flexibles en las partes del marco cubiertas por el DSL utilizado. El uso de los marcos de metamodelo propuestos en nuestro modelo es lo suficientemente extensible y adaptable para incluir nuevos servicios y canales de comunicación de NLU. Weaver proporciona aprendizaje interactivo que facilita a los desarrolladores el desarrollo de flujos conversacionales, pero esto está fuera del alcance de este documento. es difícil comparar el resto del metamodelo porque los propósitos de sus metamodelos son diferentes a los nuestros. Aunque MDA tiene muchas ventajas en el desarrollo de software, dicho desarrollo a menudo está limitado por el tipo de herramienta que utilizan. Conga ya cuenta con 15 herramientas de generación. Solo son flexibles en las partes del marco cubiertas por el DSL utilizado. El uso de los marcos de metamodelo propuestos en nuestro modelo es lo suficientemente extensible y adaptable para incluir nuevos servicios y canales de comunicación de NLU. Weaver proporciona aprendizaje interactivo que facilita a los desarrolladores el desarrollo de flujos conversacionales, pero esto está fuera del alcance de este documento. es difícil comparar el resto del metamodelo porque los propósitos de sus metamodelos son diferentes a los nuestros. Aunque MDA tiene muchas ventajas en el desarrollo de software, dicho desarrollo a menudo está limitado por el tipo de herramienta que utilizan. Conga ya cuenta con 15 herramientas de generación. Solo son flexibles en las partes del marco cubiertas por el DSL utilizado. El uso de los marcos de metamodelo propuestos en nuestro modelo es lo suficientemente extensible y adaptable para incluir nuevos servicios y canales de comunicación de NLU. Weaver proporciona aprendizaje interactivo que facilita a los desarrolladores el desarrollo de flujos conversacionales, pero esto está fuera del alcance de este documento. Conga ya cuenta con 15 herramientas de generación. Solo son flexibles en las partes del marco cubiertas por el DSL utilizado. El uso de los marcos de metamodelo propuestos en nuestro modelo es lo suficientemente extensible y adaptable para incluir nuevos servicios y canales de comunicación de NLU. Weaver proporciona aprendizaje interactivo que facilita a los desarrolladores el desarrollo de flujos conversacionales, pero esto está fuera del alcance de este documento. Conga ya cuenta con 15 herramientas de generación. Solo son flexibles en las partes del marco cubiertas por el DSL utilizado. El uso de los marcos de metamodelo propuestos en nuestro modelo es lo suficientemente extensible y adaptable para incluir nuevos servicios y canales de comunicación de NLU. Weaver proporciona aprendizaje interactivo que facilita a los desarrolladores el desarrollo de flujos conversacionales, pero esto está fuera del alcance de este documento.

3. Soporte de NLU

NLU es un subconjunto de PNL y AI conversacional. Ayuda a las computadoras a comprender el lenguaje humano al comprender, analizar e interpretar mensajes básicos o partes del habla. NLU está entrenado con enunciados naturales del usuario etiquetados con entidades y difundidos con el uso de sinónimos. El concepto de utilizar la IA y el lenguaje humano para realizar la comunicación con las máquinas no es nuevo; sin embargo, los investigadores han subestimado la complejidad de los lenguajes humanos durante años, incluso décadas. Incluso hoy en día, tras los recientes avances en IA y PNL, se han realizado varios estudios que demuestran que los usuarios no pueden ser engañados y siempre sabrán que se están comunicando con una computadora y no con otro ser humano [28]. Recientemente, se han empleado chatbots impulsados por IA para diversas tareas desde diferentes dominios de aplicación, como medicina y autodiagnóstico [29] y apoyo al servicio de estudiantes universitarios [30,31].

En la primera versión de nuestra plataforma Weaver, elegimos Dialogflow como el servicio NLU. Cuando vimos algunas deficiencias, presentamos RASA como el servicio NLU de código abierto más confiable en términos de puntuación de confianza [32]. Después de eso, nos dimos cuenta del problema de la dependencia de NLU y luego se nos ocurrió la idea que hemos implementado en nuestra arquitectura. Dialogflow y RASA son dos servicios NLU diferentes con el mismo propósito. Cada nuevo servicio de NLU se aplicaría de manera similar en nuestra solución general propuesta. Explicaremos sus diferencias y los motivos para elegirlos como ejemplo para nuestra arquitectura.

Ambos servicios de NLU utilizan NLU basado en ML y están entrenados con expresiones de usuario naturales etiquetadas con entidades. Dialogflow tiene una validación, lo que indica que hay una intención específica que necesita más datos de entrenamiento. También se le han proporcionado entidades del sistema como números, fecha, hora, etc. RASA, por otro lado, tiene una selección de canalización personalizada basada en la cantidad de datos de entrenamiento. Podemos usar el modelo pre-entrenado y usando más datos, podemos entrenar desde cero eligiendo una canalización. RASA proporciona funcionalidad para

evaluar intenciones y entidades. RASA también tiene una opción de importación de datos de entrenamiento para importar datos en diferentes formatos y de diferentes fuentes. En Dialogflow, no tenemos la opción de importar o usar datos de entrenamiento, pero acepta una línea por enunciado como datos de entrenamiento. Ambas plataformas brindan soporte en varios idiomas. Para admitir el idioma serbio en Dialogflow, debemos agregar Google Translator a nuestra arquitectura. RASA nos permite agregar este soporte con la ayuda de spaCy como componentes incorporados. Para los idiomas que no tienen incrustaciones de palabras previamente entrenadas, RASA sugiere una canalización sin la biblioteca spaCy. Las incrustaciones de palabras previamente entrenadas son útiles porque ya codifican el conocimiento lingüístico. Para la mayoría de los idiomas utilizados, es una biblioteca muy útil porque puede "comprender" grandes volúmenes de texto que se utilizan para construir sistemas de comprensión del lenguaje natural. Desafortunadamente, para el serbio, no hay incrustaciones de palabras previamente entrenadas, y esta es la razón por la que no usamos la biblioteca spaCy. Para apoyar al serbio, usamos una tubería NLU ajustable. La canalización consta de diferentes componentes que trabajan secuencialmente para procesar la entrada del usuario en una salida estructurada. Hay componentes para clasificación de intenciones, reconocimiento de entidades, tokenización, preprocesamiento, etc. Cada componente procesa una entrada y / o crea una salida. El orden de los componentes se especifica en un archivo de configuración. Probamos varios archivos de configuración con la prueba RASA. Para aprovechar al máximo nuestros datos de entrenamiento, entrenamos y evaluamos nuestro modelo en diferentes canales y diferentes cantidades de datos. Repitiendo este proceso con diferentes porcentajes de datos de entrenamiento, Intentamos comprender cómo se comportaba cada canal aumentando la cantidad de datos de entrenamiento. Todo el proceso se realizó cinco veces para cada configuración especificada.

RASA es de código abierto, donde puede usarlo sin ningún costo. RASA se puede implementar localmente y en la nube. Dialogflow almacena e implementa modelos en la nube de Google. RASA ha agregado Representación de codificador bidireccional de Transformer en la tubería, lo que ayuda a crear mejores modelos. RASA también proporciona una ventaja adicional de agregar su propio modelo personalizado a la tubería para cualquier tarea. Dialogflow es una gran plataforma con buenos modelos y entidades previamente entrenadas, pero no admite modelos personalizados. RASA también ha agregado soporte para Tensorboard 2, que se utiliza para visualizar métricas de entrenamiento. Ayuda a comprender si el modelo se ha entrenado correctamente y podemos realizar cambios en los hiperparámetros en función de las métricas para mejorar el modelo.

4. Arquitectura propuesta para el chatbot

La arquitectura lógica de la solución propuesta, llamada Weaver, se basa en el modelo de chatbot de referencia propuesto por los autores E. Adamopoulou y L. Moussiades [33]. Esta arquitectura se puede utilizar para desarrollar una amplia gama de chatbots. Permite la comprensión, implementación, mantenimiento y mayor desarrollo de un chatbot independiente de la plataforma.

En este documento, describimos nuestro enfoque para construir un chatbot basado en IA. Generalmente, existen dos tipos de modelos que prevalecen en el desarrollo de chatbot basados en IA [13]:

- Modelos basados en la recuperación: utilizan un repositorio de respuestas predefinidas o utilizan algún tipo de heurística para elegir la respuesta adecuada en función de la consulta y el contexto. Pueden proporcionar respuestas más confiables y gramaticalmente precisas. Son más fáciles de enseñar ya que requieren menos información, pero no pueden responder a preguntas más allá de su base de conocimientos.
- Modelos de generación: estos no se basan en respuestas predefinidas. Generan nuevas respuestas desde el principio, ya que pueden responder a preguntas ambiguas. Estos chatbots se vuelven más inteligentes con el tiempo y pueden aprender de preguntas, respuestas y conversaciones anteriores. Sin embargo, son difíciles de entrenar ya que exigen una gran cantidad de datos y los modelos de recuperación a menudo disfrutan de un mejor control sobre la calidad de la respuesta que los modelos generativos.

Weaver usa la conversación basada en la recuperación. Nuestro modelo basado en la recuperación utiliza técnicas de NLU para predecir la intención más precisa. Después de eso, Weaver toma las respuestas de un conjunto cerrado de respuestas usando una lista de salida clasificada de posibles respuestas. La arquitectura del chatbot independiente de NLU se define a través de tres partes principales: (1) un metamodelo genérico que define los conceptos generales de NLU y sus relaciones, (2) un metamodelo genérico de NLU específico

servicio (por ejemplo, Dialogflow, RASA), y (3) un conjunto de reglas de mapeo que mapea cada concepto de servicio NLU específico al concepto de metamodelo genérico. La plataforma Weaver simplemente sigue estas reglas de mapeo y luego ejecuta la lógica general del chatbot.

Aquí se propone, la plataforma de IA conversacional proporciona un conjunto de características analíticas que ayudan a conocer el número de sesiones, intenciones, número de usuarios, usuarios recurrentes, número de preguntas respondidas y no respondidas, y el flujo de preguntas. La herramienta de administración Weaver nos permite crear escenarios de chatbot por nuestra cuenta en sencillos pasos, desde definir nuevos escenarios hasta agregar intenciones y respuestas definitivas. Nos damos cuenta de las eficiencias en el desarrollo de la arquitectura del chatbot al explotar las herramientas del marco y las instalaciones de modelado. La arquitectura lógica y la estructura de componentes del Weaver BotFramework implementado se dan a continuación, con explicaciones detalladas de cada componente.

4.1. Arquitectura

La arquitectura lógica del modelo propuesto en este trabajo se muestra en la Figura 1. Como puede verse, la arquitectura consta de cuatro bloques de construcción principales:

- Usuario.
- Plataformas de comunicación (canales) a través de las cuales se comunican los usuarios.
- Weaver: plataforma de IA conversacional.
- Servicios externos entre los que NLU es el más importante.

La plataforma de chatbot incluye BotFramework y dos conectores: conectores de mensajería API y conectores de mapeo externo. Entre otras cosas, esta plataforma también sirve para la conexión, construcción, prueba y despliegue del chatbot inteligente. La comunicación entre el BotFramework y ambos conectores (mensajero y externo) se realiza mediante Hypertext Transfer Protocol Secure (HTTPS). Toda la plataforma se basa en microservicios. La validación y el mapeo de los mensajes que se reciben desde una plataforma de comunicación se realizan en API Messenger Connector. Esto tiene como objetivo preparar el mensaje, que se reenviará al siguiente componente en función del tipo de mensaje recibido del mensajero.

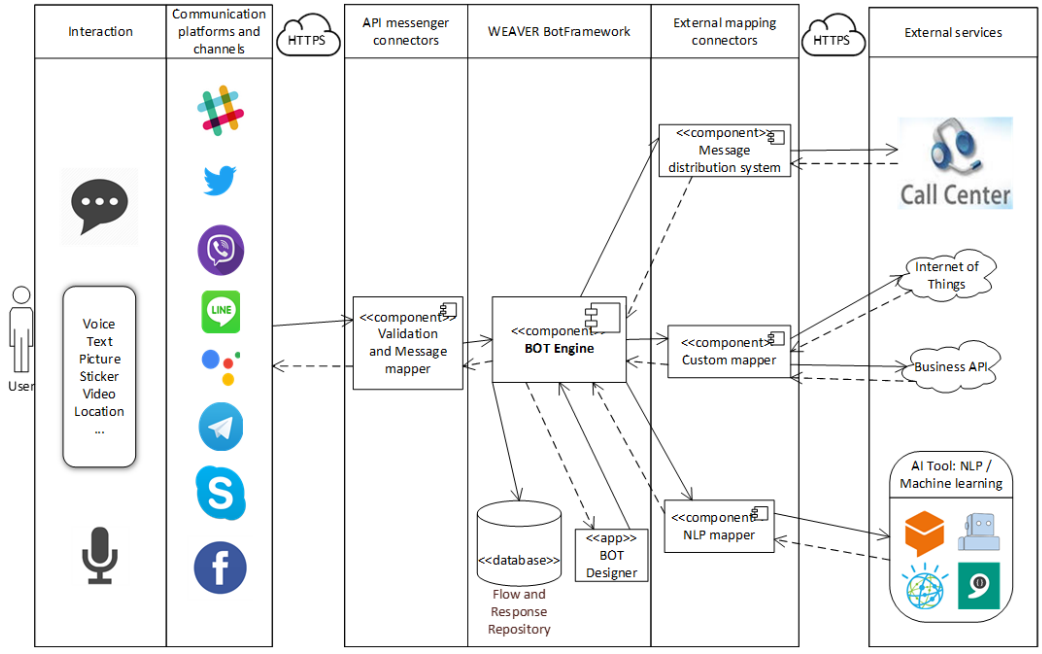

Figura 1. La arquitectura lógica de la solución propuesta.

El ciclo de vida del mensaje comienza y termina en el conector, ya que su función es recibir el mensaje y devolver el mensaje del chatbot al canal desde el que se envió el mensaje. Para que el mensaje se reciba desde la plataforma de comunicación a nuestro sistema, debemos configurar el método webhook. Para ello utilizamos el conector de mensajería, que contiene los métodos definidos que aceptan callback por parte de la plataforma de comunicación. Nuestro conector de mensajería primero debe estar suscrito a los eventos de mensajería de la plataforma de comunicación aprobada y configurar un localizador uniforme de recursos (URL) de devolución de llamada. La plataforma de chatbot podrá recibir webhooks enviados desde las plataformas de comunicación. La primera función del componente conectado es reconocer desde dónde se envió el mensaje, es decir, desde qué plataforma de comunicación ocurrió el evento, haciendo que el sistema reciba el mensaje. Este componente lo reconoce simplemente a través de la URL. Según los datos de la URL, el componente reconoce desde qué plataforma de comunicación se recibió el mensaje y recupera las reglas para la validación y el mapeo del contexto que llegó a la solicitud. En base a las reglas se comprueba la validez del mensaje, es decir, si el mensaje fue enviado por la plataforma de comunicación a la que nos hemos suscrito o no. Si el mensaje es válido, se puede reenviar al módulo de mapeo. En esta parte, el mensaje se mapea en el objeto con el que opera BotFrameworks utilizando el metamodelo y las reglas de mapeo definidas. Los conectores de Messenger deben garantizar el mapeo adecuado de todos los tipos de mensajes para una plataforma de comunicación en particular. Los objetos reenviados al BotFramework también contienen datos de la plataforma de comunicación y el usuario, de modo que el mensaje se puede reenviar correctamente al usuario correcto. Dicho mapeo también funciona en la dirección opuesta, es decir, cuando el BotFramework debería devolver el mensaje hacia una determinada plataforma de comunicación. Los metamodelos y las reglas de mapeo de las plataformas de comunicación no forman parte de este trabajo; queremos centrarnos en los metamodelos y las reglas de mapeo de los servicios NLU explicados en la Sección5. El BotFramework consta de muchos componentes que hacen que este sistema sea escalable y es compatible con la mayoría de los criterios para construir un buen marco empresarial, sugeridos en [34]. Este marco es la parte central y principal de la arquitectura que contiene toda la lógica para procesar los mensajes recibidos, trabaja con contextos, flujo de chat, toma de decisiones lógicas, integración con escenarios y respuestas predefinidos, y varias otras reglas requeridas. Contiene varios componentes importantes como motor de bot, diseñador de bot, procesamiento de lenguaje, motor de conversación, etc. Cada mensaje recibido se procesa por separado y pasa por una serie de etapas. Cada usuario tiene su propia sesión de chat (conversación) que mantiene un registro de todos los mensajes entre los usuarios y el chatbot. Todas las instancias de escenarios iniciadas, suspendidas y completadas después de su definición de escenarios también se ven aquí. Por lo tanto, cada sesión de chat representa uno o más escenarios entre usuarios y chatbots que contiene contextos que afectan el flujo de chat actual y futuro. El motor bot es un componente complejo que constituye la parte central del BotFramework y consta de varios componentes menores que se detallarán en la Sección4.2.

Los conectores de mapeo externos sirven para la conexión con servicios externos, como los servicios NLU. Estos conectores también tienen reglas de mapeo predefinidas independientes para cada tipo e implementación específicos. Además de los conectores predefinidos que funcionan con los servicios antes mencionados, también existen conectores personalizados para datos externos necesarios en la conversación, en los que se pueden cambiar los mapeos en función de la necesidad de realizar con éxito cualquier conversación. Los conectores de mapeo externo también contienen componentes que exponen los servicios a otros sistemas que participan en el proceso de comunicación con el usuario, por ejemplo, un centro de contacto que podría hacerse cargo de la conversación del usuario. El BotFramework se puede integrar con la aplicación del centro de contacto existente con la ayuda del componente Conector de distribución de mensajes. En el caso de preguntas complejas, el usuario es redirigido y chatea con el operador desde el centro de contacto correspondiente. La extensibilidad del marco propuesto simplifica la integración de otros servicios externos útiles. El BotDesigner es una herramienta visual equipada con el Software Development Kit (SDK) para la plataforma de implementación correspondiente. El SDK ofrece posibilidades que facilitan las interacciones de chatbotuser. Usando SDK, es posible enseñar directamente a la NLU para evitar la dependencia

en la interfaz NLU del proveedor. Además, BotDesigner permite la generación visual del escenario, es decir, su flujo. El flujo de la conversación mapea todas las direcciones potenciales en las que puede desarrollarse la discusión, con muchas ramas para todas las posibilidades. El flujo de conversación es responsable de predecir todos los posibles mensajes de entrada y reacciones del chatbot. Esto ayuda a los usuarios a alcanzar rápidamente sus objetivos y a obtener información más fácilmente y a dar prioridad a las necesidades comerciales. El BotDesigner es intuitivo y fácil de entender y usar. Todos los datos necesarios para la gestión del flujo de escenarios y las respuestas basadas en el contexto, las intenciones identificadas y las entidades se encuentran en el repositorio de flujos y respuestas. Esta base de datos se utiliza para almacenar todos los escenarios predefinidos con sus acciones y condiciones, respuestas con la segmentación necesaria, conversaciones, usuarios y logs. Se adapta a todas las configuraciones relacionadas con conectores, componentes y configuraciones generales del sistema. Todos los metamodelos y reglas de mapeo también se almacenan en esta base de datos.

4.2. La estructura de componentes del motor de bots

En la Figura se muestra una representación detallada del motor Bot 2. Consiste en una serie de componentes que están interconectados. El motor central (CE) es un componente independiente que es la parte central de este motor. El motor central es un componente que administra otros componentes dentro del BotFramework. Debe conocer la secuencia correcta de ejecución de la acción, rastrear la ejecución de acciones y errores, enviar notificaciones, etc. Dependiendo de la configuración, cada acción pasa por debug e info log para seguir el flujo de un componente tan complejo.

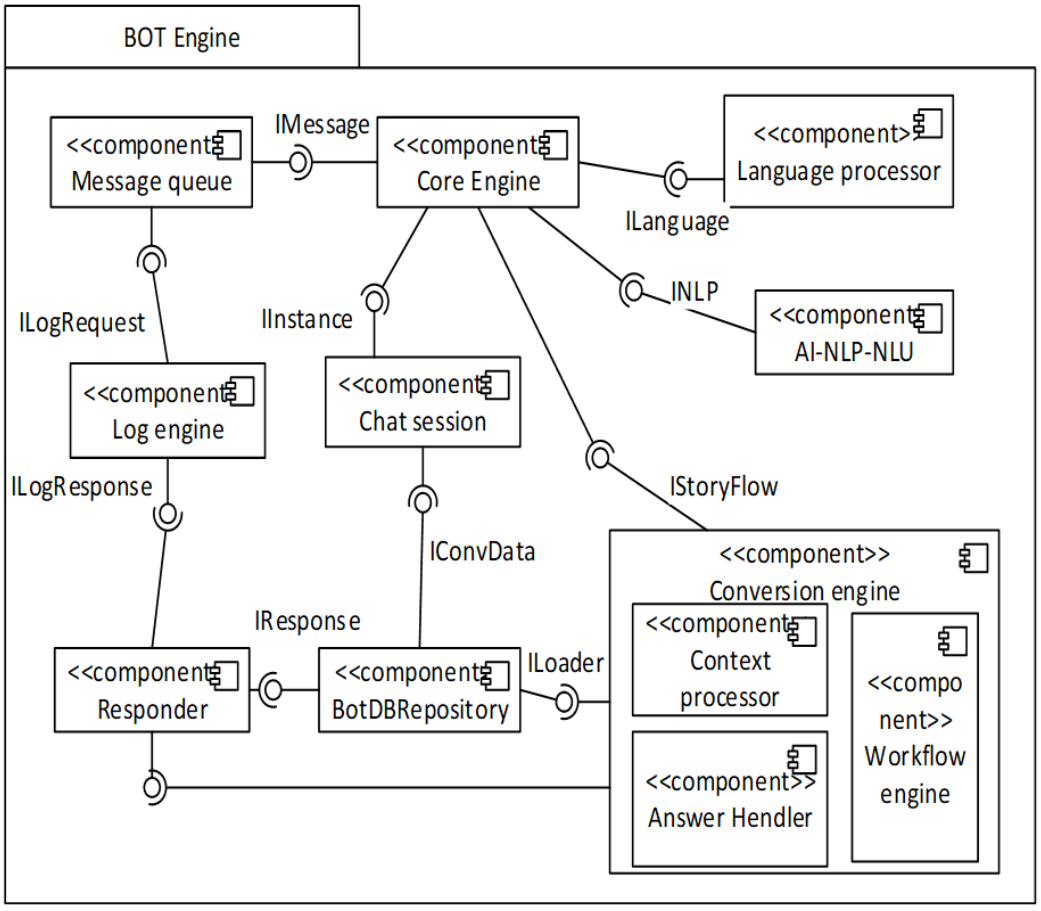

Figura 2. Diagrama de componentes del motor bot.

En el componente Cola de mensajes, se presta atención a la secuencia de mensajes que llegan al canal y su liberación para procesamiento. Todos los mensajes que ingresan y salen del motor Bot se monitorean en el sistema de registro usando este componente. Cada sesión de chat tiene su vida

ciclo que sigue al menos una definición de un escenario creado a través de BotDesigner. Cuando el usuario inicia un escenario, el componente Sesión de chat crea una instancia del escenario para ese usuario. Tiene la posibilidad de ofrecer contextos reconocidos de la conversación anterior como contexto para la conversación actual (relleno de contexto) y tiene la opción de reconocer si el usuario quería cambiar algún valor de contexto utilizado en la conversación anterior (cambio de contexto). El usuario puede cambiar el idioma en el que se dirige al chatbot en cualquier momento, y el componente Language Processor es responsable de comprender el idioma cambiante. Si es necesario, este componente debe asegurar la mejor traducción posible al idioma adecuado para que el componente AI-NLP-NLU pueda procesar el texto. Basado en la intención y el contexto, La tarea del motor de conversación es decidir qué escenario debe procesar y si es un escenario nuevo o una continuación de un escenario existente. El procesamiento de contexto realizado por el Procesador de contexto también se realiza aquí. Se han incorporado mecanismos de flujo de trabajo en el marco sugerido. El motor de flujo de trabajo tiene la tarea de cuidar cada escenario que sigue su definición de escenario, es decir, la secuencia de acción de la instancia de escenario en curso. Cada acción tiene contextos, que se supone deben cumplirse para que se ejecute. Además, cada acción tiene la opción de validación incorporada o personalizada para validar la entrada del usuario y omitir el procesamiento del componente AI-NLP-NLU. Un ejemplo de esto es cuando una acción en algunos escenarios requiere un número, correo electrónico, y teléfono que se puede validar omitiendo los componentes para traducción y AI-NLP-NLU. El componente Answer Handler tiene la tarea de proporcionar respuestas adecuadas para acciones dentro del escenario. Las respuestas pueden ser de diferentes tipos: texto, imagen, URL, ubicación, es decir, unión de tipos definida por la plataforma de comunicación. Se pueden definir múltiples respuestas del mismo tipo o de diferentes tipos para una determinada acción en la secuencia adecuada en la que se entregarán. El componente Responder prepara el mensaje para enviarlo de vuelta al conector de mensajería para reenviarlo al usuario. Dado que el conector se presenta como un microservicio, este componente se dirige al conector a través del servicio RESTful del Protocolo de transferencia de hipertexto (HTTP) y reenvía el mensaje preparado. Muchos componentes de un marco de bot dependen del almacenamiento y el acceso a la base de datos. Por ejemplo, el motor de conversación necesita conocer las definiciones de escenarios y sus instancias. La sesión de chat maneja a todos los usuarios y sus comunicaciones. También es muy importante recordar todos los mensajes, entidades e intenciones. Todos los metamodelos y las reglas de mapeo correspondientes también son importantes y deben almacenarse en la base de datos. La tarea del componente BotDBRepository es responsable de todas las interfaces necesarias para la comunicación con la base de datos.

NLU tiene problemas desafiantes que exigen una experiencia significativa. Esta es una de las razones por las que las empresas eligen con frecuencia no crear su propia solución sino utilizar otras plataformas de servicios. Microsoft LUIS.ai [35], Facebook Wit.ai [36], Google Dialogflow [37], RASA [38] e IBM Watson [39] son algunas de las muchas plataformas de servicios populares de NLU. Aceptan lenguaje natural y devuelven datos estructurados (datos divididos en un formato simple y organizado para un procesamiento simple, por ejemplo, JSON, notación de objetos JavaScript). Estas empresas han convertido muchos años de experiencia en plataformas que brindan buenas soluciones a empresas más pequeñas o más grandes para el aprendizaje automático. Nuestra arquitectura apoya precisamente este enfoque a través de nuestro componente AI-NLP-NLU. Un enunciado llega a ingresar a este componente para que la intención del usuario pueda ser enviada a procesamiento y detectada con todos los parámetros necesarios. Cada servicio externo de NLU tiene sus propias reglas y asignaciones que se aplican al crear solicitudes para la serialización de servicios y respuestas. Todas estas asignaciones se configuran al agregar servicios para NLU. La tarea principal del componente es reenviar las intenciones y entidades detectadas de los servicios externos en la forma adecuada que entiende el motor del Bot para que pueda continuar con el proceso. Tiene la posibilidad de tener N servicios que estarán en modo único / multimodo, es decir, modo activo / pasivo, que pueden ayudar a una mejor detección de intenciones. Si hay dos servicios NLU en el modo múltiple, la solicitud de procesamiento se reenvía en paralelo a ambos servicios. Se toman en consideración dos respuestas y la respuesta que mejor cumple con los criterios establecidos continúa el procesamiento. Una de las funciones principales de este componente es la detección de intenciones y entidades. Además de este rol, este Tiene la posibilidad de tener N servicios que estarán en modo único / multimodo, es decir, modo activo / pasivo, que pueden ayudar a una mejor detección de intenciones. Si hay dos servicios NLU en el modo múltiple, la solicitud de procesamiento se reenvía en paralelo a ambos servicios. Se toman en consideración dos respuestas y la respuesta que mejor cumple con los criterios establecidos continúa el procesamiento. Una de las funciones principales de este componente es la detección de intenciones y entidades. Además de este rol, este Tiene la posibilidad de tener N servicios que estarán en modo único / multimodo, es decir, modo activo / pasivo, que pueden ayudar a una mejor detección de intenciones. Si hay dos servicios NLU en el modo múltiple, la solicitud de procesamiento se reenvía en paralelo a ambos servicios. Se toman en consideración dos respuestas y la respuesta que mejor cumple con los criterios establecidos continúa el procesamiento. Una de las funciones principales de este componente es la detección de intenciones y entidades. Además de este rol, este Una de las funciones principales de este componente es la detección de intenciones y entidades. Además de este rol, este Una de las funciones principales de este componente es la detección de intenciones y entidades. Además de este rol, este

El componente tiene otros roles administrados en la herramienta para la administración del servicio externo de NLU. Cada servicio NLU tiene su propio metamodelo y reglas de mapeo, que son utilizadas por el componente AI-NLP-NLU para llenar el objeto con todos los datos necesarios y ayudar a ser independiente de un solo proveedor de servicios NLU.

5. Metamodelos de NLU y reglas de mapeo correspondientes

NLU es un motor de comprensión del lenguaje natural que clasifica las expresiones por intenciones y extrae información relevante de las expresiones llamadas entidades. La función principal de NLU es tratar de comprender la entrada del usuario y la intención de la conversación y extraer las entidades necesarias para realizar una acción de usuario o lógica empresarial. Podemos pensar en los servicios de NLU como un conjunto de API de alto nivel para crear nuestro propio analizador de idiomas utilizando bibliotecas de aprendizaje automático y NLU existentes. Para ser independientes de un solo servicio externo de NLU, desarrollamos nuestro propio metamodelo de NLU, llamado metamodelo DORIUS, extracto que se muestra en la Figura3. La razón principal para desarrollar nuestra propia versión de metamodelo es permitir correspondencias más fáciles entre los conceptos de dos servicios NLU muy populares (Dialogflow y RASA). Mostramos cómo nuestro metamodelo NLU se asigna a los metamodelos de Dialogflow y RASA. Las asignaciones definidas deben seguir reglas y restricciones, que están definidas por el metamodelo de asignación [40]. El extracto de los metamodelos de Dialogflow y RASA se muestra en las figuras4 y 5. Estos también son metamodelos originales desarrollados por los autores de este artículo. Las versiones de Dialogflow y RASA que usamos en nuestro enfoque para desarrollar metamodelos se describen oficialmente en [37,38]. Todos estos metamodelos y las reglas de mapeo correspondientes se definen en la base de datos en el momento del diseño. El diseñador ingresa datos de entrenamiento en los conceptos NLU del metamodelo DORIUS para cada enunciado. Los datos de entrenamiento de NLU consisten en expresiones de usuario de ejemplo categorizadas por intención. Cuando los clientes del chatbot deciden qué servicios de NLU quieren utilizar, el metamodelo de NLU correspondiente se carga automáticamente desde el metamodelo de DORIUS. Usando reglas de mapeo, automáticamente creamos, mantenemos y reenviamos objetos con todos los datos de entrenamiento necesarios a los metamodelos NLU correspondientes. Después de eso, se publican los datos de entrenamiento para el chatbot de aprendizaje. Los resultados de estas acciones son servicios NLU que se entrenan con enunciados naturales de usuario etiquetados con entidades. Durante el tiempo de ejecución, los servicios de NLU analizan el texto (enunciado) según las técnicas de aprendizaje automático, compara y busca una coincidencia. Los servicios de NLU devuelven la intención y las entidades al componente AI-NLP-NLU con el porcentaje de precisión de la comprensión de la intención. Como ya hemos dicho, la plataforma asegura que se puedan utilizar múltiples servicios NLU, pero el componente Core Engine elige el que tiene una mayor probabilidad de precisión. Usando estos metamodelos propuestos en nuestro enfoque, también es fácil incluir nuevos servicios de NLU. Todo lo que necesita es crear un metamodelo y las reglas de mapeo adecuadas para el metamodelo DORIUS, que ampliará automáticamente la elección de los servicios de NLU. Además, el metamodelo DORIUS puede ayudar al antiguo chatbot a cambiar a otro servicio de NLU. Suponiendo que hay un chatbot que solo usa el servicio Dialogflow, Con el metamodelo DORIUS es posible cambiar fácilmente el chatbot de Dialogflow al servicio RASA y viceversa. En este uso, los conceptos de DORIUS se cargan desde el metamodelo de Dialogflow usando reglas de mapeo, y luego sus conceptos se mapean al metamodelo RASA.

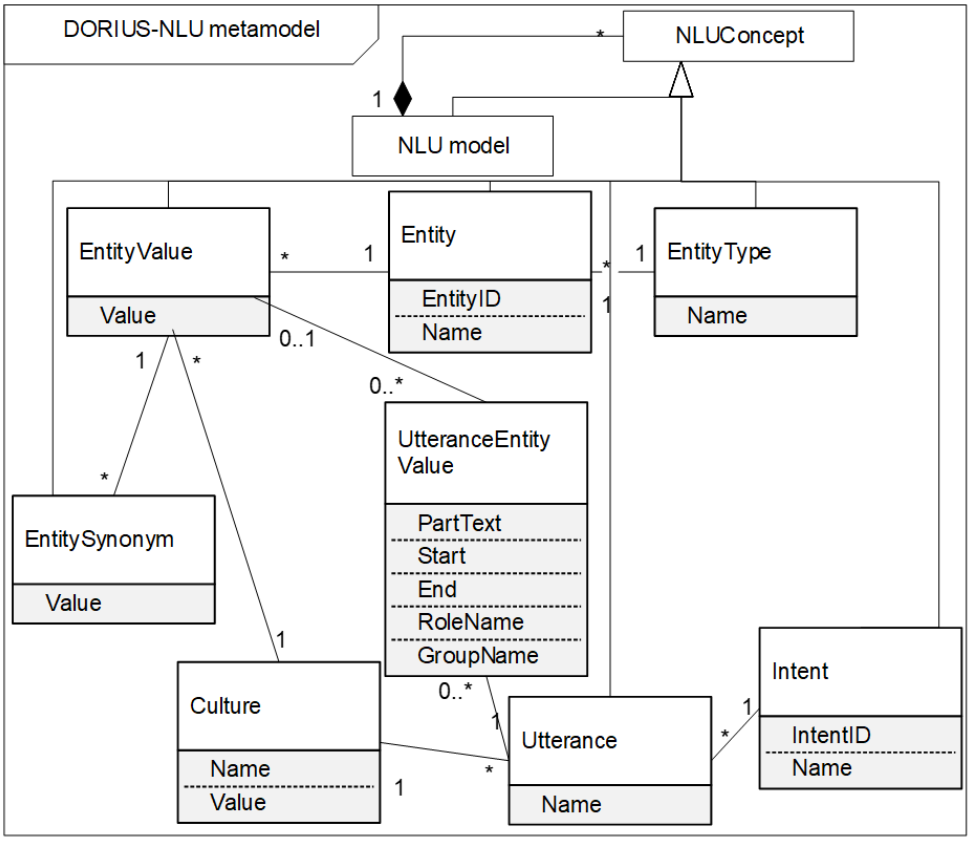

Figura 3. Metamodelo DORIUS.

El concepto DORIUS NLU representa el concepto más abstracto en el modelo de datos NLU. Está especializado utilizando conceptos de NLU que son más concretos:

- Intención categoriza la intención de un usuario en una conversación. Una intención es un grupo de expresiones con un significado similar. La intención se refiere al objetivo que el cliente tiene en mente cuando escribe una pregunta o comentario. Attribute IntentID representa un identificador único de intención. Nombre de atributo es el nombre de la intención. Por ejemplo, si el Intent Name es ProfessorConsultation, el IntentID podría ser Guid123.
- EntityType representa un tipo de entidad. El nombre del atributo puede ser sistémico o personalizado. Por ejemplo, EntityType puede ser personalizado.
- Entidad representa la entrada de entidad para un EntityType asociado. La entidad puede ser un EntityType sistémico o personalizado. Las entidades se anotan en los ejemplos de formación con el nombre de la entidad. Las entidades son un mecanismo para identificar y extraer datos útiles de las entradas del lenguaje natural. Si bien las intenciones permiten comprender la motivación detrás de la entrada de un usuario en particular, las entidades son piezas estructuradas de información que se pueden extraer de un enunciado. Además del nombre de la entidad, podemos anotar una entidad con valores y sinónimos. TeacherName puede ser un ejemplo de Entity que es un EntityType personalizado. EntityID podría ser 3en-22t.
- EntityValue representa el valor principal asociado con la entrada de la entidad. La entidad puede tener uno o más valores. Por ejemplo, si la entidad es TeacherName, el EntityValue podría ser: Rade Matić, Miloš Kabiljo, etc.
- EntitySynonym representa un sinónimo de EntityValue. Puede utilizar sinónimos cuando hay varias formas en que los usuarios se refieren a lo mismo. EntityValue puede tener uno o más EntitySynonym. Por ejemplo, si EntityValue es Rade Matić, EntitySynonym podría ser: RM, Radetom Matićem, Radu Matiću.
- La pronunciación es una frase de ejemplo (de formación) de lo que los usuarios pueden escribir o decir. Para cada intento, podemos tener muchas expresiones. Cuando un mensaje de usuario se parece a (corresponde a) una de estas expresiones, NLU coincide con la intención. Ejemplos de enunciados podrían ser:

- ¿Cuándo tiene una consulta el profesor Rade Matić? (serbio. Kada profesor Rade Matić ima konsultacije)?
- ¿Cuándo puedo consultar con el profesor Rade Matić (serb. Kad mogu da se konsultujem sa profesorom Radetom Matićem)?
- ¿Cuándo puedo visitar al profesor RM (serbio Kad mogu da dođem kod profesora RM)?

• UtteranceEntityValue representa el valor de la entidad en una o más expresiones. El enunciado puede tener cero o más valores de entidad. Las posiciones inicial y final de los atributos son importantes porque así es como el modelo sabe qué caracteres extraer y usar para entrenar el modelo. PartText representa la lista ordenada de partes del enunciado. PartText se concatena para formar el enunciado. Con RoleName podemos definir entidades con roles específicos en un enunciado. GroupName permite que las entidades se agrupen con una etiqueta de grupo específica. La etiqueta de grupo se puede utilizar para definir diferentes pedidos. Por ejemplo, en el enunciado: "¿Cuándo el profesor Rade Matić tiene una consulta?", El valor de entidad Rade Matić es PartText que comienza en la posición 20 y termina en la posición 30.

• La cultura brinda la posibilidad de soporte en varios idiomas. Los enunciados relacionados con una intención se pueden definir para cada idioma de forma individual, así como las entidades que se utilizan en ellos. Cada cultura puede tener su propio conjunto de expresiones y entidades.

El metamodelo de Dialogflow se muestra en la Figura 4. NLU Concept se especializa aún más en conceptos que son más concretos:

• Intención.
• Amable.
• EntityType.
• Entidad.
• Sinónimo.
• TrainingPhrase.
• Parte.

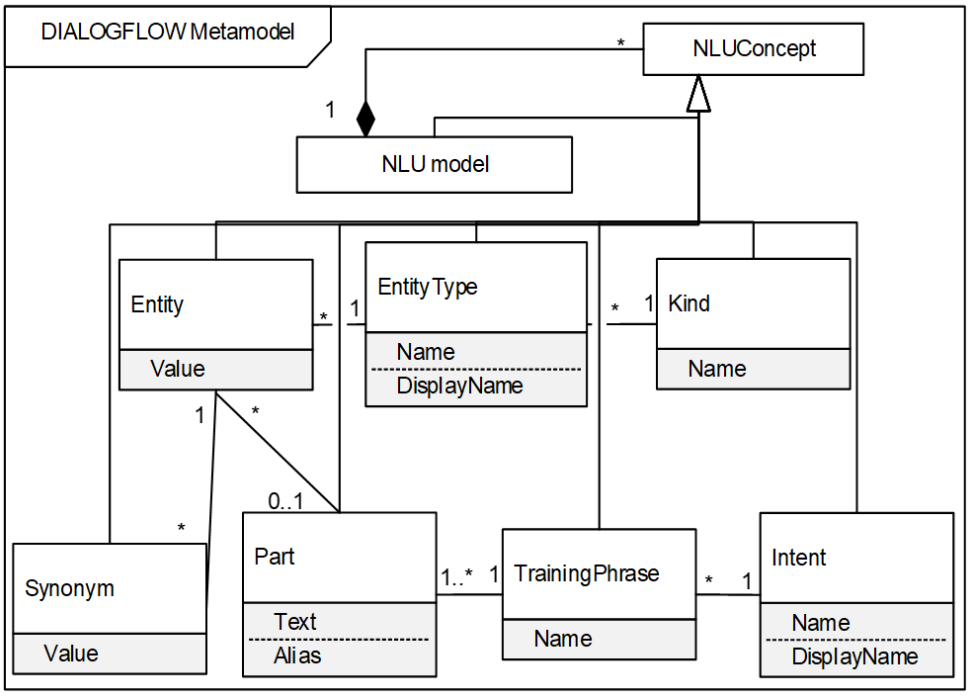

Figura 4. Metamodelo de Dialogflow.

El mapeo entre los conceptos de DORIUS NLU y Dialogflow NLU está determinado por las siguientes reglas:

• Regla de Dialogflow2DORIUS: cada modelo de Dialogflow se asigna a un modelo de DORIUS.
• Regla IN-I.ID: cada Intent.Name se asigna a Intent.IntentID.

• Regla IDN-IN: cada Intent.DisplayName se asigna a Intent.Name.
• Regla KN-ET.N: Cada Kind.Name se asigna a EntityType.Name.
• Regla ET.NE.ID: cada EntityType.Name se asigna a Entity.EntityID.
• ET.DN-EN: Cada EntityType.DisplayName se asigna a Entity.Name.
• Regla EV-EV.V: cada Entity.Value se asigna a EntityValue.Value.
• Regla SV-ES.V: cada Synonym.Value se asigna a EntitySynonym.Value.
• Regla TP.NU.N: cada TrainingPhrase.Name se asigna a Utterance.Name.
• Regla PT-UEV.PT: cada Part.Text se asigna a UtteranceEntityValue.PartText.
• Regla PA-UEV.RN: Cada Part.Alias se asigna a UtteranceEntityValue.RoleName.

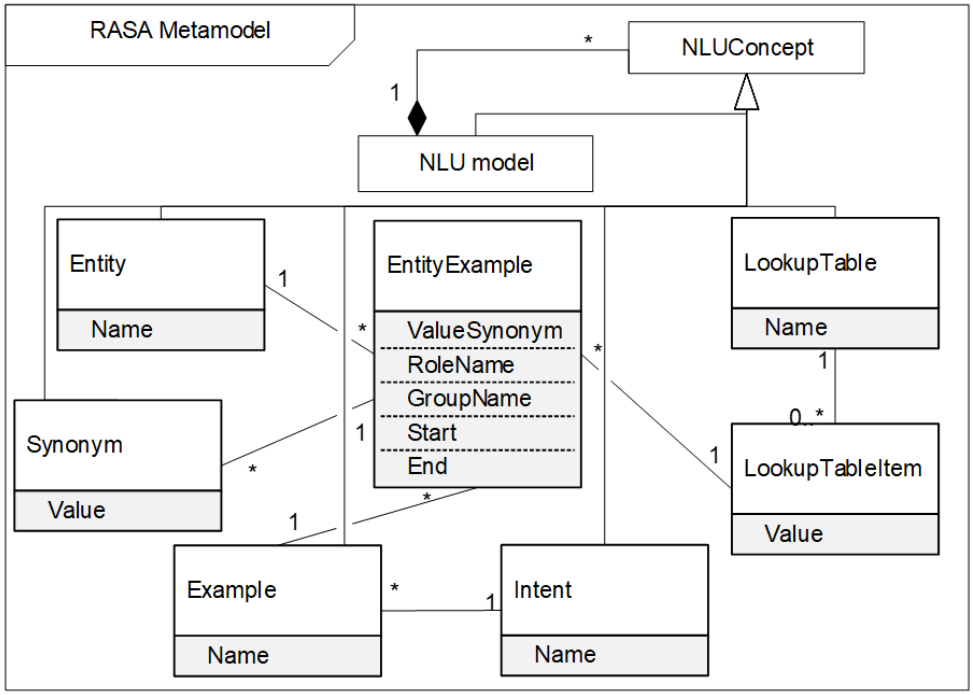

Figura 5. Metamodelo RASA.

El metamodelo de mapeo, que se muestra en la figura 6, define las correspondencias permitidas entre los conceptos de Dialogflow NLU y los conceptos de DORIUS NLU según las reglas. Un mapeo entre dos modelos concretos de Dialogflow y DORIUS está representado por la clase Dialogflow2DORIUS, que abarca todas las correspondencias entre elementos concretos de los modelos Dialogflow y DORIUS realizados utilizando las reglas de mapeo. La clase NLUElement representa una instancia de tales correspondencias. Para cada regla de mapeo, hay subclases apropiadas de NLUElement, que reciben el nombre de la regla. Como se desprende del metamodelo, las asignaciones entre los conceptos del modelo Dialogflow y el modelo DORIUS son únicas y no ambiguas.

El extracto del metamodelo RASA con el concepto RASA NLU más abstracto se muestra en la Figura 5. Está especializado utilizando conceptos de NLU más concretos:

• Intención.
• Entidad.
• EntitySynonym.
• Ejemplo.
• EntityExample.
• Tabla de búsqueda.
• LookupTableItem.

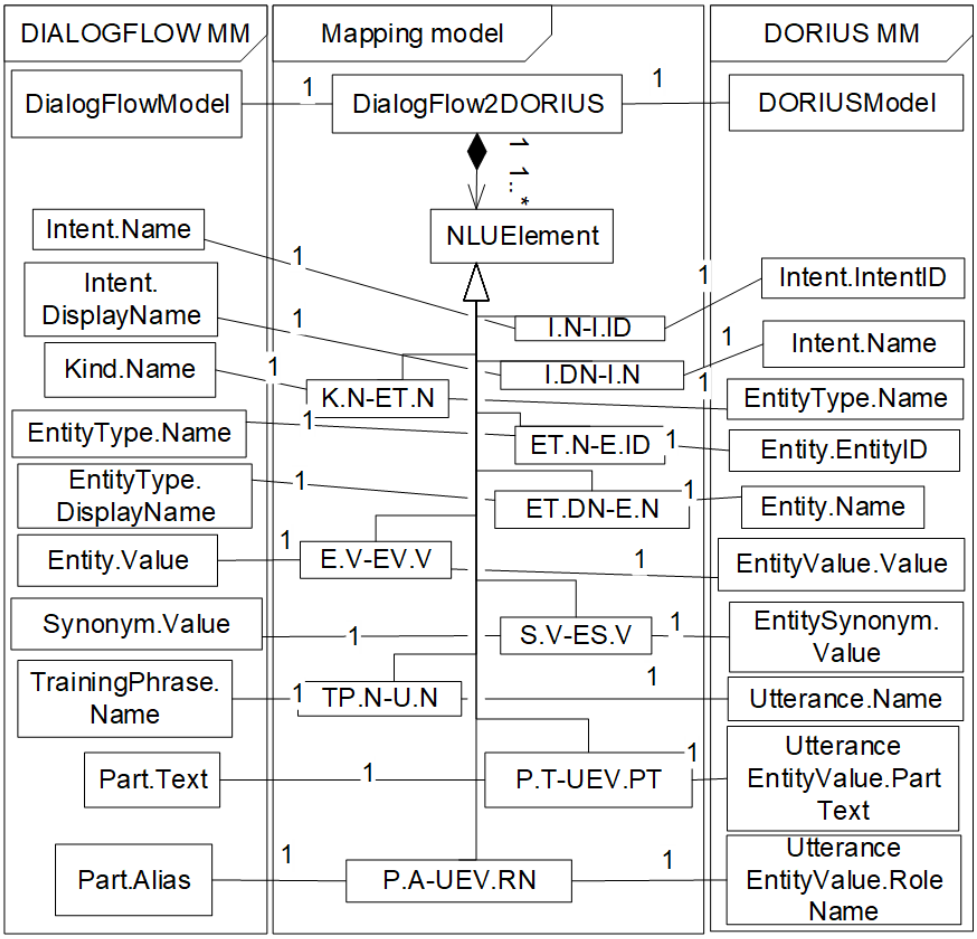

Figura 6. Dialogflow: metamodelo de asignación de DORIUS.

El mapeo entre los conceptos de RASA NLU y DORIUS NLU está determinado por las siguientes reglas:

- Regla RASA2DORIUS: Cada modelo RASA se asigna a un modelo DORIUS.
- Regla IN-I.ID: cada Intent.Name se asigna a Intent.IntentID.
- Regla IN-IN: cada Intent.Name se asigna a Intent.Name.
- Regla EN-EN: Cada Entity.Name se asigna a Entity.EntityID.
- Regla LT.NE.N: cada LookupTable.Name se asigna a Entity.EntityID.
- Regla LTI.V-EV.V: cada LookupTableItem.Value se asigna a EntityValue.Value.
- Regla SV-ES.V: cada Synonym.Value se asigna a EntitySynonym.Value.
- Regla EN-UN: Cada Example.Name se asigna a Utterance.Name.
- Regla EEVS-EV.V: cada EntityExample.ValueSynonym se asigna a EntityValue.Value.
- EE. Inicio-UEV.Start regla: cada EntityExample.Start se asigna a UtteranceEntityValue.Start.
- EEEnd-UEV.End regla: cada EntityExample.End se asigna a UtteranceEntityValue.End.
- Regla EE. RN-UEV.RN: cada EntityExample.RoleName se asigna a UtteranceEntityValue. Nombre de rol.
- Regla EE. GN-UEV.GN: cada EntityExample.GroupName se asigna a UtteranceEntity-Value.GroupName.

El metamodelo de mapeo, que se muestra en la figura 7, define las correspondencias permitidas entre los conceptos RASA NLU y los conceptos DORIUS NLU basados en las reglas. Un mapeo entre dos modelos concretos RASA y DORIUS está representado por la clase RASA2DORIUS, que abarca todas las correspondencias entre elementos concretos de los modelos RASA y DORIUS realizadas utilizando las reglas de mapeo. La clase NLUElement representa una instancia de tales correspondencias. Para cada regla de mapeo, hay subclases apropiadas de NLUElement, que reciben el nombre de la regla. Lo mismo que en el mapeo anterior

metamodelo, las asignaciones entre los conceptos del modelo RASA y el modelo DORIUS son únicas y no ambiguas.

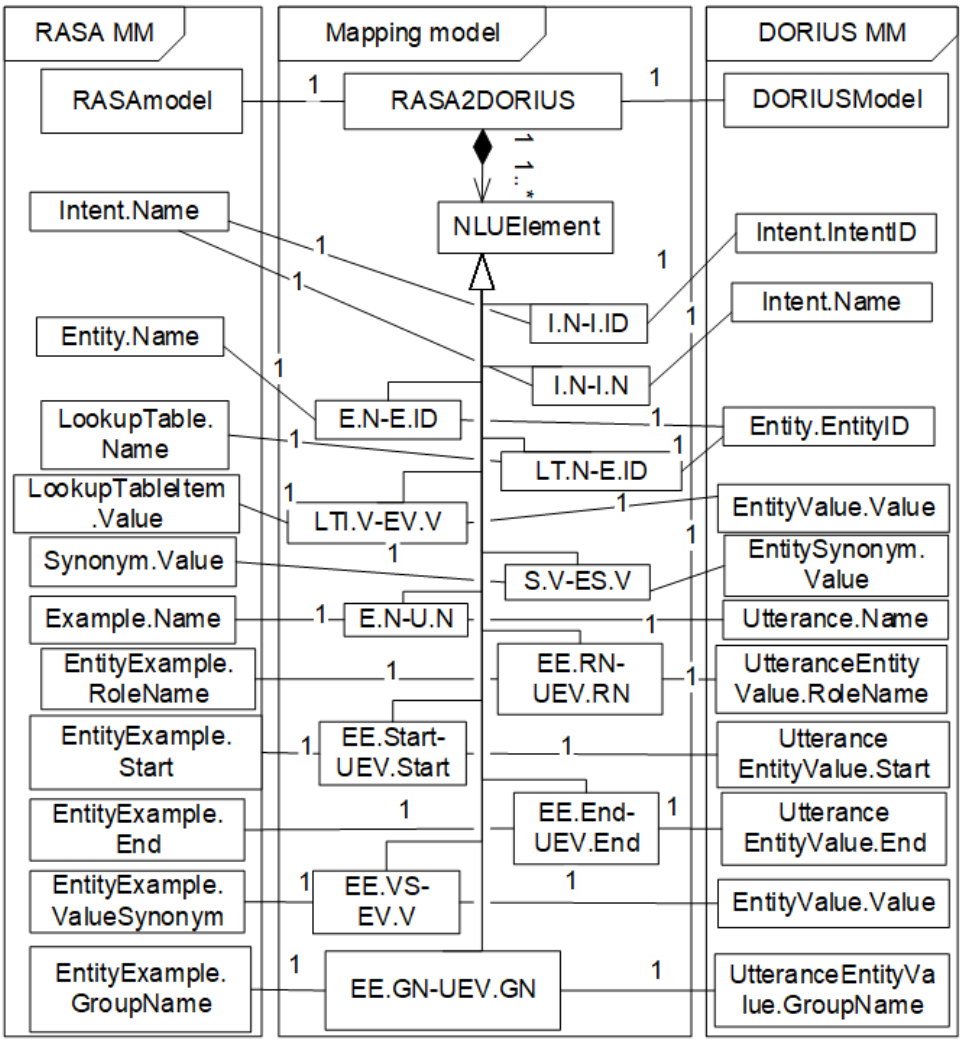

Figura 7. RASA — DORIUS Mapping metamodel.

6. Ejemplo de modelo de mapeo

Una parte del modelo de mapeo correspondiente entre el modelo de Dialogflow y el modelo DORIUS se muestra en el diagrama de objetos de la Figura 8. El paquete del modelo de Dialogflow contiene instancias del metamodelo de Dialogflow de la Figura4, mientras que el paquete modelo DORIUS contiene instancias del metamodelo DORIUS de la Figura 3. El modelo de mapeo de paquetes contiene instancias de las reglas de mapeo correspondientes mediante las cuales los conceptos de Dialogflow se mapean a partir del concepto de DORIUS. Por ejemplo, la intención i1 (Consulta del profesor) se asigna a partir de la intención correspondiente del mismo nombre utilizando la regla I.DN-IN. El tipo k1 se asigna desde et1: EntityType del mismo nombre utilizando la regla KN-ET.N. Por otro lado, el atributo Name (3en-22t) de et1: EntityType, se mapea a partir del atributo EntityID (3en-22t) de e1: Entity por la regla ET.NE.ID. Además, el atributo DisplayName (TeacherName) de et1: EntityType, se asigna a partir del atributo Name (TeacherName) de e1: Entity por la regla ET.DN-EN Por lo tanto, las entidades e1 (Rade Matić) y e2 (Miloš Kabiljo) se asignan a la entidad correspondiente valores ev1 y ev2 del mismo valor utilizando la regla EV-E.VV. Además, los sinónimos s1 (RM) y s2 (Radu Matiću) se asignan a los sinónimos de entidad correspondientes es1 y es2 del mismo valor utilizando la regla SV-E.SV. Las frases de entrenamiento tp1 y tp2 se mapean a partir de los enunciados correspondientes u1 y u2 del mismo nombre utilizando la regla TP.NU.N. Texto de atributo de p1: La parte se mapea desde at-

tributo PartText de uev1: UtteranceEntityValue por la regla PT-UEV.RN. UtteranceEntityValue representa el valor de la entidad en una o más expresiones. Por ejemplo, en el enunciado: "¿Cuándo tiene una consulta el profesor Rade Matić?", El valor de entidad Rade Matić es PartText. Aquí se da un ejemplo de un modelo de mapeo con un solo metamodelo NLU. Se puede mostrar un ejemplo similar de mapeo para el metamodelo RASA, pero debido a limitaciones de trabajo, los detalles del mapeo no son parte de este trabajo.

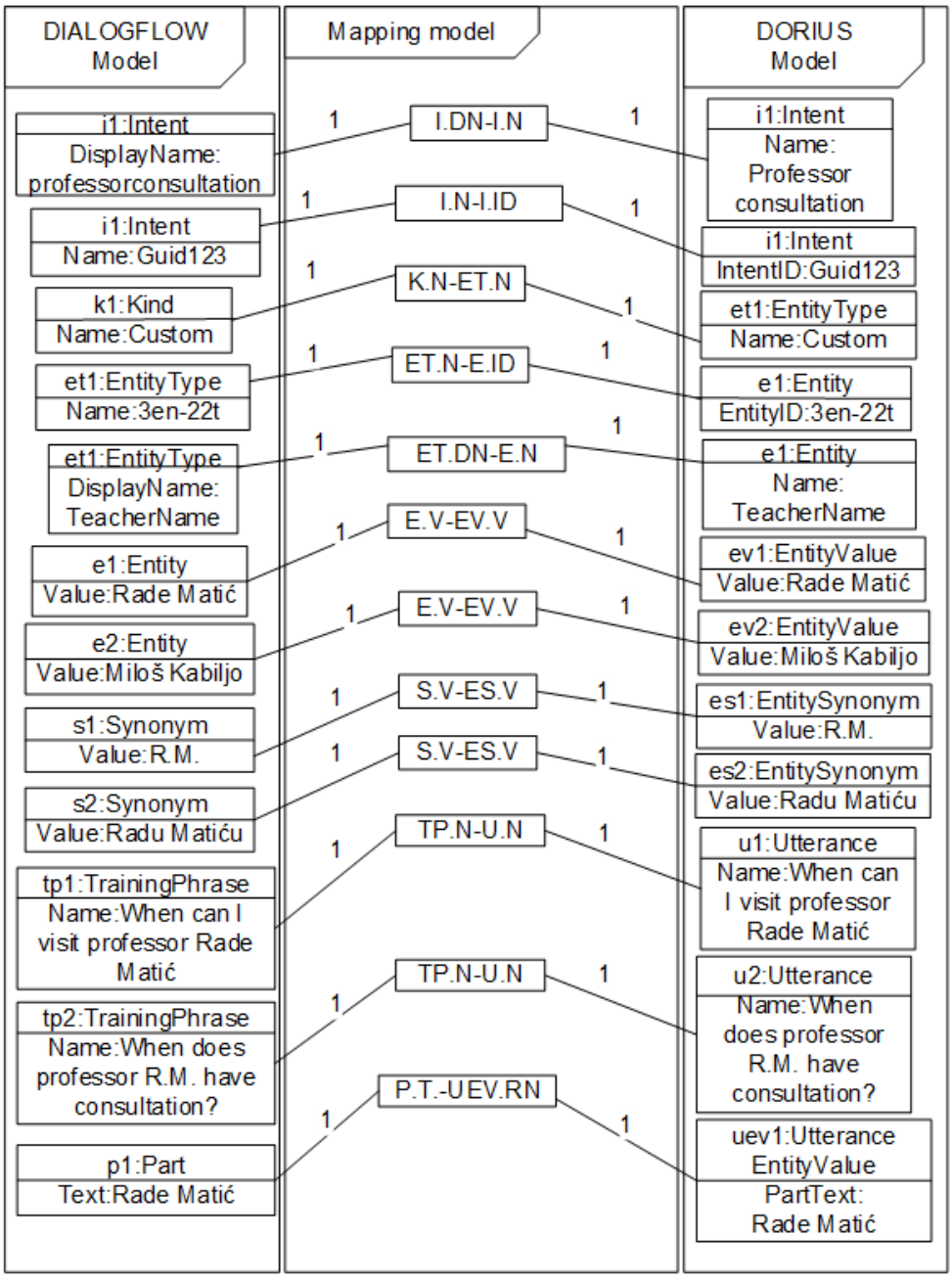

Figura 8. Una parte del modelo de mapeo entre Dialogflow y el modelo DORIUS.

7. Dos estudios de caso y sus implementaciones

Algunas experiencias en la implementación de la arquitectura de chatbot descrita existen en dos estudios de caso. El primero es un chatbot ADA de la Academia de Estudios Aplicados de Arte y Negocios de Belgrado y el segundo es “COVID-19 Info Serbia”. En las condiciones de la pandemia de COVID-19 y debido al cierre del sistema educativo, la Academia de Artes y Negocios de Belgrado de Estudios Aplicados [41] se requiere para cambiar a la entrega en línea de interacción con sus estudiantes y contenido educativo [42]. Mejorar, modernizar,

y digitalizar los servicios educativos, la Academia desarrolló un ADA (Asistente digital académico) [43] chatbot que usa la plataforma Weaver para brindar a sus estudiantes un mejor servicio e información necesaria durante sus estudios, como se muestra en la Figura 9.

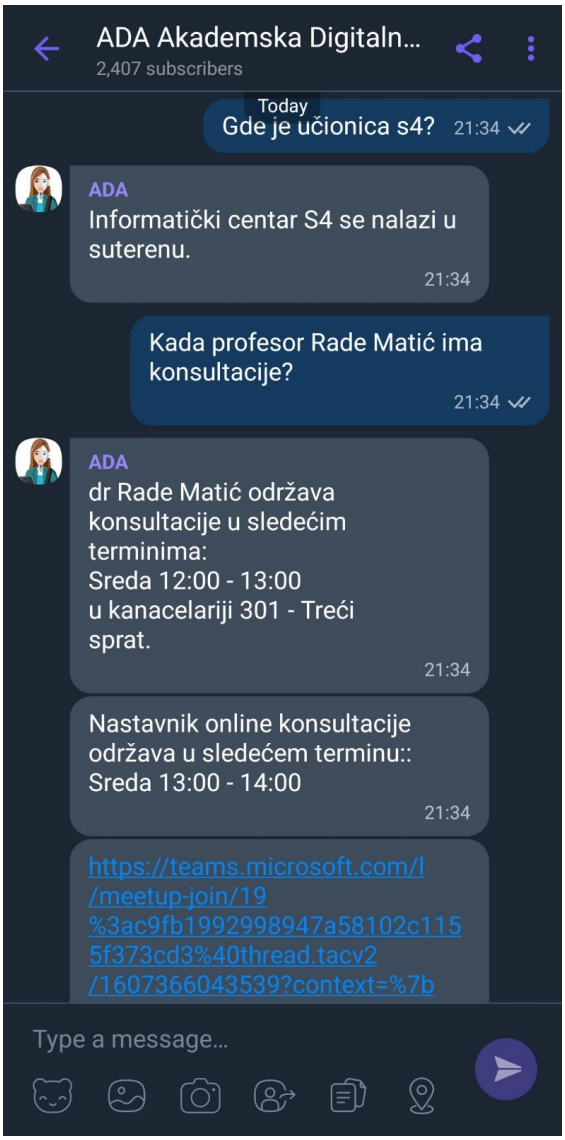

Figura 9. Chatbot ADA (Asistente digital académico), implementado en la Academia de Artes y Negocios de Belgrado de Estudios Aplicados.

Actualmente, ADA utiliza dos servicios externos de NLU: RASA y Dialogflow. Gracias a la solución propuesta aquí, fue fácil cambiar de un servicio NLU a otro. La prueba se realizó en tres fases. La primera fase implicó pruebas internas de las intenciones debido a las especificaciones del idioma serbio. Confirmamos las deficiencias de Dialogflow en el documento [32] y cambió a RASA. La segunda fase de prueba se llevó a cabo en un ambiente controlado con un grupo de 50 estudiantes usando RASA. Les dijimos a estos estudiantes lo que sabe la ADA, pero no les dijimos cómo preguntar. El objetivo era lograr un 75% de precisión. Los estudiantes ingresaron los enunciados apropiados, pero obtuvimos una precisión promedio de reconocimiento de intenciones de alrededor del 55%. Nuestro modelo de aprendizaje no fue tan bueno como esperábamos, pero las declaraciones que la ADA no reconoció nos ayudaron a impulsar la capacitación. El modelo se enriqueció mucho con nuevas expresiones. La tercera y última fase de las pruebas involucró a todos los estudiantes de una especialización en TI en la Academia. Nos proporcionaron muchas expresiones que agregamos constantemente a la capacitación hasta que logramos que la ADA respondiera el 85% de las preguntas. Se excluyeron los problemas que estaban fuera del modelo aprendido

del entrenamiento y la puntuación de confianza porque algunos enunciados estaban fuera del alcance de esta parte de la prueba. Para aprovechar al máximo nuestros datos de entrenamiento, entrenamos y evaluamos nuestro modelo en diferentes canales y diferentes cantidades de datos de entrenamiento. Para admitir serbio, no usamos la biblioteca spaCy. Se hace la misma recomendación para otros idiomas que no tienen soporte en spaCy. Para inglés y otros idiomas más populares, es muy útil incluir spaCy. Independientemente del idioma que usemos, es muy importante aprovechar al máximo los datos de entrenamiento. Tuvimos que entrenar y evaluar nuestro modelo en diferentes canalizaciones y diferentes cantidades de datos de entrenamiento.

En las figuras se muestra una instantánea de la definición de un escenario y la validación de NLU 10 y 11, respectivamente. Figura10 muestra cómo el administrador puede elegir qué servicio NLU utilizar. Figura11 muestra la validación NLU de enunciados para las dos intenciones solicitadas por los estudiantes, así como su núcleo de confianza en el servicio RASA NLU. Usando la validación NLU es posible enseñar directamente el servicio NLU para evitar la dependencia de la interfaz NLU del proveedor.

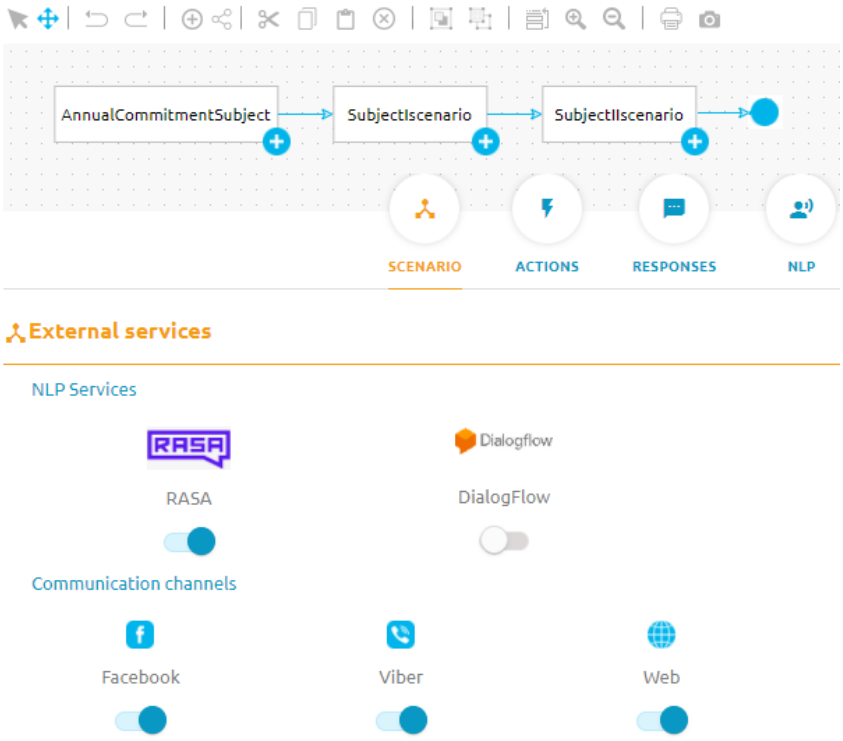

Figura 10. Diseñador de bots.

Figura 11. Validación NLU.

ADA ayuda a los futuros estudiantes a aprender más sobre la Academia. ADA está capacitada para brindar a los estudiantes un servicio educativo de la más alta calidad con una amplia gama de información sobre los maestros, las aulas, el plan de estudios, la literatura, las horas de trabajo, la biblioteca, las materias, las notificaciones, el contenido educativo, el calendario de conferencias y la lista de precios de manera fácil. de usar, de manera conversacional. Los estudiantes solicitan a la ADA diversa información, como: el costo total de la matrícula, qué documentos se requieren para la inscripción en el nuevo año escolar, el calendario de conferencias o la fecha de admisión de un profesor en particular, como se muestra en la Figura9. ADA también puede realizar algunos procesos como registro o cancelación de exámenes, cambio de profesor en el examen, restablecimiento de contraseña o compra de un libro. En los meses analizados, el interés de los estudiantes fue acorde con el calendario escolar, es decir, en enero, mes de exámenes, los estudiantes se interesaron más cuando el docente imparte, examina o realiza ejercicios de una determinada materia, como se muestra en la Tabla1. En febrero, mes de matrícula en el nuevo curso escolar, en la parte superior de la lista de consultas estaba la información relacionada con el costo de la matrícula y el tipo de documentos requeridos para la matrícula, como se muestra en la Tabla2. Cuando ambas tablas1 y 2 se consideran, también es posible observar el aumento significativo en la cantidad total de consultas.

Tabla 1. Estadísticas de consultas de estudiantes en enero de 2021.

Las preguntas más frecuentes	sin. de conversaciones
Qué profesor enseña y examina una materia en particular	178
El monto de cualquier tasa de matrícula ¿Cuál	176
es el horario de conferencias? Cuándo	140
comienzan las clases en un semestre determinado	137
¿Cuándo comienza la inscripción del estudiante?	134

Tabla 2. Estadísticas de consultas de estudiantes en febrero de 2021.

Las preguntas más frecuentes	sin. de conversaciones
Documentación requerida para la inscripción para un año escolar específico	513
El monto de cualquier tasa de matrícula	472
En qué gabinete y a qué hora el profesor realiza las consultas Ubicación,	470
horario de trabajo, libros ofrecidos en la tienda de guiones de la Academia	459
Información de clasificación presupuestaria	298

Gracias a esto, logramos que casi el 60% de los estudiantes usaran ADA como su principal canal de comunicación con la Academia. También proporciona enlaces a recursos adicionales para aquellos que deseen aprender más sobre la Academia. El proyecto de chatbot de ADA es el primer proyecto de este tipo en los Balcanes en el campo de la educación que proporciona ahorro de tiempo, una comunicación mejor y más fácil, y una implementación más rápida y eficiente de los servicios educativos. La tecnología Chatbot tiene el potencial de influir significativamente en la forma en que los estudiantes experimentan las instituciones educativas y la forma en que interactúan con ellas. El objetivo de la plataforma, además de estar disponible para muchos estudiantes para toda la información necesaria relacionada con la Academia, es obtener información oficial sin esperas, así como relevar al personal del centro de contacto y a los profesores, permitiéndoles prestar atención a tareas más creativas. El Chatbot está disponible gratuitamente y se lanza en serbio a través de Viber y FB Messenger.

Debido a que el idioma serbio es muy difícil, se necesitaba un conjunto de entrenamiento muy grande con varias expresiones para que el modelo fuera lo más eficiente posible. Además de los enunciados, cuidamos las entidades, sus valores, así como sus sinónimos debido a la complejidad del idioma serbio, que puede ser un problema durante el reconocimiento de intenciones. ADA, a diferencia de COVID-19, tiene muchas entidades que tienen el mismo valor pero significados diferentes. En ese caso, usamos el concepto de rol de entidad para que la misma entidad en una oración pueda tener dos significados. Un ejemplo de esta entidad es "gestión", que puede ser el nombre de la materia en un programa de estudios y el nombre de otro programa de estudios. Mesa3 presenta el

número de intenciones, expresiones, entidades para hacer que ADA y COVID-19 sean funcionales en serbio. Estos números están creciendo porque trabajamos constantemente para aprender y mejorar el modelo.

Tabla 3. La cantidad de intenciones, expresiones y entidades para hacer que ADA y COVID-19 sean funcionales en serbio.

	ADA	COVID-19
Intención	202	107
Declaración	7549	3770
Entidad	20	10
Valores de entidad	525	120
Sinónimos de entidad	3524	150

Para facilitar el proceso de difusión de información eficiente y oportuna, una propuesta de solución está participando en el desarrollo de “COVID-19 Info Serbia” [44]. Esto incluye una amplia gama de información sobre el coronavirus y sus síntomas. También proporciona enlaces a recursos adicionales para aquellos que deseen aprender más sobre la enfermedad. El chatbot está disponible gratuitamente y se lanza en serbio e inglés a través de Viber [45]. “COVID-19 Info Serbia” es un chatbot en versión beta desarrollado en la plataforma Weaver. Este es un modelo basado en la recuperación conectado con NLU, que se clasifica en dominios cerrados y proporciona información más fácil y rápida sobre la epidemia del virus COVID-19 en Serbia [46]. El Asistente Virtual responde todas las posibles preguntas de los ciudadanos sobre COVID-19 sin esperar, y está disponible las 24 horas del día, los 7 días de la semana en el sitio web oficial del gobierno relacionado con la situación del virus COVID-19. El chatbot “COVID-19 Info Serbia” tiene el potencial de crear experiencias de aprendizaje individuales sobre COVID-19. El chatbot responde con respuestas oficiales a las preguntas más comunes de los ciudadanos sobre COVID-19, incluidos detalles sobre el estado de emergencia, números de emergencia locales, síntomas, las últimas actualizaciones del Ministerio de Salud y otra información útil. También ayuda a reducir la presión sobre las líneas directas de atención médica y a mantener abiertas las líneas telefónicas para las personas que realmente necesitan hablar con un médico. “COVID-19 Info Serbia” es obviamente una simplificación de lo que podría hacer un chatbot.

COVID-19 había acelerado las pruebas debido a plazos cortos y una situación de pandemia. Por lo tanto, este chatbot es diferente porque tiene escenarios simples y también hay escenarios basados en botones. La producción piloto se inició muy rápidamente, donde participó una gran cantidad de ciudadanos, quienes fortalecieron la capacitación con diversos temas y ampliaron el dominio del conocimiento del modelo. También implementamos un escenario que brinda la oportunidad a los ciudadanos de registrarse para la vacunación. Cerca de 30.000 ciudadanos solicitaron la vacunación en la primera ronda a través del chatbot COVID-19. Este chatbot tiene actualmente alrededor de 300.000 suscriptores en el canal de Viber, y la tasa de éxito de las respuestas correctas fue del 80%.

La plataforma Weaver se basa en motores NLU entrenados, pero cada chatbot que se ha creado tiene un conjunto diferente de diseños de conversación y resuelve diferentes problemas de la industria. Para lograr el mejor resultado, necesitamos una cantidad decente de datos de buena calidad. Con la condición de proporcionar el mejor flujo de conversación, la plataforma podrá funcionar de la mejor manera.

8. Conclusiones

Impulsados por la promesa de asistentes digitales inteligentes que siempre estarán disponibles para una resolución rápida y consistente de las solicitudes de los clientes, los chatbots son cada vez más útiles. Uno de los principales objetivos era crear una arquitectura de chatbot ágil y extensible para manejar un entorno altamente variable de plataformas modernas y tecnologías emergentes. Las ventajas que ofrece la arquitectura de chatbot propuesta son las siguientes:

- Es extensible y admite nuevos canales de comunicación y comprensión del lenguaje natural para las interacciones del usuario.
- Se basa en metamodelos como principal mecanismo extensible. Para ello proporcionamos:

- Metamodelo general de NLU.
- Metamodelos para los dos servicios NLU específicos (Dialogflow y RASA).
- Metamodelos correspondientes, así como reglas para un mapeo entre metamodelos NLU genéricos y específicos.

• Es adaptable y personalizable, refiriéndose a la capacidad de la arquitectura para ser administrada y personalizada por un modelador.
• Es inherentemente escalable mediante microservicios.
• La solución se implementó en dos estudios de caso sobre el idioma serbio.
• Proporciona una funcionalidad de tiempo de ejecución madura dentro del dominio de los chatbots.
• Proporciona un flujo de trabajo que representa procesos de negocio (escenarios) detallados y dinámicos relacionados con la secuencia de actividades paso a paso que se utilizan para completar las intenciones.

En este documento, mostramos arquitecturas de software que se pueden expandir, adaptar, administrar y personalizar para cumplir con los cambios futuros y las tecnologías de chatbot. La independencia de la plataforma está estrechamente relacionada con la preocupación por las API abiertas y el soporte para objetos distribuidos. Hemos presentado la arquitectura avanzada del marco de chatbot construido sobre microservicios, es decir, arquitectura de servicio API. También presentamos una solución para crear un chatbot que sea independiente de un servicio externo de NLU. Las lecciones aprendidas de ambos casos demostraron que la API web era una buena opción, entre otras cosas, porque:

• El servicio se basa en el servicio RESTful que habilita el protocolo HTTP y los objetos JSON.
• Puede alojarse dentro de la aplicación o IIS (Internet Information Services).
• Puede ser utilizado por cualquier cliente que entienda JSON o lenguaje de marcado extensible (XML).
• Tiene una arquitectura sencilla.

Al desarrollar y diseñar la arquitectura del chatbot, utilizamos varias técnicas para la escalabilidad. Gracias a los microservicios, nuestra arquitectura puede reemplazar un componente por otros más potentes y rápidos a medida que crecen los requisitos y se desarrolla la tecnología. La plataforma de chatbot no se preocupa por el front-end de las plataformas de comunicación. El mantenimiento es más fácil porque el servicio de comunicaciones mantiene el front-end. Para la primera fase, solo usamos la caché local para acelerar la recuperación de solicitudes. Usamos tablas de índice para resolver una búsqueda rápida. Regulamos las solicitudes concurrentes y manejamos muchas de ellas. Realizamos consultas de forma asincrónica y las ponemos en cola. El procesamiento asincrónico elimina algunos de los cuellos de botella que afectan al rendimiento. Aumentamos la escalabilidad eligiendo Angular como uno de los marcos modernos para aplicaciones de una sola página mejorando el rendimiento general. La interfaz de usuario para el aprendizaje de NLU se define en un solo lugar, independientemente de los diferentes servicios externos de NLU. Las definiciones de escenarios también se definen en un solo lugar, independientemente de las diferentes plataformas de comunicación. Cambiar de una a más NLU o introducir una nueva NLU es muy fácil de lograr con un par de clics debido a los metamodelos y al bajo acoplamiento de las partes básicas de la arquitectura. Esto reduce la curva de aprendizaje del usuario y ayuda a comprender mejor el mensaje del usuario. La solución se implementó en dos estudios de caso: “ADA” y “COVID-19 Info Serbia”. ADA Chatbot es una plataforma que utiliza NLU para encontrar la respuesta correcta a todas las preguntas de los estudiantes y resolver sus problemas. De ese modo, aumenta la satisfacción de los estudiantes y logran una mejor interacción con la Academia utilizando más de una plataforma de comunicación y servicios de NLU. COVID-19 ha acelerado la necesidad de soluciones de chatbot. Los chatbots pueden no ser la solución a todos los problemas de los estudiantes, pero es una herramienta poderosa que puede aumentar la eficiencia de la Academia y apoyar la realización del proceso educativo. Después de la pandemia, el uso de chatbots para aplicaciones de información y educación seguirá creciendo. Lo que más dice sobre el éxito de esta implementación es que la Academia dedicó toda la campaña de imagen de marca a ADA. Para informar mejor a los ciudadanos de Serbia sobre COVID-19, el Gobierno de Serbia ha lanzado un chatbot.47]. La plataforma propuesta cubre escenarios generales y temas relacionados con el virus, y se enfoca en información sobre síntomas, medidas de prevención, teléfonos importantes, datos actuales y

decisiones del Gobierno de la República de Serbia. Sin embargo, este procedimiento de formación requiere más paciencia, tiempo y conocimiento del dominio. Siempre es necesaria la intervención humana, incluso cuando se implementan chatbots. Este trabajo de investigación puede continuar en la dirección de la automatización total y la extensión de nuestro prototipo con nuevas posibilidades. Por ejemplo, se podría agregar una extracción automática de la opinión del usuario de la comunicación. Necesitamos crear una caché distribuida para proporcionar la distribución de la pieza de datos en todos los nodos. Luego, el servicio de soporte debe habilitarse para mensajes de voz y mediciones de desempeño adicionales para identificar cuellos de botella, utilizando técnicas de evaluación como [17,18]. Los autores afirman que la arquitectura de chatbot propuesta presentada en este documento es un excelente punto de partida para estas direcciones de investigación.

Contribuciones de autor: Conceptualización, RM y MK; metodología, MZ y MC; software, RM y MK; validación, RM, MZ y MC; análisis formal, MK y RM; investigación, RM y MK; recursos, RM, MK, MZ y MC; redacción: preparación del borrador original, RM y MK; redacción — revisión y edición, MZ y MC; visualización, RM; supervisión, RM y MK; administración del proyecto, RM Todos los autores han leído y están de acuerdo con la versión publicada del manuscrito.

Fondos: Esta investigación no recibió financiación externa.

Conflictos de interés: Los autores declaran no tener ningún conflicto de intereses.

Referencias

1. Bansal, H .; Khan, R. Un artículo de revisión sobre la interacción entre humanos y computadoras.*En t. J. Adv. Res. Computación. Sci. Softw. Ing.***2018**, *8*, 53. [CrossRef]
2. Dale, R. El regreso de los chatbots. *Nat. Lang. Ing.***2016**, *22*, 811–817. [CrossRef]
3. Smutny, P .; Schreiberova, P. Chatbots para aprender: una revisión de los chatbots educativos para Facebook Messenger.*Computación. Educ.***2020**, *151*, 103862. [CrossRef]
4. Informe Chatbot 2019: Análisis y Tendencias Globales | por BRAIN [BRN.AI] CÓDIGO DE EQUIDAD | Revista Chatbots. Disponible en linea: https://chatbotsmagazine.com/chatbot-report-2019-global-trends-and-analysis-a487afec05b (consultado el 5 de junio de 2021).
5. Tamaño del mercado de chatbot, participación | Tendencias y análisis de la industria para 2027. Disponible en línea: https://www.alliedmarketresearch.com/ chatbot-market (consultado el 5 de junio de 2021).
6. Fayad, ME; Hamza, HS; Sánchez, HA Hacia arquitecturas de software escalables y adaptables. En Actas de la Conferencia Internacional IRI-2005 IEEE sobre Reutilización e Integración de la Información, Las Vegas, NV, EE. UU., 15-17 de agosto de 2005; págs. 102-107.
7. Weaver. Disponible en linea:https://weaverbot.ai/ (consultado el 5 de junio de 2021).
8. Weizenbaum, J. ELIZA — Un programa de computadora para el estudio de la comunicación del lenguaje natural entre el hombre y la máquina. *Comun. ACM***1966**, *9*, 36–45. [CrossRef]
9. Weizenbaum, J. Eliza — Un programa de computadora para el estudio de la comunicación del lenguaje natural entre el hombre y la máquina. *Comun. ACM***1983**, *26*, 23-28. [CrossRef]
10. Marietto, MDGB; de Aguiar, RV; Barbosa, GDO; Botelho, WT; Pimentel, E .; França, RDS; da Silva, VL Lenguaje de marcado de inteligencia artificial: un breve tutorial.*arXiv* **2013**, arXiv: 1307.3091.
11. Wallace, RS La anatomía de ALICE. En*Analizando la prueba de Turing*; Springer: Dordrecht, Holanda, 2009; págs. 181–210.
12. Rodríguez Cardona, D .; Werth, O .; Schönborn, S .; Breitner, MH Un análisis de métodos mixtos de la adopción y difusión de la tecnología Chatbot en el sector de seguros alemán. En Actas de la AMCIS, Cancún, México, 15-17 de agosto de 2019.
13. Adamopoulou, E .; Moussiades, L. Chatbots: Historia, tecnología y aplicaciones.*Mach. Aprender. Apl.***2020**, *2*, 100006.
14. Mu, J .; Sarkar, A. ¿Necesitamos el lenguaje natural? Explorando interfaces de lenguaje restringidas para dominios complejos. En Proceedings of the Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Reino Unido, 4 a 9 de mayo de 2019; págs. 1–6.
15. Daniel, G .; Cabot, J .; Deruelle, L .; Derras, M. Modelado e implementación de chatbot multiplataforma con el marco Jarvis. En Actas de la Conferencia Internacional sobre Ingeniería Avanzada de Sistemas de Información, Roma, Italia, 3 a 7 de junio de 2019; Springer: Basilea, Suiza, 2019; págs. 177-193.
16. Abdul-Kader, SA; Woods, J. Encuesta sobre técnicas de diseño de chatbot en sistemas de conversación por voz.*En t. J. Adv. Computación. Sci. Apl.* **2015**, *6*. [CrossRef]
17. Radziwill, NM; Benton, MC Evaluación de la calidad de chatbots y agentes conversacionales inteligentes.*arXiv* **2017**, arXiv: 1704.04579.
18. Pereira, J .; Díaz, O. Un análisis de calidad de los chatbots más populares de Facebook Messenger. En Actas del 33 ° Simposio Anual de ACM sobre Computación Aplicada, Pau, Francia, 9-13 de abril de 2018; págs. 2144–2150.
19. Nimavat, K .; Champaneria, T. Chatbots: una descripción general. Tipos, arquitectura, herramientas y posibilidades futuras.*En t. J. Sci. Res. Dev.***2017**,*5*, 1019-1024.
20. Jwala, K .; Sirisha, G .; Raju, GP Desarrollando un Chatbot usando Machine Learning.*En t. J. Recent Technol. Ing. (IJRTE)***2019**, *8*, 89–92.
21. Swanson, K .; Yu, L .; Fox, C .; Wohlwend, J .; Lei, T. Construyendo un modelo de producción para Chatbots basados en recuperación.*arXiv* **2019**, arXiv: 1906.03209.

22. Peng, Z .; Ma, X. Una encuesta sobre métodos de construcción y mejora en el diseño de chatbots de servicios.*CCF Trans. Computación generalizada. Interactuar.***2019**, *1*204–223. [CrossRef]
23. Hussain, S .; Sianaki, OA; Ababneh, N. Una encuesta sobre técnicas de diseño y clasificación de agentes conversacionales / chatbots. En Actas de los talleres de la Conferencia internacional sobre redes y aplicaciones de información avanzada, Matsue, Japón, 27 a 29 de marzo de 2019; Springer: Basilea, Suiza, 2019; págs. 946–956.
24. Khan, R. Arquitectura estandarizada para agentes conversacionales, también conocidos como chatbots. *En t. J. Comput. Trends Technol.***2017**, *50*, 114-121. [CrossRef]
25. Motger, Q .; Franch, X .; Marco, J. Agentes conversacionales en ingeniería de software: encuesta, taxonomía y desafíos.*arXiv* **2021**, arXiv: 2106.10901.
26. Daniel, G .; Cabot, J .; Deruelle, L .; Derras, M. Xatkit: Un marco de desarrollo de chatbot de código bajo multimodal.*Acceso IEEE* **2020**,*8*, 15332-15346. [CrossRef]
27. Pérez-Soler, S .; Guerra, E .; de Lara, J. Desarrollo de chatbot impulsado por modelos. En Actas de la Conferencia Internacional sobre Modelado Conceptual, Viena, Austria, 3 a 6 de noviembre de 2020; Springer: Basilea, Suiza, 2020; págs. 207–222.
28. Hill, J .; Ford, WR; Farreras, IG Conversaciones reales con inteligencia artificial: una comparación entre las conversaciones en línea entre humanos y humanos y las conversaciones entre humanos y chatbot.*Computación. Tararear. Behav.***2015**, *49*, 245–250. [CrossRef]
29. Divya, S .; Indumathi, V .; Ishwarya, S .; Priyasankari, M .; Devi, SK Un chatbot médico de autodiagnóstico que utiliza inteligencia artificial.*J. Web Dev. Web Des.***2018**, *3*, 1-7.
30. Petrovic, A .; Zivkovic, M .; Bacanin, N. Singibot-A Chatbot de servicios para estudiantes. En Actas de la Conferencia Científica Internacional Sinteza 2020 sobre Tecnología de la Información e Investigación Relacionada con Datos, Belgrado, Serbia, 17 de octubre de 2020; Universidad Singidunum: Belgrado, Serbia, 2020; págs. 318–323.
31. Ranoliya, BR; Raghuwanshi, N .; Singh, S. Chatbot para preguntas frecuentes relacionadas con la universidad. En Actas de la Conferencia Internacional de 2017 sobre Avances en Computación, Comunicaciones e Informática (ICACCI), Udupi, India, 13-16 de septiembre de 2017; págs. 1525-1530.
32. Abdellatif, A .; Badran, K .; Costa, D .; Shihab, E. Una comparación de plataformas de comprensión del lenguaje natural para chatbots en ingeniería de software.*IEEE Trans. Softw. Ing.***2021**. [CrossRef]
33. Adamopoulou, E .; Moussiades, L. Una descripción general de la tecnología de chatbot. En Actas de la Conferencia Internacional IFIP sobre Aplicaciones e Innovaciones de Inteligencia Artificial, Halkidiki, Grecia, 5 a 7 de junio de 2020; Springer: Basilea, Suiza, 2020; págs. 373–383.
34. Fayad, ME; Hamu, DS; Brugali, D. Características, criterios y desafíos de los marcos empresariales.*Comun. ACM***2000**, *43*, 39–46. [CrossRef]
35. LUIS (Comprensión del lenguaje) —Servicios cognitivos — Microsoft. Disponible en linea:https://www.luis.ai/ (consultado el 5 de julio de 2021).
36. Wit.ai. Disponible en linea:https://wit.ai/ (consultado el 5 de julio de 2021).
37. Dialogflow, plataforma de comprensión del lenguaje natural. Disponible en linea:https://cloud.google.com/dialogflow/docs/ (consultado el 5 de junio de 2021).
38. AI conversacional de código abierto | Rasa. Disponible en linea:https://rasa.com/ (consultado el 5 de junio de 2021).
39. IBM Watson | IBM. Disponible en linea:https://www.ibm.com/watson (consultado el 5 de julio de 2021).
40. Nešković, S .; Matić, R. Modelado de contexto basado en modelos de características expresados como puntos de vista sobre ontologías a través de mapeos. *Computación. Sci. Inf. Syst.***2015**, *12*, 961–977. [CrossRef]
41. BAPUSS — Beogradska Akademija Poslovnih i Umetničkih Strukovnih Studija. Disponible en linea:https://www.bpa.edu.rs/(consultado el 5 de julio de 2021).
42. Kabiljo, M .; Vidas-Bubanja, M .; Matić, R .; Zivković, M. Sistema educativo en la república de serbia bajo condiciones COVID-19: asistente digital Chatbot-académico de la academia de estudios aplicados de artes y negocios de belgrado.*Knowl. En t. J.***2020**, *43*, 25-30.
43. Chatbot de la ADA. Disponible en linea:https://chatbot.bpa.edu.rs/en/index.html (consultado el 5 de julio de 2021).
44. Ministarstvo Zdravlja Republike Srbije — COVID-19. Disponible en linea:https://covid19.rs/homepage-english/ (consultado el 5 de julio de 2021).
45. COVID-19 Info Srbija en Viber. Disponible en linea:https://chats.viber.com/covid19info (consultado el 5 de julio de 2021).
46. COVID-19 Info Serbia | Saga — New Frontier Group. Disponible en linea:https://saga.rs/news/COVID-19-info-serbia/?lang=en (consultado el 5 de julio de 2021).
47. Nguyen, TT; Nguyen, QVH; Nguyen, DT; Hsu, EB; Yang, S .; Eklund, P. Inteligencia artificial en la batalla contra el coronavirus (COVID-19): una encuesta y direcciones de investigación futuras.*arXiv* **2020**, arXiv: 2008.07343.