JOURNAL OF MODERN SCIENCE
SPECIAL ISSUE
3/57/2024
www.jomswsge.com
DOI: doi.org/10.13166/jms/191223

TOMASZ SMUTEK
WSEI University in Lublin, Poland
ORCID iD: orcid.org/0009-0003-4756-7354

EWA GOLEC
WSEI University in Lublin, Poland
ORCID iD: orcid.org/0009-0009-0226-9955

PAWEŁ RYMARCZYK
Netrix S.A., Poland
ORCID iD: orcid.org/0000-0002-5990-4771

MICHAŁ JARMUŁ
WSEI University in Lublin, Poland
ORCID iD: orcid.org/0009-0000-4626-2741

ADAM HERNAS
Graduate School of Business – National-Louis University, Poland
ORCID iD: orcid.org/0000-0002-0519-6546

# ENHANCING CONVERSATIONAL AI WITH THE RASA FRAMEWORK: INTENT UNDERSTANDING AND NLU PIPELINE OPTIMIZATION

# UDOSKONALANIE KONWERSACYJNEJ SZTUCZNEJ INTELIGENCJI ZA POMOCĄ RASA FRAMEWORK: ZROZUMIENIE INTENCJI I OPTYMALIZACJA PROTOKOŁU NLU

## Abstract

**Purpose:** The primary objective of this research is to explore the effectiveness of the Rasa framework in developing a Natural Language Understanding (NLU) pipeline tailored for conversational AI systems. The study aims to advance our understanding of user intent recognition through structured NLU processes, optimizing the interaction between human users and AI agents.

**Methods:** The study utilizes a detailed configuration of the Rasa NLU pipeline, which orchestrates a sequence of steps from input text to structured understanding. Components such as tokenizers, featurizers, intent classifiers, and entity extractors are analyzed for their role in transforming unstructured user input into actionable insights. Advanced techniques, including machine learning algorithms, data augmentation, and pre-trained transformer models, are applied to enhance the accuracy and responsiveness of the AI system.

**Results:** Implementing the Rasa NLU pipeline allowed intent detection and entity recognition, particularly in complex scenarios with multi-intent queries. Communication within the Rasa NLU pipeline was effectively managed, ensuring seamless data flow between components, which preserved context and enhanced interpretability. The voice assistant developed with STT and TTS capabilities demonstrated robust real-time natural language processing, handling spoken queries efficiently. This confirmed the practical viability of using the Rasa framework for scalable and customizable conversational AI applications.

**Discussing:** The findings underscore the robustness of the Rasa NLU pipeline in handling diverse conversational demands and the flexibility of its components to adapt to different linguistic contexts. The research discusses the potential of integrating sophisticated NLU techniques to create more intuitive and responsive conversational agents, highlighting the critical role of context-aware processing in improving user interaction with AI systems.



## Streszczenie

**Cel:** Głównym celem tego badania jest zbadanie skuteczności narzędzia Rasa w rozwijaniu rozumienia języka naturalnego (NLU) przeznaczonego dla systemów konwersacyjnej AI. Badanie ma na celu pogłębienie zrozumienia rozpoznawania intencji użytkowników przez strukturyzowane procesy NLU, optymalizując interakcje między ludzkimi użytkownikami a agentami AI.

**Metody:** Badanie wykorzystuje szczegółową konfigurację narzędzia NLU Rasa, który określa sekwencję kroków od tekstu wejściowego do strukturalnego rozumienia. Analizowane są komponenty takie jak tokenizery, featurizery, klasyfikatory intencji i ekstraktory jednostek, pod kątem ich roli w przekształcaniu nieustrukturyzowanych danych wejściowych użytkownika w praktyczne wnioski. Zaawansowane techniki, w tym algorytmy uczenia maszynowego, augmentacja danych oraz wstępnie wytrenowane modele transformatorów, są stosowane w celu zwiększenia dokładności i responsywności systemu AI.

**Wyniki:** Wdrożenie narzędzia Rasa NLU umożliwiło wykrywanie intencji i rozpoznawanie jednostek, szczególnie w złożonych scenariuszach z zapytaniami zawierającymi wiele intencji. Komunikacja w ramach Rasa NLU była skutecznie zarządzana, zapewniając płynny przepływ danych między komponentami, co zachowało kontekst i zwiększyło możliwości interpretacji. Asystent głosowy opracowany z funkcjami STT i TTS wykazał się solidnym przetwarzaniem języka naturalnego w czasie rzeczywistym, skutecznie obsługując zapytania głosowe. Potwierdziło to praktyczną wykonalność wykorzystania narzędzia Rasa do skalowalnych i konfigurowalnych aplikacji konwersacyjnej sztucznej inteligencji.

**Omówienie:** Wyniki podkreślają solidność narzędzia NLU Rasa w obsłudze różnorodnych wymagań konwersacyjnych i elastyczność jego komponentów do adaptacji do różnych kontekstów językowych. Badanie omawia potencjał integracji zaawansowanych technik NLU w celu tworzenia bardziej intuicyjnych i responsywnych agentów konwersacyjnych, podkreślając kluczową rolę przetwarzania świadomego kontekstu w poprawie interakcji użytkowników z systemami AI.

**Keywords:** *Intent understanding, RASA framework, Natural language processing, Artificial intelligence, Entity extraction, Intent classification*

**Słowa kluczowe:** *Rozumienie intencji, narzędzie RASA, Przetwarzanie języka naturalnego, Sztuczna inteligencja, Ekstrakcja jednostek, Klasyfikacja intencji*

# Introduction

As an open-source machine learning framework, RASA has proven its versatility and ability to automate text and voice conversations across multiple platforms. Its integration with well-known portals such as Facebook Messenger, Slack, Google Hangouts, Microsoft Bot Framework, and Twilio and its compatibility with voice assistants such as Alexa or Google Home Actions underscores its adaptability in creating contextual assistants for a wide range of applications. The development of a university-specific chatbot using the RASA platform illustrates its usefulness in educational settings, allowing the retrieval of student information, grades, and attendance through user-friendly interfaces like Telegram, eliminating the need to download separate applications (Bhutada et al., 2023). In addition, applying RASA in health awareness through creating a multilingual chatbot demonstrates its potential to provide vital information in users' native languages, thereby improving the user experience and accessibility of information related to diseases, treatments, and precautions. The Chat-IBN-RASA project further illustrates RASA's capability in network management, allowing users to interact with network systems using a high-level language, thereby simplifying complex technical interactions (Bhattacharyya et al., 2024; Upadhyaya & Kaur, 2023). RASA's adaptability is also evident in its application to administrative services, where a conversational agent was developed to respond to student inquiries, demonstrating the effectiveness of machine learning techniques and natural language processing in improving service satisfaction (Luise et al., 2023; Poser et al., 2022). In addition, the integration of RASA with social robotics platforms for assistive robotics and elderly care highlights its potential to enhance human-robot interactions (Lemaignan et al., 2023). The growing reliance on chatbots for client handling and customer service, as well as their role in addressing mental health issues, further emphasizes the importance of technological advancements like RASA in improving various sectors, including health and customer service (Gandhi et al., 2024; Vivek et al., 2022). Through these diverse applications, RASA is a powerful tool in building contextual assistants capable of multi-layered conversations, offering significant opportunities for two-way interactions across different domains.

The construction of a Natural Language Understanding (NLU) pipeline within the Rasa framework for the development of conversational AI agents involves several critical components, each playing a unique role in transforming unstructured user input into structured data, categorized as intents and entities (Arevalillo-Herraez et al., 2022; Hwang et al., 2021; Mishra et al., 2022).

The pipeline begins with intent detection, a fundamental NLU task crucial for understanding the user's purpose. Techniques for enhancing intent detection include leveraging different text representations and applying machine learning algorithms to improve classification accuracy, especially in scenarios where training data might be scarce (El-Rif et al., 2023; Rizou et al., 2023). Following intent detection, slot filling and entity recognition extract structured information from the user's input. This involves identifying and classifying named entities within the text, which can be achieved through models that combine Intent Extraction (IE) and Named Entity Recognition (NER) in a joint approach, thereby improving the efficiency of understanding user queries (El-Rif et al., 2023). Moreover, the integration of stack-propagation techniques can further refine the accuracy of these tasks by leveraging intent information directly in the model, enhancing the semantic understanding of the input (Caldelli et al., 2023).

Advanced models based on global pointers and type-slot label interaction networks have been developed to address the challenges of multi-intent scenarios, where single user input may contain multiple intents. These models facilitate the handling of nested and non-nested slots, improving the pipeline's ability to classify intents and fill slots even in complex queries accurately (Moura et al., 2023).

Additionally, the pipeline benefits from data augmentation methods, which generate new training examples to improve model performance, mainly when original datasets are limited. This approach helps in expanding the variety of user intents the system can recognize and respond to accurately (Wang et al., 2023). Throughout the pipeline, the use of pre-trained transformer models and ensemble clustering techniques can significantly enhance text representation and intent classification, providing a robust foundation for the conversational AI agent to understand and interact with users effectively (Bercaru et al., 2023).

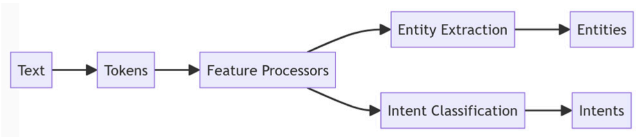
The NLU pipeline in Rasa's framework is a sophisticated assembly of components designed to interpret and analyze user input meticulously, employing a combination of intent detection, entity recognition, slot filling, and advanced modeling techniques to ensure accurate and efficient user intent understanding (Al-Tuama & Nasrawi, 2022; Benayas et al., 2023; Wan et al., 2023).

## Research Methodology

This study aims to elucidate the mechanism of intent understanding within the Rasa framework, emphasizing the construction of a Natural Language Understanding (NLU) pipeline during the development of conversational AI agents. In Rasa, the NLU pipeline delineates processing steps that transform unstructured user input (text) into structured data, categorized as intents and entities. This pipeline comprises various configurable and adaptable components facilitating intricate text analysis and interpretation. Below is an expanded description of these components, detailing their roles within the NLU pipeline and explicating their interactions.

The construction of the NLU pipeline in Rasa is meticulously defined within a config.yml file. This configuration file specifies each step in Rasa's pipeline for intent detection and entity recognition. The process initiates with the receipt of text as input. Subsequently, the defined modules within the pipeline sequentially analyze this input, continuing their processing until intents and entities are successfully extracted as output data. (Figure 1)

**Figure 1.** *General overview of the NLU pipeline*




The principal components of the Natural Language Understanding (NLU) pipeline, as delineated by the provided diagram, are as follows:

- Tokenizers segment the input text into discrete tokens, the fundamental units for further analysis.
- Featurizers: Responsible for translating tokens into feature vectors, these components effectively encode the textual data into a numerical format amenable to machine learning models.
- Intent Classifiers: They categorize the processed text into predefined intents, facilitating the determination of the overarching purpose or goal behind the user's message.
- Entity Extractors: These components are adept at isolating and identifying entities within the user's message, such as personal names or locations, thereby capturing critical information for contextually rich interactions.

The initial step involved segmenting the input statement into smaller text sections called tokens. This segmentation was essential before the feature extraction required for machine learning applications; therefore, tokenizers were typically listed at the commencement of the pipeline. Some tokenizers were also designed to enrich tokens with additional information. For instance, the functionalities of the spaCy library extended to the generation of lemmatized forms of the tokens, which could later be utilized by tools such as CountVectorizer.

Tokenizers operated by dividing each word in a statement into a separate token, and the output customarily consisted of a list of words. Punctuation tokens can also be obtained depending on the tokenizer and its configurations. The WhiteSpaceTokenizer was commonly employed in English language processing, whereas for other languages, such as Polish, the spaCy library was the preferred tool. It was crucial to acknowledge that tokenizers did not alter the base text but segmented it into tokens. For instance, implementing features was necessary if the pipeline aimed to encode text exclusively in lowercase.

Features were employed to generate numerical features for machine learning models. They existed in two distinct types:

- Sparse: These are typically generated by a CountVectorizer. The provided numbers can also represent subwords. An example is LexicalSyntacticFeaturizer, which creates window-based functions that are useful for entity recognition. In conjunction with spaCy, the LexicalSyntacticFeaturizer can also be configured to include parts of speech.
- Dense: It is composed of numerous pre-trained embeddings, usually derived from SpaCyFeaturizers or from huggingface through LanguageModelFeaturizers. The proper functioning of dense features also necessitates the inclusion of a suitable tokenizer within the constructed pipeline.

Consequently, sparse features return feature vectors with many missing values, such as zeros. Because these feature vectors typically occupy substantial memory, they are stored as sparse features. This approach ensures that only non-zero values and their positions in the vector are preserved. This saves considerable memory, allowing for training on larger datasets. All features can return two different types of features: sequence features and sentence features. Sequence features are matrices sized (number of tokens x feature size), each containing a feature vector for every token in the sequence. Sentence features are represented by a matrix (1 x feature size) containing a feature vector for the entire utterance. Sentence functions can be used in any model. Therefore, the appropriate classifier can decide which type of features to use.

To encode the word *Hello,* we can consider two main types of feature representations: sparse and dense. Let's illustrate how this word can be encoded using each method. Sparse Encoding: In sparse encoding, the word *Hello* might be represented in a high-dimensional space where each dimension corresponds to a word in the vocabulary, but most values are zero. For instance, if our vocabulary had 10,000 words and *Hello* was the second word, the sparse encoding could look like this: [0, 1, 0, 0, … , 0]. Here, the length of the vector is 10,000, and we have a ‚1' in the position corresponding to the word *Hello* and ‚0' elsewhere. Dense Encoding: For dense encoding, *Hello* would be represented as a lower-dimensional vector of continuous numbers. The dense vector might look like: [0.25, – 1.3, 0.8, …, 2.5].

In addition to features for tokens, we also generate features for the entire sentence. This is sometimes referred to as the CLS token. The sparse features in this token are the sum of all sparse features across individual tokens. Dense features are either the cumulative sum/average of word vectors (in the case of spaCy) or a contextualized representation of the entire text (in the case of hugging face models). Significantly, from the perspective of building custom solutions and project design, one can freely add components using custom feature creation tools.

Upon the completion of feature generation for all tokens and the entire sentence, these features can be forwarded to an intent classification model. Typically, the Dual Intent and Entity Transformer (DIET) model, proposed by Rasa, is employed and can handle both intent classification and entity extraction tasks. Moreover, it is proficient in learning from both token-level and sentence-level features. The DIET architecture is a multitask framework for intent classification and entity recognition. It is founded upon a transformer architecture shared between the two tasks. The sequence of entity labels is predicted using a Conditional Random Field (CRF) layer positioned on top of the transformer's output sequence, which correlates with the input sequence of tokens. For intent labels, the transformer's output for the entire utterance and intent labels are embedded within a unified semantic vector space. A softmax loss function maximizes the similarity with the target label and minimizes similarities with negative samples, thus fine-tuning the model's predictive accuracy.

Although the DIET algorithm is proficient in learning to detect entities, it is not universally recommended for every type of entity extraction.

For entities that follow a structured pattern, such as telephone numbers, the application of this algorithm may be unnecessarily complex. In such instances, a RegexEntityExtractor is advised. This component isolates entities by employing predefined lookup tables and regular expressions within the training data. It examines user messages for occurrences that match any entry from the lookup tables or conform to the specified regular expressions. Upon identifying a match, the corresponding value is extracted as an entity. Hence, employing multiple types of entity extractors within the pipeline is common to optimize performance for different entity categories.

This approach enables a structured and systematic analysis of user input. Each component in the pipeline—from tokenizers and featurizers to intent classifiers and entity recognizers—plays a critical role in dissecting and understanding the text. By configuring these components, developers can tailor the NLU pipeline to meet specific needs and enhance the system's performance in interpreting user interactions.

Moreover, this configuration-driven approach provides scalability and flexibility, allowing for easy modifications and upgrades to the NLU pipeline as requirements evolve or new technologies become available. The ability to precisely define and adjust the sequence of operations in the NLU pipeline through a single configuration file simplifies the development and maintenance of complex conversational AI systems, ensuring they remain robust and responsive to user inputs across various scenarios.

## Communication within the Rasa NLU Pipeline

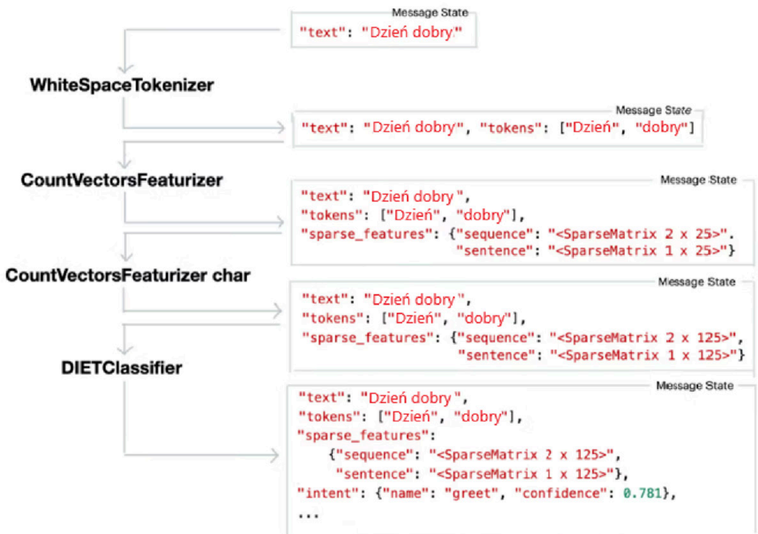
The constituent elements of the Rasa framework's processing pipeline exhibit a synergistic relationship, necessitating a holistic analysis to fully appreciate their operation. A systematic evaluation of an exemplar config.yml configuration file can provide substantial insights into the interplay of these components, elucidating their collaborative functions in facilitating language understanding tasks within the Rasa pipeline.

**Figure 2.** *Example `config.yml` file*

```
pipeline:
  – name: WhitespaceTokenizer
  – name: CountVectorsFeaturizer
  – name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: I
    max_ng ram: 4
  – name: DIETCIassifier
      epochs: 100
```

The Natural Language Understanding (NLU) pipeline represents a sequential array of components wherein each constituent is trained and processed in the order in which they are arrayed within the pipeline. This implies that the pipeline configuration can be envisaged as a linear sequence of processing steps that data must traverse. Each time a user interacts with the assistant, Rasa internally tracks the state of the utterance via a *Message* object. This object undergoes processing by each successive component within the pipeline. Figure 3 provides an overview of the events transpiring as a message undergoes processing.

**Figure 3.** *Overview of the processing of an example sequence*



Analyzing Figure 3 reveals several observations. The message initially presents itself as a container holding only the utterance of an end-user. Upon encountering the tokenizer, the message is segmented into tokens. It is imperative to note that while the tokens are depicted as strings in the illustration, they are internally represented by a *Token* object. As the message proceeds through the CountVectorsFeaturizer, sparse features are appended. As previously outlined, a distinction is made between features for sequences and the entire sentence. It is also notable that the dimensionality of sparse features increases after processing by the second feature. Subsequently, the DIETClassifier searches within the message for *sparse_features* and *dense_features* to make predictions.

Upon processing, it appends the intent predictions to the message object. Each time the message traverses a stage in the pipeline, the message object accrues new information. This implies that the addition of steps to the pipeline can be continued if there is a need to augment the message with further data. Consequently, additional entity extraction models can be incorporated. As demonstrated, each step in the pipeline can enrich the message with information. This means multiple entity extraction steps can be added, operating in parallel to append entities to the message.

## Create a voice assistant using selected tools

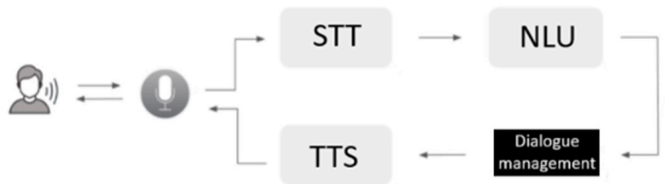
The domain of voice-operated assistants has observed a significant evolution, shifting from basic scripted interactions to advanced conversational interfaces that engage users through natural linguistic exchanges. The process describes the engineering feats achieved using the RASA framework, underpinned by the latest developments in NLP and ML, to foster a more intuitive form of human-computer dialogue.

A voice assistant's architecture is a complex assembly where each component ensures functionality and fluidity. The core components of our voice assistant architecture include (Figure 4.):

- User Interface (UI): The application through which the user communicates with the assistant (e.g., a web application, a mobile app, a smart speaker).

- Speech to Text (STT): This component takes the user's spoken input in audio format and converts it into textual data.
- Natural Language Understanding (NLU): This component reads the transcribed text from the user's speech to extract meaningful information (intents and entities) to understand the user's needs.
- Dialogue Management: This component decides the appropriate response to the user's query and generates this response in text form.
- Text to Speech (TTS): This component takes the text form of the assistant's response and creates a spoken version to be delivered to the user.

**Figure 4.** *The voice assistant's construction – general scheme*




Initiating the voice assistant's creation was characterized by a reasonable environment setup and the discerning selection of pivotal tools for its development. The environment was meticulously configured to facilitate the integration of various components and ensure an efficient workflow. To this end, the Python environment was isolated and tailored to act as a foundational layer for developing and operating the voice assistant within Docker containers.

A cornerstone of this environment was implementing the SpeechRecognition library, a decision dictated by its comprehensive STT capabilities. In parallel, the gTTS (Google Text to Speech) library was selected for TTS functionality, prized for its simplicity and its ability to produce a natural and intelligible voice output. The gTTS library was leveraged to transform the assistant's text responses into clear and natural-sounding speech, with customization options allowing for adjusting speech rate and pitch to enhance intelligibility and user comfort.

The limited availability of pre-trained Polish language models in speech processing presented a unique challenge. To address this, the tool selection was mainly oriented toward libraries that supported the Polish language or allowed custom model training. This strategic choice ensured that the voice assistant could competently handle Polish language inputs and outputs, a feature crucial for the target user base.

Docker containers played a critical role in the setup, encapsulating each service required by the voice assistant. These included the core RASA services and ancillary services like the action server and rasa-duckling, which were essential for understanding and processing structured content from user inputs. Docker ensured that each component could be developed, tested, and deployed consistently and isolated. It offered the benefits of containerization, such as scalability, portability, and efficient resource utilization, making the development process more streamlined and manageable.

The construction of the voice assistant began with setting up a controlled Python environment, crucial for the development of RASA and HTTP servers, which were deployed within Docker containers to ensure isolation and consistency. This setup facilitated the deployment of action servers and the rasa-duckling service, which handles the parsing of structured data from text. Docker Compose orchestrated these services, allowing them to operate synchronously with precise network configurations and port settings.

Within this environment, Dockerfiles were created to configure the action servers, specifying the necessary environment settings and dependencies to align with the RASA framework's requirements. The docker-compose.yml file was meticulously prepared to define services, network configurations, and volume mappings, emphasizing non-default ports to accommodate specific deployment needs.

Following the environment setup, the RASA framework was installed; configuration files were then crafted to define the assistant's dialogue flows, intents, entities, and other parameters, enhancing readability and ease of updates.

The HTTP server was also configured to handle cross-origin requests and provide necessary endpoints for the assistant's operation, ensuring seamless interaction with the UI and backend services. This setup created a robust platform ready for the subsequent phases of the assistant's lifecycle, including training and user interaction.

The User Interface (UI) development phase was approached with meticulous care, recognizing its vital role as the primary point of interaction between the user and the voice assistant. The paramount goal was establishing a UI that exemplified user-centric design principles, ensuring the interface was intuitive, inviting, and accessible to users with varying technical expertise.

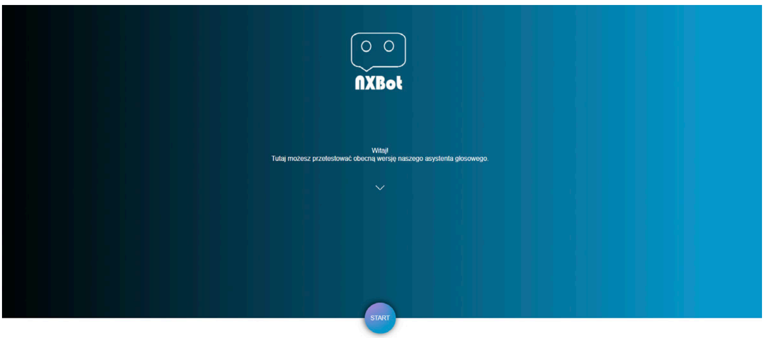
Initial modifications began with an assessment of the template's layout. The UI was restructured to foster a clean and uncluttered user experience, focusing on color schemes that enhance readability and reduce cognitive load. Navigation was streamlined to facilitate easy access to the assistant's features, ensuring users could intuit the pathway to perform tasks without explicit instruction.

Finally, the UI was subjected to rigorous user testing, involving participants from diverse backgrounds to gather a wide array of feedback. Based on this feedback, iterative refinements were made, with adjustments ranging from the micro-interactions to the overall flow of tasks within the UI.

A controlled series of integration tests was meticulously performed to validate the interoperability of the system's constituent components. The RASA framework's inherent compatibility with the SpeechRecognition and gTTS libraries facilitated their integration, yielding a voice assistant that demonstrated remarkable proficiency in handling and managing dialogues.

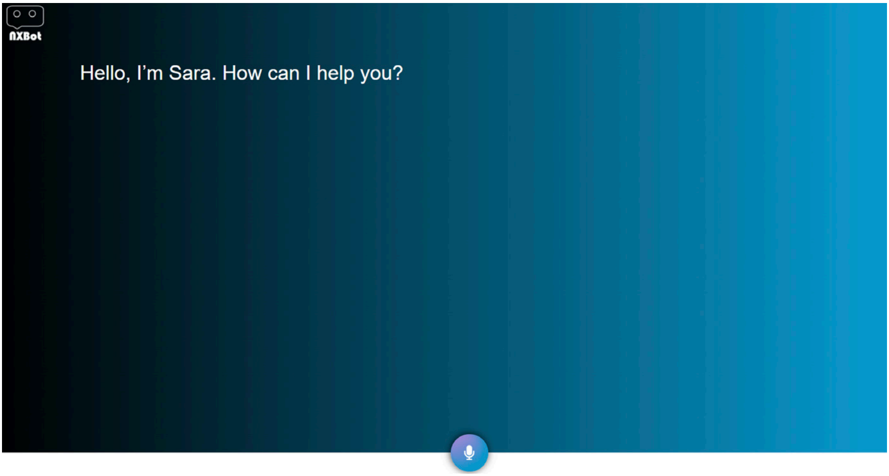
Post-integration, the voice assistant was tested, revealing its capability to engage users through its conversational prowess. A compendium of dialogues was provided to showcase the system's operational effectiveness, demonstrating the voice assistant's adeptness at interpreting user queries and delivering responses with high accuracy and contextual awareness.

**Figure 5.** *Welcome screen view*




The assistant's operation is shown below. After pressing the START button (Figure 5.), the Sara bot welcomes the user (Figure 6.).

**Figure 6.** *Welcome by the bot*




The following is a brief dialogue between the user (U) and the assistant (A):

*U: Hi Sara, how are you?*

*A: I'm great! Thanks for asking.*

*U: What can you tell me?*

*A: You can ask me how to get started with Rasa, the difference between Rasa and Rasa X, how to subscribe to our newsletter, or how to book a sales call.*

*U: OK, what is the difference between Rasa and Rasa X?*

*A: Rasa Open Source is a framework for building an assistant. Rasa X is a toolset built on top of Rasa Open Source that comes with a UI that helps you learn from honest conversations and improve your assistant. Rasa X is entirely optional — if you'd like, you can use Rasa Open Source on its own.* (Figure 7)

*U: Ok, thank you.*

*A: I'm happy to help. Is there anything else I can help you with?*

*U: Maybe you could tell me what time is it?*

*A: In an ever-expanding universe, the real question is: what time isn't it?*

*U: It's so funny*

*A:* 😍

*U: Ok, that's all. Thank you, bye.*

*A: Bye*

**Figure 7.** *Selected Assistant's Answer*


This approach enables the activation of the assistant, and by modeling its actions and the topics it discusses, a solution relevant to the specific audience can be created. Furthermore, the enhancement and development of custom Speech-to-Text (STT) and Text-to-Speech (TTS) models provide opportunities for improved pronunciation and understanding of spoken words.

## Conclusions

The research systematically explored the intricacies of developing a sophisticated Natural Language Understanding (NLU) pipeline within the Rasa framework, tailored for conversational AI agents. The meticulous examination of each pipeline component—from intent detection to entity recognition—reveals a robust architecture capable of accurately interpreting and responding to unstructured user inputs.

The study has highlighted the versatility of the Rasa framework in managing diverse conversational demands, employing advanced text representation techniques, and integrating machine learning algorithms to enhance the classification of intents and entities.

The pipeline successfully transforms user inputs into actionable data through the detailed configuration and interaction of tokenizers, features, classifiers, and entity extractors. This transformation is pivotal for generating responses that are not only accurate but also contextually relevant, thereby significantly improving user interaction with AI agents.

Moreover, the flexible architecture of the Rasa pipeline ensures that the system can be customized and scaled according to specific operational needs. This adaptability is crucial for the development of conversational agents that are tailored to meet the diverse requirements of users across different sectors.

## References


Al-Tuama, A. T., Nasrawi, D. A. (2022). Intent Classification Using Machine Learning Algorithms and Augmented Data. *2022 International Conference on Data Science and Intelligent Computing, ICDSIC 2022*, 234–239. https://doi.org/10.1109/ICDSIC56987.2022.10075794

Arevalillo-Herraez, M., Arnau-Gonzalez, P., Ramzan, N. (2022). Adapting the DIET Architecture and the Rasa Conversational Toolkit for the Sentiment Analysis Task. *IEEE Access*, *10*, 107477–107487. https://doi.org/10.1109/ACCESS.2022.3213061

Benayas, A., Sicilia, M. A., Mora-Cantallops, M. (2023). Automated Creation of an Intent Model for Conversational Agents. *Applied Artificial Intelligence*, *37*(1). https://doi.org/10.1080/08839514.2022.2164401

Bercaru, G., Truică, C. O., Chiru, C. G., Rebedea, T. (2023). Improving Intent Classification Using Unlabeled Data from Large Corpora. *Mathematics*, *11*(3), 769–769. https://doi.org/10.3390/MATH11030769

Bhattacharyya, S., Banerjee, J. S., Köppen, M. (2024). *Virtual Scribe for Visually Impaired Learners Through Interactive Speech-Based Application*. 111–121. https://doi.org/10.1007/978-981-99-7711-6_10

Bhutada, Dr. S., Rakshit, C., Vaishnavi, G., Varshitha, V. (2023). Chatra Paryesana (Student Enquiry System). *International Journal For Science Technology And Engineering*, *11*(6), 670–674. https://doi.org/10.22214/IJRASET.2023.53722

Caldelli, R., Castoldi, P., Gharbaoui, M., Martini, B., Matarazzo, M., Sciarrone, F. (2023). On helping users in writing network slice intents through NLP and User Profiling. *2023 IEEE 9th International Conference on Network Softwarization: Boosting Future Networks through Advanced Softwarization, NetSoft 2023 – Proceedings*, 545–550. https://doi.org/10.1109/NETSOFT57336.2023.10175400

El-Rif, E., Leivadeas, A., Falkner, M. (2023). Intent Expression Through Natural Language Processing in an Enterprise Network. *IEEE International Conference on High Performance Switching and Routing, HPSR*, *2023-June*, 14–19. https://doi.org/10.1109/HPSR57248.2023.10148046

Gandhi, R., Jain, P., Thakur, H. K. (2024). Mental Health Analysis Using RASA and BERT: Mindful. *Communications in Computer and Information Science*, *2054 CCIS*, 246–258. https://doi.org/10.1007/978-3-031-56703-2_20

Hwang, M. H., Shin, J., Seo, H., Im, J. S., Cho, H. (2021). KoRASA: Pipeline Optimization for Open-Source Korean Natural Language Understanding Framework Based on Deep Learning. *Mobile Information Systems*, *2021*. https://doi.org/10.1155/2021/9987462

Lemaignan, S., Cooper, S., Ros, R., Ferrini, L., Andriella, A., Irisarri, A. (2023). Open-source Natural Language Processing on the PAL Robotics ARI Social Robot. *ACM/IEEE International Conference on Human-Robot Interaction*, 907–908. https://doi.org/10.1145/3568294.3580041

Luise, R. S. A. De, Arevalillo-Herraez, M., Arnau, D. (2023). On Using Conversational Frameworks to Support Natural Language Interaction in Intelligent Tutoring Systems. *IEEE Transactions on Learning Technologies*, *16*(5), 722–735. https://doi.org/10.1109/TLT.2023.3245121

Mishra, D. S., Agarwal, A., Swathi, B. P., Akshay, K. C. (2022). Natural language query formalization to SPARQL for querying knowledge bases using Rasa. *Progress in Artificial Intelligence*, *11*(3), 193–206. https://doi.org/10.1007/S13748-021-00271-1/TABLES/5

Moura, A., Lima, P., Mendonça, F., Mostafa, S. S., Morgado-Dias, F. (2023). On the Use of Transformer-Based Models for Intent Detection Using Clustering Algorithms. *Applied Sciences*, *13*(8), 5178–5178. https://doi.org/10.3390/APP13085178

Poser, M., Küstermann, G. C., Tavanapour, N., Bittner, E. A. C. (2022). Design and Evaluation of a Conversational Agent for Facilitating Idea Generation in Organizational Innovation Processes. *Information Systems Frontiers 2022 24:3*, *24*(3), 771–796. https://doi.org/10.1007/S10796-022-10265-6

Rizou, S., Theofilatos, A., Paflioti, A., Pissari, E., Varlamis, I., Sarigiannidis, G., Chatzisavvas, K. C. (2023). Efficient intent classification and entity recognition for university administrative services employing deep learning models. *Intelligent Systems with Applications*, *19*, 200247–200247. https://doi.org/10.1016/J.ISWA.2023.200247

Upadhyaya, P., Kaur, G. (2023). Smart Multi-linguistic Health Awareness System using RASA Model. *International Conference on Sustainable Computing and Smart Systems, ICSCSS 2023 – Proceedings*, 922–927. https://doi.org/10.1109/ICSCSS57650.2023.10169275

Vivek, B., Maheswaran, S., Prabhuram, N., Janani, L., Naveen, V., Kavipriya, S. (2022). Artificial Conversational Entity with Regional Language. *2022 International Conference on Computer Communication and Informatics, ICCCI 2022*. https://doi.org/10.1109/ICCCI54379.2022.9740996

Wan, X., Zhang, W., Huang, M., Feng, S., Wu, Y. (2023). A Unified Approach to Nested and Non-Nested Slots for Spoken Language Understanding. *Electronics*, *12*(7), 1748–1748. https://doi.org/10.3390/ELECTRONICS12071748

Wang, Y., Yang, Z., Zhang, X. (2023). Improving NLP Accuracy with Stack-Propagation and Knowledge Distillation: A Joint Model for Intent Detection and Slot Filling. *Frontiers in Computing and Intelligent Systems*, *3*(2), 106–109. https://doi.org/10.54097/FCIS.V3I2.7525