Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2024) 000–000

www.elsevier.com/locate/procedia

18th CIRP Conference on Intelligent Computation in Manufacturing Engineering

Empowering Manual Assembly: Dialog System for Enhanced Customization and Efficiency of Cognitive Assistance Systems

Klaus Fink[a]*, Milos Solaja[a], Rüdiger Daub[a,b]

[a]*Fraunhofer Institute for Casting, Composite and Processing Technology IGCV*
[b]*Technical University of Munic, Institute for Machine Tools and Industrial Management (iwb)*

* Corresponding author. Tel.: +49 15120461454; fax: +49 82190678799. *E-mail address:* klaus.fink@hirschvogel.com

Abstract

Cognitive assistance systems (CAS) used in manual assembly aim to improve worker's productivity and accuracy by providing guidance and process control. Various CAS in manual assembly can be used simultaneously through integration into the existing IT-infrastructure. The integration of CAS is often enabled by custom software solutions that enable seamless synchronization between CAS. To increase productivity and accuracy, continuous reconfiguration of CAS is required to align it with specific tasks and individual preferences. This can be achieved by directly obtaining workers' opinions and thus allowing workers to decide what CAS are necessary to optimize individual performance and efficiency. This paper presents a dialog system based on Natural Language Processing (NLP) that leverages voice and text inputs to improve the reconfiguration of CAS.

Peer-review under responsibility of the scientific committee of the 18th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 10-12 July, Gulf of Naples, Italy

Keywords: Manufacturing system, Production, Human aspect

1. Introduction

To manage increasingly multifaceted tasks, workers in manual assembly use cognitive assistance systems (CAS). CAS consist of hardware and software components and provide guidance and process control for enhanced worker productivity and accuracy. [1] Various CAS in manual assembly can be used simultaneously through appropriate integration, often facilitated by custom software solutions, that enable seamless coordination and synchronization between these systems.

To optimize the human performance and ensure that CAS align seamlessly with the diverse capabilities and requirements of each worker, adapting various CAS to the individual skills and needs of workers is required. [2]

Efficient procedures have been researched and implemented to establish initial configuration of CAS, aligning them with the individual skills of the workers. Over time employees acquire competencies in assembling processes and develop personalized routines that increases their productivity. CAS can become redundant, or even disturbing to the worker, leading to the overall reduction in productivity. For optimal productivity and usability, continuous customization of CAS is required to align it with workers' specific tasks and preferences. [2]

There are two approaches to adapting CAS. The first involves the implementation of learning models that monitor workers' performance over time and adjust CAS accordingly [2]. The second approach involves obtaining the workers' opinion directly, allowing them to decide which CAS are necessary to optimize their individual performance and efficiency. To allow workers to customize CAS based on individual needs, a suitable interface which enables

Peer-review under responsibility of the scientific committee of the 18th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 10-12 July, Gulf of Naples, Italy

reconfiguration is required. While systems designed for synchronizing CAS usually come with menus for CAS reconfiguration, this approach is not particularly efficient and intuitive. It demands a considerable amount of time and necessitates workers to be familiar with the system to effectively navigate through menus and make adjustments. In addition, workers are typically engaged in assembling tasks, and additional system handling during the assembling process can be disruptive, potentially resulting in a decline in overall productivity.

The idea of this paper involves development of a dialog system, aimed at enhancing CAS reconfiguration using voice or text inputs and its implementation into existing infrastructure.

1.1. Structure of the paper

Chapter 2 introduces the current state of research and technology regarding dialog systems in manufacturing processes. Chapter 3 describes the methodology for finding the optimal interaction modality and tools required to develop a dialog system. Chapter 4 focuses on the development process and implementation of the dialog system into existing infrastructure. Chapter 5 presents the accuracy evaluation of the developed dialog system. Chapter 6 discusses the extent to which the central research question and objectives are covered, summarizes the findings, and discusses the need for further research.

2. State of the research and technology of dialog systems in manufacturing processes

AI has already been implemented in manufacturing for instance by intelligent robotic arms and computer vision systems. Still dialog systems are not much involved in this field. The reason could be that dialog systems often struggle with understanding user's context. These related to multi-step tasks using only text input, which often results in irrelevant responses. It is more common to find multi-modal systems that can accept various types of inputs such as, text, video, audio or images. [3]

Dialog systems in general can find their application within manufacturing process in three stages, namely orientation training, assembly line and repair and maintenance. Dialog systems can serve as a supervisor in a factory that delivers a lecture to the novices, or act as a question-answering system that workers can ask for help when encountering difficulties. The latter use case can be utilized both in assembly line as well as during repair and maintenance. [3]

One example of a dialog systems designed for manufacturing process and assembly line represents the YOLO-based Masker with CNN (YMC) model developed by the author CHEN ET AL. [3]. The YMC model is a task-oriented dialogue system, that aims to answer workers questions during assembly process of a robot. [3]

There are commercial solution, such as *Andi* developed by company Andonix, which is described as “the AI-powered manufacturing chatbot”. Its goal is to provide workers with immediate access to expert guidance, support, and built-in knowledge of various manufacturing skills. [4]

The state of science and technologies shows that there are no dialog systems that are used in CAS in manual assembly and their reconfiguration.

3. Integrated methodology for developing an efficient dialog system in manual assembly

3.1. Prerequisites for assistance system for manual assembly

Today workers usually perform assembling of different product types and each one of them consists of many assembly tasks which have unique ID. To increase efficiency and decrease cognitive load, worker can be additionally supported by different CAS in every task.

Each worker possesses a unique competence profile and varied experiences, the initial CAS setup customized to individual needs becomes important. The definition of this initial setup has already been investigated and described in [5]. It consists of a set of questionnaires that aim to collect the data for the creation of the competence profile of the employee. [5]

As assembly tasks have repetitive nature, workers over time usually establish their own routines that enable them to achieve optimal performance and reduce dependence on CAS. Therefore, CAS have be continuously adapted according to the workers needs to provide optimal level of support.

The dialog system presented in this paper enables intuitive CAS adaptation according to personal preferences at any time during assembling.

Figure 1 demonstrates how a dialog system can be implemented with existing infrastructure and how it works. The infrastructure includes a configurator which represents custom software used for CAS configuration and synchronization.

In this setting, the worker can express his desired CAS reconfigurations in two ways. The first one involves explicit commands to change CAS status in a task. The second approach is that the worker can provide feedback to the dialog system once assembly has been successfully completed. The worker thus can give feedback to the dialog system on how satisfied he is with the current setup of the CAS. Based on the sentiment of feedback the system determines whether CAS reconfiguration is required or not. To make this work, the dialog system requires a sentiment analyzer which can label sentiment of the feedback as positive, meaning the systems should remain unchanged and the worker still requires the same amount of support. Or negative, meaning the worker requires different level of assistance and some CAS should be reconfigured.

The development of the dialog system and sentiment analyzer is constrained by the non-existing of large-scale datasets and limited computational capacity for model training. Furthermore, due to data privacy concerns, it is required that the system, including all components, run on a local machine, requiring simple yet accurate models and architectures with the ability to communicate with external devices for CAS control.

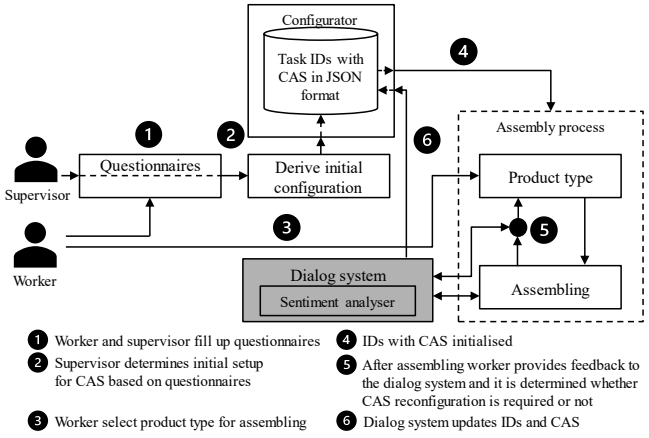

Fig. 1. Illustration of existing infrastructure together with a dialog system.

3.2. Optimizing interaction strategy for dialog systems in manual assembly

In the context of the specific environment in which the dialog system is to be implemented, as well as varying levels of technological expertise and different age among the workers who will be interacting with the system, selecting the appropriate interaction mode between workers and system presents a unique challenge. To address these challenges, the author has analyzed the three commonly used interaction modes based on his own experience, as illustrated in Figure 2.

Menu-based systems, while offering predefined input options for ease of use, may face limitations as complexity grows. In scenarios where menu structures become intricate, users may find it challenging to efficiently navigate and locate specific options. Another issue with menu-based systems is that the worker would only be able to make direct changes by picking options from the menu without possibility to use arbitrary inputs and feedback.

On the other hand, text-based and voice-based systems offer significantly greater flexibility, enabling users to express themselves in natural language when providing different commands and feedback to the system. In manual assembly scenarios, where workers' hands are occupied with various tasks, the practicality of voice-based systems becomes evident. These hands-free options simplify interaction and enhance usability in such environments.

When it comes to industrial settings particularly, sensitivity to environmental noise can pose a challenge for voice assistants. In the context of manual assembly processes, while power tools can generate some noise, it is generally less disruptive compared to the extensive machinery present on standard assembly lines. In general, environmental noise can be significantly mitigated by implementing robust noise-canceling technology, which enables voice assistants to recognize and process commands accurately despite noise in most cases.

When evaluating responses, text and voice-based systems also excel over menu-based ones in providing more flexibility and customization of reactions in response to input and voice assistants can even deliver responses in spoken language.

In assembly processes, the utmost priority is placed on worker safety and well-being, with employees often utilizing protective gear. Protective gloves can pose challenges for touch screen interactions and even typing on keyboard for text-based interaction can be unpractical.

When considering the development process, menu-based systems are the most cost-effective and easiest to develop and require no additional hardware. On the other hand, text-based and voice-based systems, which are based on machine learning (ML) and deep learning (DL) models, require additional hardware like a keyboard or microphone for interaction, but offer the potential for extended functionality such as question-answering capabilities.

The assistant developed in this paper aims to be used by diverse user base, consisting of workers with varying levels of technological expertise and spanning different age groups. As a result, ensuring practicality and user-friendliness to all users regardless of their prior technical knowledge and experience should have highest priority.

Despite certain advantages that menu-based systems have, voice and text-based systems are characterized by their capability to understand natural language eliminating the need for complex menus. This allows for the provision of direct commands and customization of CAS based on the individual worker's needs. It also enables the incorporation of arbitrary feedback, allowing the system to autonomously assess whether adjustments are necessary. All this together creates simplified, more accessible, and seamless user experience to all users regardless of their prior technical knowledge. Hence voice inputs complemented by the option of text inputs can be seen as the optimal way of interacting with assistant for this specific use case. Responses are to be displayed on the screen in the textual form to avoid necessity for wearing additional headphones that might not be practical and safe for the entire duration of a manual assembly shift.

	MENU-BASED	TEXT-BASED	VOICE-BASED
Impact of background noise	●	●	○
Ease of use wearing protective gloves	◑	◑	●
Ease of use for different age groups	◑	◑	●
User interaction practicality during assembling process	○	◑	●
User interaction input response time	◑	○	●
Content richness of provided inputs and outputs	○	◑	●
Flexibility in inputs and conversation	○	●	●
Additional hardware required	●	◑	◑
Extension with additional options and features (e.g. QA)	○	●	●

Legend:
○ non-fulfillment of the requirements
◑ partial fulfillment of the requirements
● fulfillment of the requirements

Fig. 2. Comparison of dialog system interaction modes in manual assembly.

3.3. Optimal architecture, components, and tools for dialog system development

The dialog system in this paper is developed based on *Rasa* infrastructure using custom dataset. *Rasa* represents an open-source platform designed to simplify the creation of custom dialog systems based on a pipeline architecture. [6] It offers a high degree of flexibility that allows users to customize the system's functionality to align it with specific requirements. [6] A pipeline architecture enables individual component development, fine-tuning and simplifies synthetic dataset generation consisting of separate utterances. [7]

To process voice inputs, it is necessary for dialog system based on *Rasa* to be equipped with an additional Speech-to-Text (STT) component. Once transcribed, the resulting text can be used analogously to text inputs. Among different open-source STT options that exist, *Whisper* has been selected as optimal STT for this dialog system as testing on different examples has proven optimal performance in this specific domain. *Whisper* also transcribes numbers directly as digits and hence requires no additional processing steps for ID recognition. [8]

3.4. Datasets for dialog system and sentiment analyzer training

To enable reconfiguration of the CAS through voice or text inputs, a comprehensive set of intents that align with the required functionalities has been defined. Examples for intents used for training with focus on enabling or disabling CAS were manually crafted. At the same time various synonyms associated with CAS names have been collected, along with diverse formats for task IDs. The dataset was subsequently expanded with all possible combinations between examples and collected entities and every entity was appropriately labeled in every example to enable training of transformative *DIET* model [9, 10].

The dataset used to train sentiment analyzer consists of labeled examples with both positive and negative feedback regarding the setup. Initially number of straightforward examples were manually collected.

3.5. Sentiment analyzer for dialog system

Different methods exist that are used for sentiment analysis such as sentiment lexicons, constructing sentiment analyzers from the ground up using ML and DL techniques, and fine-tuning pretrained models as Bidirectional Encoder Representations from Transformers (*BERT*) [11, 12, 13, 14, 15].

Because of limited dataset size for this use case, only the last approach was considered using smaller *BERT* model called *DistilBERT* [14]. To evaluate the performance of different options for sentiment analyzer small dataset consisting of ten positive and ten negative synthetic feedback was generated. All these sentences were domain specific and not previously included into training dataset. Without any fine-tuning, *DistilBERT* accurately determined sentiment for 16 out of 20 sentences. To further improve accuracy, transfer learning on the *DistilBERT* model was performed where all weights were updated using labeled dataset with around 200 synthetic feedback samples. This improved accuracy to 18 out of 20 sentences and this sentiment analyzer was further used and implemented into the dialog system.

4. Development and integration of a dialog system for manual assembly into existing infrastructure

4.1. Implementation of Rasa-based dialog system

The dialogue system utilizes the default *Rasa* pipeline, with one additional custom component, a sentiment analyzer. The only modification to the default *Rasa* pipeline lies in the utilization of a *DistilBERT* model as the featurizer, aimed at enhancing the model's robustness in handling spelling errors or encountering unseen utterances with similar meanings to those in the training dataset.

Rasa natural language understanding (NLU) for this dialog system was trained on the explained dataset, and Table 1 provides a list of the intents and entities incorporated within NLU.

To address the challenge of users occasionally providing incomplete information *Rasa* introduced the concept of forms [6] which is also implemented into this dialog system. It is triggered once negative sentiment is recognized to collect required information from user regarding desired CAS reconfiguration including CAS name, task ID and whether system should be switched on or off.

Table 1. Intents and entities for Rasa-based dialog system.

	Intents	Entities (Slots)
Assistance system adaptation	• switch_on_system_in_id • switch_off_system_in_id • switch_on_system • switch_off_system • feedback_provided • get_active_systems	• **taskID** • **assistanceSystem** • **onoff** • sentiment
Other	• greet • goodbye • thank • ask_whoisit • ask_languagesbot • bot_challenge	

Another element ensuring the functionality of the *Rasa* dialog system involves custom actions. [6] Table 2 contains the list of actions that are implemented into this dialog system with corresponding functionality description.

Table 2. Implemented custom actions in Rasa-based dialog system.

Actions	Functionality
• action_switch_off_system_in_id • action_switch_on_system_in_id • action_switch_off_system • action_switch_on_system	• Take extracted slots (e.g. taskID and assistanceSystem) and send API request to adapt assistance system in ID • In case ID is not specified, take the current task ID and perform changes
• action_get_active_systems	• Request information from control system regarding assistance systems
• action_reset_all_slots • get_info_adapt_systems_form	• Action that resets all slots • Form used to gather information and fill required slots before seding API request

To guide the training of a dialogue management model, *Rasa* necessitates the stories to be defined that act as representations

of dialog flows between user and the system and enable capturing contextual nature of conversations and learning sequential patterns of interactions. [6] For this dialog system 25 different stories were composed that mimic potential interactions between worker and dialog system.

4.2. Entity validation for improved user experience and entity recognition

In dialog system's pipeline, entity extraction is performed by *DIET* model. [6, 10] A potential challenge arises when user provide false information, provided entities contain typos, or STT engine fails to recognize entity names or their parts accurately. All this can result in an inability to implement appropriate CAS changes.

One potential solution for CAS names involves lookup tables. However, due to impracticality in covering all potential cases an alternative approach based on string similarity was considered. One such algorithm is *Jaro* similarity, which relies on the number of matching characters and transpositions between strings. This similarity is general performs well for strings with slight spelling variations and it has been implemented into dialog system to assess the similarity between extracted entities that refer to CAS and default CAS names. [16]

To accommodate various names users might use to refer to each CAS, a comprehensive list with synonyms has been additionally generated for each CAS. When an entity is extracted, the system performs a similarity computation between the extracted entity and every synonym within the respective CAS synonyms list. The maximum similarity value obtained is considered the overall similarity between the extracted entity and the considered CAS. This process is systematically repeated for each CAS. To determine the CAS to which the extracted entity belongs, the maximum similarity value across all CAS is identified, and the corresponding CAS is considered to be the one to which the extracted entity refers to. To further ensure that recognized entities closely align with valid CAS names and to minimize false positives, a predefined threshold of 0.7 is set that recognized entity needs to surpass to be considered as valid.

Besides CAS names, another essential entity required to customize CAS is the task ID. After the extraction of the entity ID, the dialog system initiates validity checks to ensure that the extracted number corresponds to a valid task ID. When prompt contains an ID, the action, before triggering requested changes, examines whether provided ID is present in the list with all task IDs. Only if the provided ID is found in the list, the requested change is executed. Conversely, if action for switching on or off CAS without specified ID is triggered, the action retrieves information about the currently active task, and the changes are applied accordingly.

Additionally, every action includes checks to ascertain whether CAS specified for switching on or off is already in the specified state. If the system is already in the desired state, the user is informed of this status. Otherwise, the state of the CAS is appropriately adjusted.

4.3. Dialog system implementation into existing infrastructure

Overall architecture consists of three main components as illustrated in Figure 3.

Interaction with the dialog system occurs through a graphical user interface (GUI) which is integrated into the Configurator. Two interaction modalities exist, voice inputs, involving STT conversion, and direct text inputs. Based on assistant's prediction regarding intent of the provided input, appropriate action on action server is triggered using application programming interface (API) requests. Upon completion of the predicted action, the action server returns JSON payload with responses which are directed back to the user. [17]

The interaction between Configurator and action server is also enabled using API requests. For instance, when CAS reconfiguration is requested, API request is sent from action server specifying necessary change, task ID, and name of CAS to be modified. Once this request is received, corresponding change within CAS Configuration is made and user is informed upon its execution.

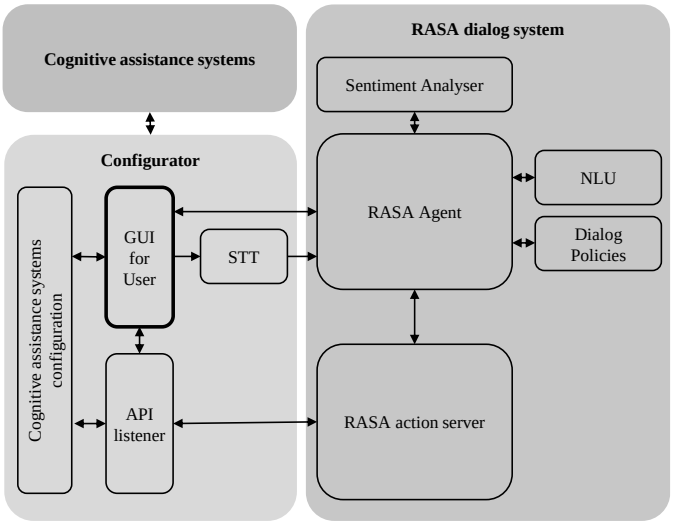

Fig. 3. Overview of the system architecture.

5. Accuracy evaluation of the developed dialog system

When focusing on NLU independently, *Rasa* provides a testing framework for accuracy assessment using NLU dataset. [6] The results for the developed dialog system reveal that NLU, trained on the specific dataset, correctly recognizes intents in most cases. Nevertheless, the used dataset primarily consists of similar examples without diversity in user query formulations and consequently this cannot be considered representative of the overall accuracy of the dialog system.

To extend the evaluation to unseen data, 20 test stories were generated that cover various scenarios where the dialog system is intended to be used. Queries within stories are further rephrased using ChatGPT to ensure they are unbiased and unseen as much as possible and additional noise, in the form of random typos, is introduced. Figure 4 illustrates few examples used for testing. It showcases instances where the system identified intent and executed appropriate actions correctly, as well as situations in which its performance was less proficient.

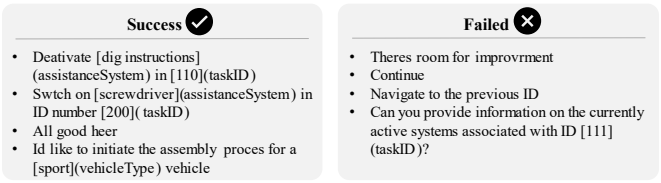

Fig. 4. Successfully identified and failed queries.

After testing, it can be noted that the dialog system performs reasonably well when queries remain within the predefined domain and functionality. The system successfully categorized 17 out of 20 test stories accurately even in the presence of typos. However, given the richness and complexity of language, ongoing detection of scenarios where the system does not perform as expected is necessary. The system requires continuous maintenance, with instances of failure being consistently incorporated into training dataset for continuous improvement.

6. Summary and outlook

The developed dialog system provides required functionalities necessary for smooth CAS reconfiguration. Despite the positive aspects observed in the developed dialog system, it has also revealed certain shortcomings, indicating opportunities for further improvement. Due to many components and models that run in parallel, the dialog system can require a considerable amount of time to execute because of speech-to-text transcription that is performed locally. This is also contingent upon the hardware where the dialog system runs.

Another concern represents dialog system's accuracy. Despite the promising accuracy evaluation, the dialog system relies on a relatively limited dataset consisting of examples without substantial difference between them. As a result, there is a considerable possibility of coming across situations and user requests that have not been covered in the dataset, potentially leading to non-optimal performance of the dialog system. A suggested improvement involves collecting and incorporating real examples into the dataset for NLU, dialog stories, and feedback. These instances could also serve as guidelines for generating a more comprehensive synthetic dataset.

In essence, the developed dialog system still needs to undergo testing optimally in a manual assembly environment, as well as user acceptance evaluation. Additionally, user acceptance evaluation can be used to collect user's suggestions for further improvement.

In general, the idea of integrating dialog systems into manual assembly processes holds significant potential, offering possibility for diverse applications. Besides CAS customization, dialog systems can be further extended with additional question answering (QA) feature that supports inquiries related to the assembly process or providing instructions and answering questions related to the handling of the system itself. Furthermore, textual answers can be in the form of step-by-step instructions, guiding workers through the resolution of potential challenges they may encounter throughout assembling process. Such implementation has the capacity to enhance and streamline processes and ultimately reduce costs. As such, dialog systems in manual assembly emerges as a promising field for deeper exploration in future research.

References

[1] Pokorni B, Popescu D, Constantinescu C. Design of Cognitive Assistance Systems in Manual Assembly Based on Quality Function Deployment. Applied Sciences, Volume 12.
[2] Sehr P, Moriz N, Heinz-Jakobs M, Trsek H. Am I Done Learning? - Determining Learning States in Adaptive Assembly Systems. 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). 6-9 Sept. 2022.
[3] Chen TY, Chiu YC, Bi N, Tsai RTH. Multi-Modal Chatbot in Intelligent Manufacturing.
[4] Andonix. Meet Andi: The AI-Powered Manufacturing Chatbot Revolutionizing Factories [Press Release]. https://andonix.com/press-release-meet-andi-the-ai-powered-manufacturing-chatbot-revolutionizing-factories/.
[5] Fink K, Ziegler A, Härdtlein C, Berger C, Daub R. Assessing the Human Competence and Individual Support Level in Manual Assembly through Cognitive Assistance Systems In: 17th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Gulf of Naples, Italy.12-14 July.
[6] Rasa. Rasa Documentation. https://rasa.com/docs/rasa/.
[7] Zhang Z, Takanobu R, Zhu Q, Huang M, Zhu X. Recent advances and challenges in task-oriented dialog systems. Science China Technological Sciences, Volume 63.
[8] Radford A, Kim JW, Xu T, Brockman G, McLeavey C, Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. https://arxiv.org/abs/2212.04356.
[9] Raj S. Building Chatbots with Python: Using Natural Language Processing and Machine Learning. Apress Publishing.
[10] Rasa. Rasa Components. https://rasa.com/docs/rasa/components.
[11] Dang NC, Moreno-García MN, De la Prieta F. Sentiment Analysis Based on Deep Learning: A Comparative Study. Electronics, Volume 9.
[12] Liu B. Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers.
[13] Vajjala S, Majumder B, Gupta A, Surana H. Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems. O'Reilly Media.
[14] Ravichandiran S. Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT. Packt Publishing.
[15] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
[16] Naumann F, Herschel M. An Introduction to Duplicate Detection (E-Book).
[17] Rasa. Rasa Action Server. https://rasa.com/docs/rasa/action-server.