Domain Expert Platform for Goal-Oriented Dialogue Collection

Didzis Gosko **Arturs Znotins** **Inguna Skadina**
Normunds Gruzitis **Gunta Nespore-Berzkalne**
Institute of Mathematics and Computer Science,
University of Latvia
Raina bulv. 29, Riga, Latvia
firstname.lastname@lumii.lv

Abstract

Today, most dialogue systems are fully or partly built using neural network architectures. A crucial prerequisite for the creation of a goal-oriented neural network dialogue system is a dataset that represents typical dialogue scenarios and includes various semantic annotations, e.g. intents, slots and dialogue actions, that are necessary for training a particular neural network architecture. In this demonstration paper, we present an easy to use interface and its back-end which is oriented to domain experts for the collection of goal-oriented dialogue samples. The platform not only allows to collect or write sample dialogues in a structured way, but also provides a means for simple annotation and interpretation of the dialogues. The platform itself is language-independent; it depends only on the availability of particular language processing components for a specific language. It is currently being used to collect dialogue samples in Latvian (a highly inflected language) which represent typical communication between students and the student service.

1 Introduction

Modeling of human-computer interaction through dialogue systems and chatbots has raised a great interest among researchers and industry already since the time when the first chatbot Eliza (Weizenbaum, 1966) was created. This interest has become viral after the successful introduction of Siri (Bellegarda, 2013). Today, virtual assistants, virtual agents and chatbots are present everywhere: on mobile devices, on different social networking platforms, on many websites and through smart home devices and robots.

The virtual conversational agents are usually implemented as end-to-end neural network models, or their components are implemented through neural network architectures (Louvan and Magnini, 2020). Such architectures require creation of datasets that represent various dialogue scenarios, as well as knowledge of the specific domain. This information has to be provided in specific data formats that in many cases are too complicated for domain experts. Moreover, the required training datasets usually include various annotation layers, such as named entities, dialogue acts, intents, etc. The creation of such datasets is a complex task, and the datasets are not completely isolated and abstracted from the particular dialogue system. Thus, domain experts that are involved in the creation of the datasets must have a high-level understanding of the overall structure of the dialogue system and its components, and how it is reflected in the annotated dialogue samples.

This demonstration paper address this issue by presenting a web-based platform for the creation and maintenance of dialogue datasets [1]. The interface of the platform is very simple and high-level: it allows a domain expert without detailed technical knowledge of the underlying dialogue system to create and update a representative training dataset, as well as to maintain the underlying database of domain- and organisation-specific information that will be used for question answering. The platform provides tools for the creation of goal-oriented dialogue systems, in particular:

- creation of datasets for dialogue systems that provide (or generate) responses depending on user input, intents and on the previous actions of the dialogue system;
- creation of datasets for dialogue systems that cover one or several topics;
- slot filling, including slot filler (e.g. named entity) normalization and annotation;
- creation and maintenance of slot filler aliases;

[1] http://bots.ailab.lv/

- creation and maintenance of knowledge base and interactive response selection;
- response generation, including the generation of inflectional forms for syntactic agreement.

Our platform not only supports collection of dialogue scenarios, but also simulates prototypical interaction between human and computer. The tool has been successfully used for the creation of a dialogue dataset for the virtual assistant that supports the work of the student service in relation to three frequently asked topics: working hours and contacts of the personnel and structural units (e.g. libraries), issues regarding academic leave, as well as enrollment requirements and documents to be submitted (Skadina and Gosko, 2020).

In the next chapters of this paper, we describe our motivation to develop the platform, its overall architecture and main components, and the domain expert interface and its main components.

2 Background and Motivation

For English and several other widely used languages, many publicly available dialogue datasets have been created and are reused for different research and development needs (e.g., Budzianowski et al. (2018), Zeng et al. (2020)). However, in the case of less-resourced languages, only few or no training datasets are available (Serban et al., 2018). To our knowledge, there is no publicly available dataset for Latvian, that could be used for goal-oriented dialogue system modelling. To overcome this obstacle, some research groups machine-translate existing English datasets into the low-resourced languages, while others try to build training datasets from scratch. When possible, crowd-sourcing, including gamification (Ogawa et al., 2020), is used as well. However, there in no best recipe for obtaining or collecting dialogue samples for a specific NLP task (in our case, dialogue modeling) for a less-resourced language with a relatively small number of speakers.

The motivation of our work is the necessity to build virtual assistants in less-resourced settings. The practical use case to test the platform has been the everyday communication between students and the student service of the University of Latvia. Since this communication has never been intentionally recorded, we started with the analysis of data retrieved from an online student forum to identify the most common topics, question and answer templates, and the typical dialogue scenarios. For the demonstration purposes, we have chosen three common topics: working hours, academic leave, and enrollment requirements. Elaborated and annotated sample dialogues constituting the training dataset have been specified by a domain expert using the dialogue management platform presented in this paper.

Since we focus on goal-oriented virtul assistants, the Hybrid Code Networks (HCN) architecture has been selected for the implementation (Williams et al., 2017) allowing us to combine recursive neural networks (RNN) with the domain-specific knowledge and action templates of the dialogue system. The concrete dialogue system is implemented within the DeepPavlov framework[2].

3 Overall Architecture and Components

The platform presented in this paper is designed to support three use cases:

1. To create and gradually improve a collection of dialogue samples necessary for developing and testing a goal-oriented dialogue system.
2. To support (re-)training of a goal-oriented dialogue system.
3. To support dialogue testing in the inference mode. Training and running a dialogue system in the inference mode is performed through the DeepPavlov framework by passing the goal-oriented bot model configuration along with relations to other objects that are specific to the platform.

Figure 1 illustrates the architecture of the platform. Apart from the domain expert user interface described in detail in Section 4, key components of the platform are four databases for storing the dialogue scenarios, the relevant entities and their aliases for slot filling, the required external knowledge for question answering, and reusable templates for response generation.

3.1 Dialogue Database

Dialogues created by the users of the platform (i.e., by domain experts not end-users) are stored in the SQLite database *Dialogues* to support concurrent modification. The dialogue database stores potential end-user utterances together with the respective

[2]`https://deeppavlov.ai/`

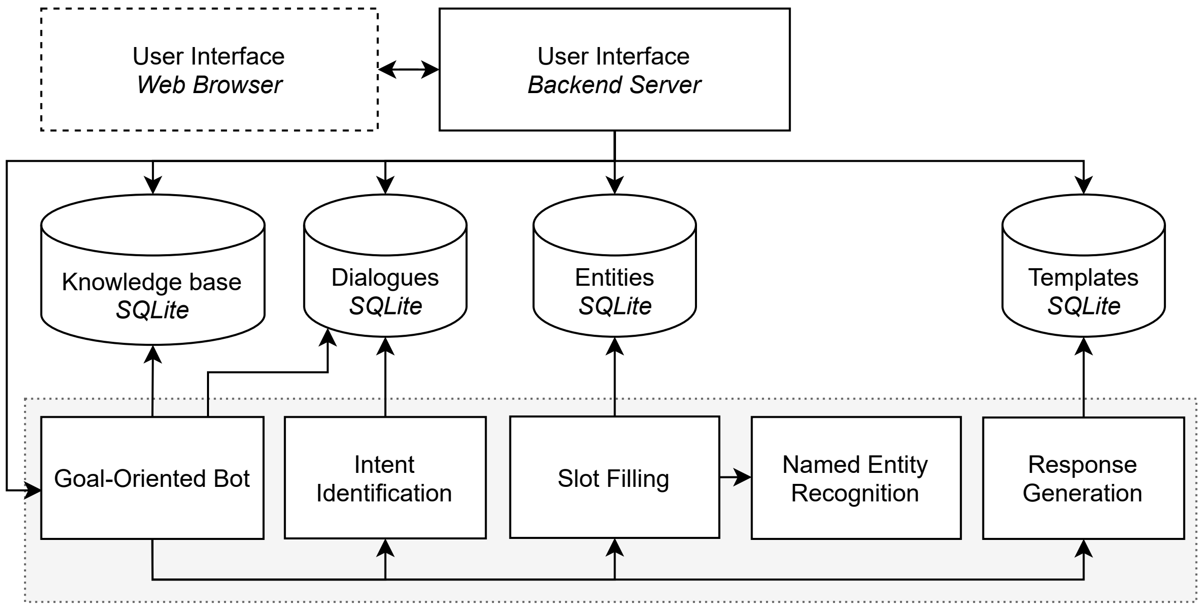


Figure 1: Architecture of the platform. The selected (grey) part of the diagram is language-specific and can be replaced or removed entirely.

intents, slot values and the corresponding bot actions.

Intents for the particular dialogue dataset are defined in a separate view of the platform's interface. The predefined intents are linked to utterances during the dialogue writing process (for details, see Section 4) and later used for training. In our demonstration dialogue system, we use a Keras classification model with Latvian fastText word embeddings for intent detection.

The configurator of the platform uses a custom data reader that reads training data from a custom-schema SQLite database. The reason why SQLite is used is the high modification rate produced by the platforms's user interface for dialogue editing.

3.2 Knowledge Base

The database *Knowledge base* stores the external knowledge that is necessary for the dialogue system to provide the answers to the end-users. Such knowledge is usually dynamic and can change while the dialogue system is deployed (e.g. working hours of the university personnel in our demonstration case).

3.3 Entity Database

For our use case, the named entity recognition (NER) model combines a neural network model and a rule-based model.

The neural network model is based on Latvian BERT word embeddings (Znotins and Barzdins, 2020). To support entity classes of a particular domain, the NER model is trained on a larger general-domain dataset (Gruzitis et al., 2018; Paikens et al., 2020) and a smaller domain-specific dataset. The combined model allows to recognize not only commonly used entity classes like persons, locations and organizations, but also domain specific entities like job positions and working hours.

The rule-based NER is based on the Aho-Corasick algorithm (Aho and Corasick, 1975) with additional regular expression rules to ensure entity detection in various inflectional forms, as well as detection of very specific domain entities like room names and specific job positions that would not be recognised otherwise due to the limited amount of training data.

In our demonstration dialogue system, a custom slot filler is implemented, which relies on normalized entities returned by the NER module to be directly filled in the respective slots.

The normalization is done in two steps. First, after the recognition of named entities (NE), an external NE normalization service is called, which provides base forms for both single-word entities and multi-word entities. Second, the database *Entities* is consulted to align the recognised and normalised entities (entered by the end-user) with the corresponding entities in the database. This also includes resolving NE aliases (for more details see Subsection 4.5).

3.4 Template Database

Responses to the end-user are generated using a template-based approach which depends on the recognised intent and slots. The response templates support additional markup for slot filler inflection that are replaced with the correctly inflected word forms during the response generation step. In our demonstration system, word forms are inflected using a Latvian morphological analyser (Paikens et al., 2013) as an external service. All template data are stored in and reused from the database *Templates*.

4 User Interface

In this section we present overall interface and constituents (sub-windows) of our dialogue data preparation platform: the window for dialogue collection, the action template editing window, the knowledge base preparation and management window, the window for intent definition and the window for creation and maintenance of slot filler aliases (see Figure 2)[3].

The user interface for data editing is powered by Python HTTP backend that serves static files and API calls. The backend modifies all four databases directly and uses slotfiller to retrieve slots from user interface. Frontend is written in Javascript and VueJS, and is running inside a web browser.

4.1 Dialogue Collection Window

The central part of the dialogue collection platform is the dialog collection window. The dialogue collection window contains all dialogues submitted by the user. The dialogues could be changed any time:

- by adding or deleting one or several utterances,
- changing text of the utterance and corresponding slot and intent values,
- changing corresponding dialogue act.

To enter a new dialogue user have to push "Add dialogue" button and write an utterance (Figure 3). By pushing button "Extract", entities (slots) in user's utterance are automatically identified by the named entity recognizer, they are extracted and grammatically normalised (see Subsection 3.3), and, if necessary, semantically normalised (for details, see Subsection 4.5). Then user can specify an intent and select an action that needs to be performed by the dialogue system. When the action of the dialogue system is selected, the expected response from the bot is displayed to the user, allowing to check the possible answer and change it, in case a mistake has been identified (for details see Subsection 4.2). The utterance entering process continues until dialogue writing is completed. After pushing "Done" button the dialogue is being add to the *Dialogues* SQLite database.

[3]Although the platform is currently being used to collect dialogue samples in Latvian, in this paper we followed recommendations from reviewers and included prototypical English dialogue samples.

4.2 Action Definition and Editing

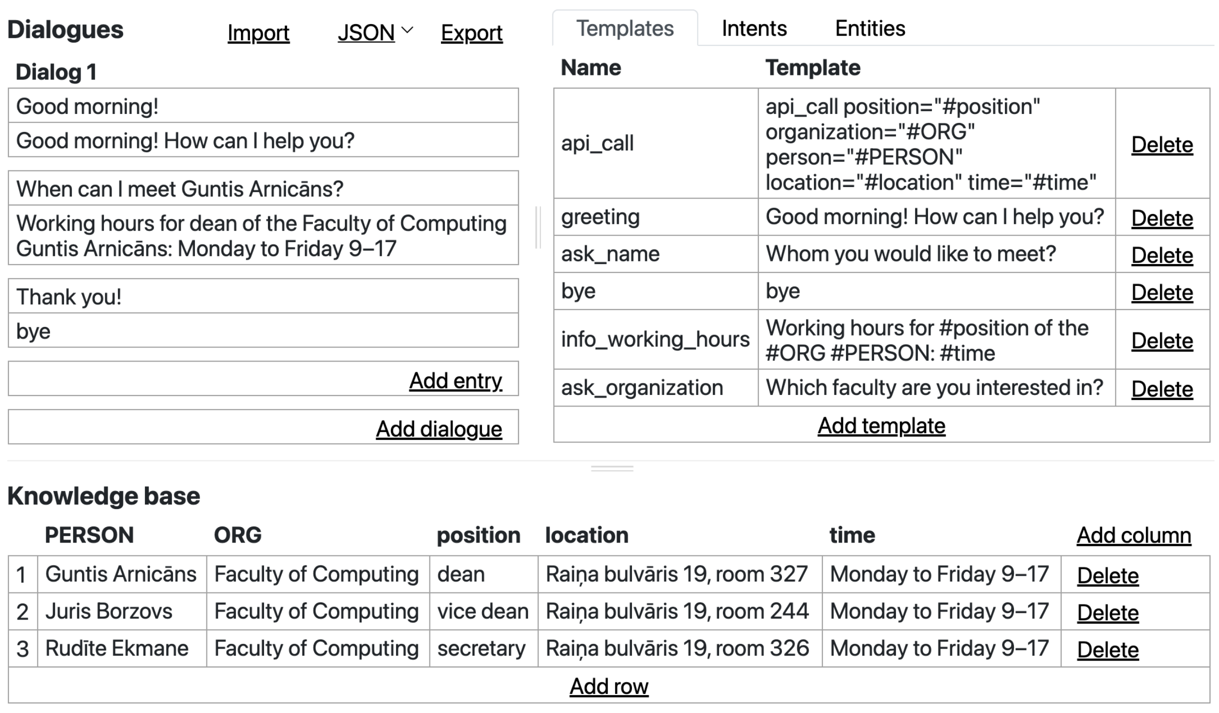
The action template window is used to define the action performed by the dialogue system. For our purposes we have introduced two types of actions: (1) templates of bot answers for particular action and (2) information retrieval request from the knowledge base depending on identified slot values.

In the simplest case system's action is a fixed utterance, specified in action template window. However, in most cases dialogue action and answer depends on previous actions and information gathered from the user. Therefore we introduced mechanisms allowing to generate context dependent grammatically correct slot values identified during the dialogue. The slot values used for answer generation could be from the last utterance or previous ones, as well as from bot's previous answers.

For example, the template for action 'info_working_hours', contains utterance template '*Working hours for #position of the #ORG #PERSON: #time*', where '#position', '#ORG', '#PERSON' and '#time' represent slot values (entities), identified during the dialogue or retrieved from the knowledge base.

Action templates can also include form generation instructions which are very important for fluent output generation in case of inflected languages. For example, in the Latvian template *'Kuras fakultātes position@g darbalaiku Jūs vēlētos noskaidrot?'* (*Which faculty #position@g working hours you would like to know?*) item *#position* represents slot (entity), identified during the dialogue (e.g., *dekāns* (*dean*)), while '@g' requests generation of the genitive form of this entity (e.g., *dekāna* instead of lemma *dekāns*).

When action requires information retrieval from the database (template api_call), the previously extracted information from user's input (slots) is used


Dialogues Import JSON Export

Dialog 1

Good morning!

Good morning! How can I help you?

When can I meet Guntis Arnicāns?

Working hours for dean of the Faculty of Computing Guntis Arnicāns: Monday to Friday 9–17

Thank you!

bye

Add entry

Add dialogue

Templates Intents Entities

Name	Template	
api_call	api_call position="#position" organization="#ORG" person="#PERSON" location="#location" time="#time"	Delete
greeting	Good morning! How can I help you?	Delete
ask_name	Whom you would like to meet?	Delete
bye	bye	Delete
info_working_hours	Working hours for #position of the #ORG #PERSON: #time	Delete
ask_organization	Which faculty are you interested in?	Delete
Add template		

Knowledge base

	PERSON	ORG	position	location	time	Add column
1	Guntis Arnicāns	Faculty of Computing	dean	Raiņa bulvāris 19, room 327	Monday to Friday 9–17	Delete
2	Juris Borzovs	Faculty of Computing	vice dean	Raiņa bulvāris 19, room 244	Monday to Friday 9–17	Delete
3	Rudīte Ekmane	Faculty of Computing	secretary	Raiņa bulvāris 19, room 326	Monday to Friday 9–17	Delete
Add row						

Figure 2: Overview of interface

to form request to the database (see Subsection 4.3). For example, if answer to the previous question is *”Faculty of Computing”*, then query to database will ask for working hours of the dean of the Faculty of Computing.

Similarly to dialogue utterances, actions and their templates could be easily modified during the dialogue writing process, new actions could be added and unnecessary actions removed.

4.3 Knowledge Base Preparation

Goal-oriented dialogue systems often include means for knowledge retrieval from database or any other type of knowledge base. In some cases database already exist, while often creation of knowledge base is part of the dialog system building process. To ensure consistency between dialogues and information in database, the database could be created, filled and modified during dialogue collection process.

The Knowledge base preparation and maintenance window has very simple interface allowing user to enter new entries, change the existing ones or even modify database structure. To ensure consistency between different information pieces of the dialogue system, names of columns in database needs to correspond to the entity types of the dialogue system.

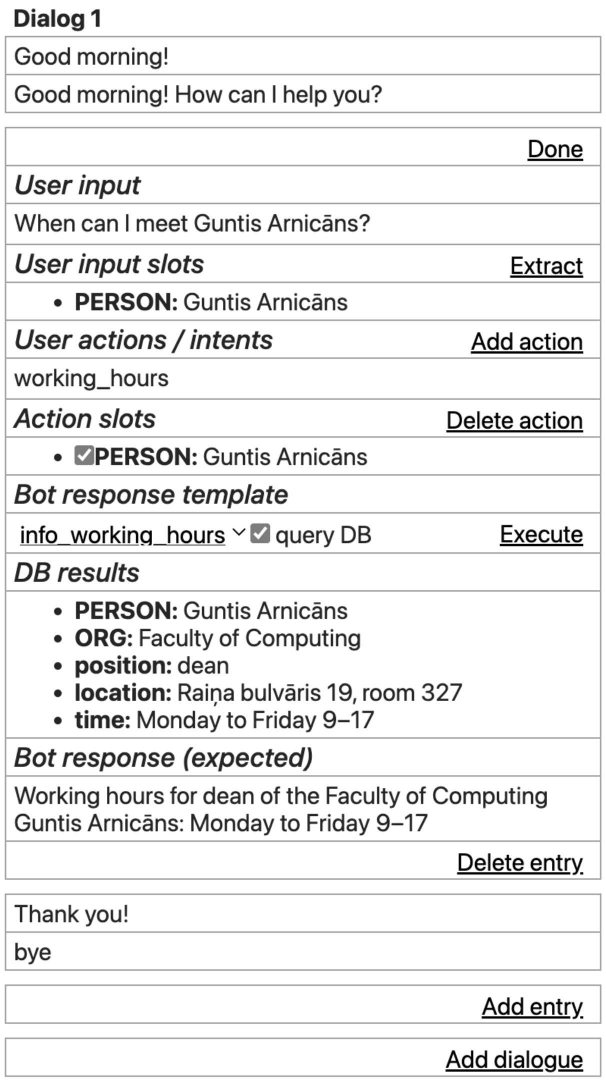
4.4 Intents

For intent management, small and simple window is created, allowing to add, modify and delete intents. Intents defined in this window, are used during dialogue writing process: they can be assigned to each user’s utterance (for details see Subsection 4.1).

4.5 Creation and Maintenance of Slot Filler Aliases

The common problem in dialogue systems that include knowledge retrieval, is incoherence between entity in utterance submitted by user and correct and normalised entity fixed in database. For instance, when dialogue system asks to specify name of particular organization, user can enter its abbreviation (e.g., ”DF” or ”FC” instead of *Faculty of Computing*), commonly used shortened form, use jargon or make spelling errors (e.g., errors in capitalization - *faculty of computing* instead of *Faculty of Computing*). To overcome this bottleneck we introduce entity alias management window, where user can specify official (normalised) form of the entity which has been stored in knowledge base and its typical aliases (see Figure 4).

Similarly to other windows of this platform, the entity editing window allows to add, edit and delete entities and their aliases. Each ”official” entity could have several aliases (synonyms). We also



Dialog 1

Good morning!

Good morning! How can I help you?

Done

User input

When can I meet Guntis Arnicāns?

User input slots Extract

- **PERSON:** Guntis Arnicāns

User actions / intents Add action

working_hours

Action slots Delete action

- ☑**PERSON:** Guntis Arnicāns

Bot response template

info_working_hours ☑ query DB Execute

DB results

- **PERSON:** Guntis Arnicāns
- **ORG:** Faculty of Computing
- **position:** dean
- **location:** Raiņa bulvāris 19, room 327
- **time:** Monday to Friday 9–17

Bot response (expected)

Working hours for dean of the Faculty of Computing Guntis Arnicāns: Monday to Friday 9–17

Delete entry

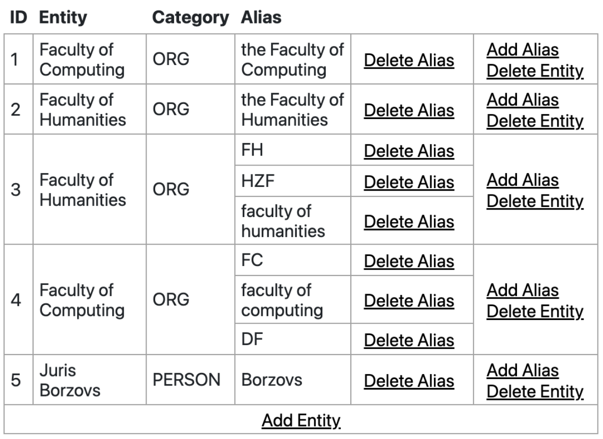
Thank you!

bye

Add entry

Add dialogue

Figure 3: Demonstration of dialogue preparation - utterance writing, slot filling, intent identification and retrieval from the database



ID	Entity	Category	Alias		
1	Faculty of Computing	ORG	the Faculty of Computing	Delete Alias	Add Alias Delete Entity
2	Faculty of Humanities	ORG	the Faculty of Humanities	Delete Alias	Add Alias Delete Entity
3	Faculty of Humanities	ORG	FH	Delete Alias	Add Alias Delete Entity
			HZF	Delete Alias	
			faculty of humanities	Delete Alias	
4	Faculty of Computing	ORG	FC	Delete Alias	Add Alias Delete Entity
			faculty of computing	Delete Alias	
			DF	Delete Alias	
5	Juris Borzovs	PERSON	Borzovs	Delete Alias	Add Alias Delete Entity
			Add Entity		

Figure 4: Window for creation and maintenance of slot filler aliases

keep entity type, in case the same string belongs to several types.

5 Conclusion

In this paper, we have presented a configurable platform for dialogue collection that supports synchronization of information necessary for building a goal-oriented dialogue system, besides specification of dialogue scripts.

The presented platform is publicly available[4]. It has been used for creation of Latvian-specific dataset of dialogues between students and a student service. Following recommendations from reviewers the platform is currently demonstrated on prototypical English dialogue samples, demonstrating its scalability to other languages. We will add a short walkthrough video demonstrating the main features of the platform.

The next development tasks include simple export of dialogue data in commonly used formats to facilitate experiments with various neural dialogue system architectures, and support for a one-click re-training process which currently is implemented as separate background process.

[4]`http://bots.ailab.lv/`

Acknowledgments

This research is funded by the Latvian Council of Science project "Latvian Language Understanding and Generation in Human-Computer Interaction", project No. lzp2018/2-0216.

References

AV Aho and MJ Corasick. 1975. Fast pattern matching: an aid to bibliographic search. *Communications of ACM*, 18(6):333–340.

Jerome R. Bellegarda. 2013. Large-Scale Personal Assistant Technology Deployment: The Siri Experience. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 2029–2033. ISCA.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Normunds Gruzitis, Lauma Pretkalnina, Baiba Saulite, Laura Rituma, Gunta Nespore-Berzkalne, Arturs Znotins, and Peteris Paikens. 2018. Creation of a Balanced State-of-the-Art Multilayer Corpus for NLU. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*, pages 4506–4513.

Samuel Louvan and Bernardo Magnini. 2020. Recent Neural Methods on Slot Filling and Intent Classification for Task-Oriented Dialogue Systems: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 480–496, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Haruna Ogawa, Hitoshi Nishikawa, Takenobu Tokunaga, and Hikaru Yokono. 2020. Gamification platform for collecting task-oriented dialogue data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7084–7093, Marseille, France. European Language Resources Association.

Peteris Paikens, Laura Rituma, and Lauma Pretkalnina. 2013. Morphological analysis with limited resources: Latvian example. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA)*.

Peteris Paikens, Arturs Znotins, and Guntis Barzdins. 2020. Human-in-the-Loop Conversation Agent for Customer Service. In *Natural Language Processing and Information Systems*, volume 12089, pages 277–284. Springer.

Iulian Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. *ArXiv*, abs/1512.05742.

Inguna Skadina and Didzis Gosko. 2020. Towards Hybrid Model for Human-Computer Interaction in Latvian. In *Human Language Technologies - The Baltic Perspective*, volume 328, pages 103 – 110. IOS Press.

Joseph Weizenbaum. 1966. ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine. *Commun. ACM*, 9(1):36–45.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Guangtao Zeng, Wenmian Yang, Zeqian Ju, Yue Yang, Sicheng Wang, Ruisi Zhang, Meng Zhou, Jiaqi Zeng, Xiangyu Dong, Ruoyu Zhang, Hongchao Fang, Penghui Zhu, Shu Chen, and Pengtao Xie. 2020. MedDialog: Large-scale medical dialogue datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9241–9250, Online. Association for Computational Linguistics.

Arturs Znotins and Guntis Barzdins. 2020. LVBERT: Transformer-Based Model for Latvian Language Understanding. In *Human Language Technologies - The Baltic Perspective*, volume 328, pages 111–115. IOS Press.