Discourse-Driven Integrated Dialogue Development Environment for Open-Domain Dialogue Systems

Denis Kuznetsov[◇] **Dmitry Evseev**[◇] **Lidiya Ostyakova**[◇]
Oleg Serikov[◇,‡,♡] **Daniel Kornev**[◇] **Mikhail Burtsev**[◇,‡]

[◇]Moscow Institute of Physics and Technology, Moscow
[‡]AIR Institute, Moscow [♡]Higher School of Economics, Moscow
kuznetsov.dp@phystech.edu, dmitrii.evseev@phystech.edu,
l.ostyakova@yandex.ru, oserikov@hse.ru,
danielko@deeppavlov.ai, burtcev.ms@mipt.ru

Abstract

Development environments for spoken dialogue systems are popular today because they enable rapid creation of the dialogue systems in times when usage of the voice AI Assistants is constantly growing. We describe a graphical Discourse-Driven Integrated Dialogue Development Environment (DD-IDDE) for spoken open-domain dialogue systems. The DD-IDDE allows dialogue architects to interactively define dialogue flows of their skills/chatbots with the aid of the discourse-driven recommendation system, enhance these flows in the Python-based DSL, deploy, and then further improve based on the skills/chatbots usage statistics. We show how these skills/chatbots can be specified through a graphical user interface within the VS Code Extension, and then run on top of the Dialog Flow Framework (DFF). An earlier version of this framework has been adopted in one of the Alexa Prize 4 socialbots while the updated version was specifically designed to power the described DD-IDDE solution.

1 Introduction

Conversational AI focused on the development of the dialogue systems is one of the most challenging areas in the artificial intelligence given the subjectivity of the human language interpretation, and is gaining more and more popularity across both academia (Viewer, 2021) and consumer & enterprise markets (Statista, 2020).

Dialogue systems are classified into task-oriented dialogue systems and chat (or non-task-oriented) dialogue systems. Usually they are studied differently, but combining them has been proposed (Lee et al., 2006), and has been found effective in improving user impressions and the relationships with users (Lucas et al., 2018). While task-oriented systems are more widespread and used in different business scenarios (Bocklisch et al., 2017), building systems that can carry multi-turn open-domain conversations is still far from a solved problem (Hu et al., 2021).

While there is a significant effort in using large generative models (Brown et al., 2020), (Roller et al., 2020) in the open-domain dialogue systems, recent tests of GPT-3 (Marcus and Davis, 2020), and Conversational AI challenges like Alexa Prize Socialbot Grand Challenge (Ram et al., 2018) highlight the inherent weaknesses of these models, namely inability to maintain context, avoid contradictions, and control the dialogue strategically in the long dialogues. These fallacies experienced by us and other teams during the Amazon Alexa Prize 3 (Gabriel et al., 2020) led us to the importance of continuing research towards the tightly-controlled open-domain dialogue management frameworks like E-STDM (Finch and Choi, 2020).

During the Amazon Alexa Prize 4 (Hu et al., 2021) competition we developed (Baymurzina et al., 2021) a first iteration of such framework called Dialog Flow Framework (DFF) (Kuznetsov and Evseev, 2021) to aid in rapid development of open-domain "skills" covering various societal topics in our DREAM Socialbot. Usage showed that while users enjoy controlled conversation in their favorite domains, they often interrupt the conversation with the out-of-domain (Konrád et al., 2021) responses leading it into the non-anticipated directions.

Feedback of our Alexa Prize 4 team members who wrote DFF-based open-domain "skills" shows that predictions of the possible user response could significantly cut the time needed by the developer to design and develop these skills.

The key insight into the conversation is that each utterance in the dialogue is also a kind of action being performed by the speaker (Austin, 1962). These actions are called speech or dialog acts (Jurafsky and Martin, 2009). Yet while dialog acts used by some of the Alexa Prize teams e.g., MIDAS (Yu et al., 2019), are quite sophisticated in understand-

Proceedings of the 2nd Workshop on Computational Approaches to Discourse, pages 29–51
November 10–11, 2021.

ing higher level user intents conceptually similar to those used in task-oriented systems (Bocklisch et al., 2017), they are single turn-based and don't support the development of the open-domain conversations that are interactional and sequential (Eggins and Slade, 1996). Indeed, these dialog act taxonomies lack connection to the discourse level of the casual conversation analysis as shown by (Halliday and M., 2004).

In the effort of making it easier for developers to design and develop sequential and interactional multi-turn open-domain dialogue "skills" we propose a new graphical Discourse-Driven Integrated Dialogue Development Environment (DD-IDDE) that combines a novel Discourse-Driven Recommendation System and a DSL for the Dialog Flow Framework, encapsulated together into an extension to the popular IDE Visual Studio Code.

2 Related Work

One of the first Development Environments designed for spoken language systems, (Denecke, 2000), was created in 2000. While it supported an entire dialogue system development lifecycle, it lacked the power of the modern NLP components such as NER, Entity Linking, etc. as well as the means for the development of the open-dialogue skills/chatbots. Today, there is a variety of dialogue development frameworks that have emerged to aid in rapid dialogue system creation. Rule-based approaches used e.g. in (Bocklisch et al., 2017) are common in the task-oriented systems broadly used in businesses. Other systems incorporate support for full development lifecycle of the dialogue systems starting with individual NLP ML models and ending with the complete chatbots (Burtsev et al., 2018). Some of these systems are also encapsulated in the form of web-based IDEs e.g. (Lawrence et al., 2021).

Domain-specific languages (DSLs) are typically used in dialogue systems to aid in rapid layout of the dialogue logic. One of the oldest DSLs in this space, AIML (Wallace, 2001) was created as an XML dialect that supports pattern matching of the user utterance and template-based entity extraction and was used in development of the famous open-domain dialogue system "A.L.I.C.E." (Wallace, 2009). RASA DSL (Bocklisch et al., 2017) was designed for the development of the task-oriented systems and includes NLU (intent detection and entity extraction) and stories (actions for intents) components. JAICF (AI, 2021), a JavaScript-based state machine-based DSL, was designed to support both task-oriented systems and chatbots.

A winner of Alexa Prize 2, Gunrock team enhanced their socialbot's (Yu et al., 2019) internal Dialog Flow framework with the MIDAS dialog act scheme to create an adaptive and unique conversation experience with the user. However it does not enable bot developers with the mechanisms to manage dialogue strategically across multiple conversation turns. Slugbot (Bowden et al., 2019), another Alexa Prize team, proposed a DRDM dialogue model to control the coherence of the open-domain dialogue using discourse relations. Their approach introduced a combination of dialog acts and four discourse relations from the PDTB 2.0 (Prasad et al., 2008) as means to model interaction within individual turns and at a higher level. Nonetheless, PDTB 2.0 is based on the 1-million-word Wall Street Journal corpus which is a written language and is not best suited for the spoken casual conversation analysis.

DD-IDDE is powered by the Dialog Flow Framework (DFF) derived from the E-STDM framework which is conceptually similar to both Dialog Flow framework described in (Yu et al., 2019) and the DRDM-dialogue model. It's newly developed DSL was specifically created so that: (1) all conditions, responses and even dialogue flows could be implemented as Python functions, and (2) conditions for state transition could use any of the available annotators like Speech Function Classifier and Predictor and others within the larger DREAM socialbot or derived chatbots and assistants. However, while it is relatively common for dialogue flow frameworks to have a built-in DSL and support pluggable NLU components, DD-IDDE specifically incorporates the support for the novel Discourse-Driven Recommendation System as well as the visual tools aiding in designing open-domain dialogues across multiple conversation steps.

3 Concepts

Development of the discourse-driven open-domain dialogue "skills" and chatbots is an ongoing iterative cycle that includes: (1) design of the "skill"/chatbot as the sequence of conditional transitions between states, aided by the discourse moves recommendations, (2) development of the "skill"/chatbot, (3) deployment, ending with the (4) analysis of the running solution, followed by the

restart of the cycle.

Flows, Nodes, Transitions, Conditions and Responses: We split the dialogue graph into the logically related parts called *Flows* to make it more convenient to work with. Each *Flow* represents a dialogue subgraph. Typically, flows can be organized to cover different topics in the dialogue graph. Like in (Finch and Choi, 2020), we identify the states of the dialogue at a certain moment, such states correspond to the *Nodes* of the dialogue graph. Being in a certain node of the graph, we can go to an another node if the *Condition* that corresponds to this *Transition* is true. After each transition, the *Responses* corresponding to the node are returned.

Speech Functions and Discourse Moves: While other solutions often incorporate dialogue/speech acts (DAMSL-SWBD, ISO standard, HCRC etc.) to recognize the higher-level user intentions in a particular context and describe discourse of conversations, they still lack the ability to represent conversational structure and make it impossible to analyze discourse at the dialogue level. In DD-IDDE, we propose implementation of more enhanced intents called Speech Functions that not only represent single turns but also reflect their role in the larger dialogue context.

Eggins and Slade in (Eggins and Slade, 1996) introduced an approach connecting dialogue turns and cross-turn discourse structure patterns specific for casual conversation as the higher-level abstraction. At turn level, they extended Halliday's concept of Speech Functions (Halliday and M., 2004) that express pragmatic goals of speakers and can be used as an enhanced alternative to Dialogue and Speech Acts. Advantages of Speech Functions are in specified grammatical criteria for Speech Function identification and the ability to constrain a comprehensive systematic discourse model of dialogues with the help of them.

These Speech Functions express different discourse purposes and can compound patterns that are expressed by so-called moves. There are three types of Discourse Moves as defined by Eggins and Slade: (i) **Opening** moves are used to start a discussion of a new topic in the dialogue. (ii) **Sustaining** moves develop the current topic of conversation. (iii) **Reacting** moves denote the transfer of the role of the current speaker. The first subtype of reacting moves are **responses** (`React.Response`), which lead dialogues or topics of conversation to completion, the second one is **rejoinders** (`React.Rejoinder`), which, on the contrary, prolong the conversation.

To enable users design dialogue flows with the aid of the Speech Functions and Discourse Moves, we built **Speech Function Classifier** (4.2) and **Speech Function Predictor** (4.3).

DFF DSL: DFF DSL is Python-based language for definition of a dialogue graph, conditions of transitions between nodes (e.g. checks for Speech Functions), and functions for processing user or dialogue system utterances (e.g. entity extraction). All the elements in DFF DSL can be callable (functions), including dialogue graphs, which makes the DSL flexible and extensible.

DFF Extensions: DFF DSL extensions are modules which can be used in the dialog flow for NLU or NLG. NLU extensions include Speech Function Classifier, Speech Function Predictor and Entity Extraction Module. DFF DSL extensions for NLG are Slot Filling Module (fills the dialogue system's response template with extracted entities), Factoid Response Generation Module (provides content of pages from text databases) and Generic Responses Module (returns short utterances such as "Yes" or "No", which represent appropriate Speech Function responses to user utterance's Speech Function).

4 Implementation

4.1 Core

We use **Dialog Flow Framework** (DFF) as a core of the DD-IDDE. Creating conversational agents with the support of the open-domain skills is challenging as was shown, for example, in (Kuratov et al., 2019). Currently, there are many ready-made solutions (Finch and Choi, 2020), (Lison and Kennington, 2016), (Ultes et al., 2017), (Bocklisch et al., 2017).

The alpha version of DFF was developed during the Alexa Prize 4 Challenge, based on E-STDM (Finch and Choi, 2020). After the Challenge, all the shortcomings formulated during the competition were taken into account and the DFF DSL was rewritten. DFF (Kuznetsov and Evseev, 2021) is flexible, expressive, minimal, easy to expand through ready-made extensions or own extensions, as well as collect statistics with subsequent analysis.

Having minimal dependencies, DFF is lightweight and stateless. This allows it to be used

directly as a service in the case of the microservice architecture applications or as a separate API accessible from outside.

DFF has an expressive Python-based DSL called **DFF DSL** specially designed to be a minimalistic and extensible scripting markup language when compared to other solutions (e.g., AIML, RiveScript, ChatScript, botml), which supports using specific extensions (e.g., discourse-driven integrated dialogue development, a mechanism to extract and use entities based on their Wikidata, generic responses based on the speech functions ontology and etc).

There is a set of community-created extentions that are well documented, and it is straightforward to add own extentions to the framework.

4.2 Speech Function Classifier

To aid in the development of the **Speech Function Classifier**, we picked the Santa Barbara Corpus of Spoken American English, which consists of 60 transcriptions of the naturally-occurring spoken conversations. Three face-to-face dialogues were preprocessed and then labeled with the Speech Functions into a small dataset including about 1700 manually annotated utterances. Two annotators reached an inter-annotator agreement of $\kappa = 0.71$ on 1200 utterances which is considered to be a good result. By limiting taxonomy from 45 to 33 classes, using a hierarchical algorithm based on several Logistic Regression models with different parameters, and a rule-based approach, the last version of **Speech Function Classifier** achieved an F1-score varying from 52% to 71% depending on the distribution of the Speech Functions in a particular dialogue. The resulting **Speech Function Classifier** annotator classifies each phrase in the user's utterance as well as each dialogue system's response candidate with Speech Functions. This classification enables the dialogue system to predict the dialogue system's response Speech Functions most expected by the user.

4.3 Speech Function Predictor

Speech Function Predictor yields probabilities of speech functions that can follow a speech function predicted by Speech Function Classifier. **Speech Function Predictor** is a model based on the statistics of speech function sequences in the manually annotated data. As the collected data is not fully representative, rules were added covering missing cases of speech functions sequences. **Speech Function Predictor** annotation to each last phrase of the user's utterances serves as a recommendation for a dialogue system's next response. It is deployed as the HTTP endpoint available for the VS Code extension as the Recommendation API.

4.4 Discourse-Driven Recommendation System

The recommendation system uses Speech Functions used to classify each bot utterance as an input for the Speech Function Predictor to suggest the possible kinds of user responses. The same mechanism is then used to design next bot responses based on the user responses, enabling the rapid design of the sequential discourse-driven open-domain dialogue system.

4.5 Graphical User Interface

The DD-IDDE offers a split GUI (Serikov and Babadeev, 2021) that offers both visual design of the current open-domain skill's/chatbot's dialogue flows and a mechanism for working with the code-behind that represents the logic of the aforementioned skill/chatbot. It also provides a mechanism to obtain recommendations for the discourse moves for the selected node(s).

Visual dialogue graph editor allows users to sketch DFF scenarios in form of the dialogue graph. Here, the nodes of the graph are System States, the transitions are allowed by users' utterances (see figure 4). It is powered by the modified Draw.IO (JGraph Ltd, 2021), (Dieterichs and Rouillé, 2021) editor interface.

Code behind editor is a Python editor, and it shows DFF DSL produced from the dialogue flows designed in the visual editor. It can be extended with the standard Pylint-based recommendations and aids user in fixing errors prior to launching the target skill/chatbot.

This GUI is provided in the form of the VS Code Extension.

4.6 Conversational Analysis

The DD-IDDE enables users to further improve their skills/chatbots based on the analysis of the user behavior. A DFF Extension was made that allows collecting the particular dialogue flows and individual nodes usage statistics at run-time as well as showing statistics charts in a dashboard.

4.7 General Statistical Recommendation System

The aforementioned DFF Extension for Conversation Analysis enables DFF open-domain "skill" developers to use the collected statistics of frequencies of transitions between nodes to focus developer's attention on the user behavior.

5 Evaluation

DFF DSL vs pure Python: DFF DSL helps to implement scenario skills more easily. For example, Coronavirus skill in DFF DSL format takes about 250 lines, while in pure Python format - about 650 lines. Coronavirus skill in DFF DSL has a clear structure (21 states and 27 transitions), in pure Python the skill has 19 functions and the main function which handles the scenario contains 64 if/else statements, where some of the if/else conditions are nested (making the code harder for maintenance).

Developing With and Without Discourse-Driven Recommendations: Development of DFF DSL-based open-domain "skill" is a multi-step cycle that includes initial skill design, development, test, deployment, feedback gathering, until skill quality is improved. The most challenging parts of the skill design are: (1) prediction of the out-of-domain responses that can be not so obvious for developers, and (2) sequential dialogue design. Our initial experiments with the DD-IDDE and its Discourse Moves recommendations showed that time required to address both the out-of-domain responses and to design sequential dialogue was significantly cut compared to the pure DFF DSL skills design.

Conclusions and Future Directions

In this paper, we introduced a new Discourse-Driven Integrated Dialogue Development Environment (DD-IDDE) focused on aiding users in building open-domain skills/chatbots based on the discourse-driven recommendations. Our DD-IDDE makes it possible for novice users to rapidly design their scenario-driven skills in the visual editor with the discourse move recommendations, and then expand their skills with the custom Python code. In the future versions we plan to further expand the DD-IDDE to enable seamless transitions between the manually edited DFF DSL and its visual graph representation, add deeper integration of the statistics into the main interface of the VS Code extension, as well as add other extentions to the underlying DFF framework.

Acknowledgements

Authors are deeply grateful to the Alexa Prize organizers for their feedback and advice during the Alexa Prize Grand Challenge 4. Authors also thanks our contributor Dmitry Babadeev who became an owner of the VS Code extension encapsulating the DD-IDDE, and all members of Neural Networks and Deep Learning Lab for their support and advice during the project development. Oleg Serikov is partially supported by the framework of the HSE University Basic Research Program.

References

Just AI. 2021. Jaicf. https://github.com/just-ai/jaicf-kotlin/wiki.

John Langshaw Austin. 1962. *How to do things with words*. William James Lectures. Oxford University Press.

Dilyara Baymurzina, Denis Kuznetsov, Dmitry Evseev, Dmitry Karpov, Alsu Sagirova, Anton Peganov, Fedor Ignatov, Elena Ermakova, Daniil Cherniavskii, Sergey Kumeyko, Oleg Serikov, Yury Kuratov, Lidiya Ostyakova, Daniel Kornev, and Mikhail Burtsev. 2021. Dream technical report for the alexa prize 4. *4th Proceedings of Alexa Prize*.

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management.

Kevin K. Bowden, JiaQi Wu, Wen Cui, Juraj Juraska, Vrindavan Harrison, Brian Schwarzmann, Nick Santer, and Marilyn A. Walker. 2019. Slugbot: Developing a computational model and framework of a novel dialogue genre. *CoRR*, abs/1907.10658.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, et al.

2018. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127.

Matthias Denecke. 2000. An integrated development environment for spoken dialogue systems. In *Proceedings of the COLING-2000 Workshop on Using Toolsets and Architectures To Build NLP Systems*, pages 51–60, Centre Universitaire, Luxembourg. International Committee on Computational Linguistics.

Henning Dieterichs and Vincent Rouillé. 2021. Draw.io vs code integration. https://github.com/hediet/vscode-drawio.

S. Eggins and D. Slade. 1996. Analysing casual conversation.

James D. Finch and Jinho D. Choi. 2020. Emora stdm: A versatile framework for innovative dialogue system development.

Raefer Gabriel, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, Shui Hu, Karthik Gopalakrishnan, Seokhwan Kim, Lauren Stubel, Kate Bland, Arindam Mandal, and Dilek Z. Hakkani-Tür. 2020. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge.

M. A. K. Halliday and Christian Matthiessen M. I. M. 2004. *An introduction to functional grammar / M.A.K. Halliday*, 3rd ed. / rev. by christian m.i.m. matthiessen. edition. Hodder Arnold London.

Shui Hu, Yang Liu, Anna Gottardi, Behnam Hedayatnia, Anju Khatri, Anjali Chadha, Qinlang Chen, Pankaj Rajan, Ali Binici, Varun Somani, Yao Lu, Prerna Dwivedi, Lucy Hu, Hangjie Shi, Sattvik Sahai, Mihail Eric, Karthik Gopalakrishnan, Seokhwan Kim, Spandana Gella, Alexandros Papangelis, Patrick Lange, Di Jin Nicole Chartier, Mahdi Namazifar, Aishwarya Padmakumar, Sarik Ghazarian, Shereen Oraby, Anjali Narayan-Chen, Yuheng Du, Lauren Stubell, Savanna Stiff, Kate Bland, Arindam Mandal, Reza Ghanadan, and Dilek Hakkani-Tur. 2021. Further advances in open domain dialog systems in the fourth alexa prize socialbot grand challenge. https://d7qzviu3xw2xc.cloudfront.net/alexa/alexaprize/docs/sgc4/Alexa-Prize-Technical-Paper-2021_Final.pdf.

JGraph Ltd. 2021. diagrams.net. https://github.com/jgraph/drawio.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Jakub Konrád, Jan Pichl, Petr Marek, Petr Lorenc, Van Duy Ta, and Ondrej Kobza. 2021. Alquist 4.0: Towards social intelligence using generative models and dialogue personalization. *4th Proceedings of Alexa Prize*.

Yuri Kuratov, Idris Yusupov, Dilyara Baymurzina, Denis Kuznetsov, Daniil Cherniavskii, Alexander Dmitrievskiy, Elena Ermakova, Fedor Ignatov, Dmitry Karpov, Daniel Kornev, et al. 2019. Dream technical report for the alexa prize 2019. *3rd Proceedings of Alexa Prize*.

Denis Kuznetsov and Dmitry Evseev. 2021. Dialog flow framework @ deepmipt github. https://github.com/deepmipt/dialog_flow_framework/.

Andrew Lawrence, Braden Ream, Michael Hood, and Tyler Han. 2021. Visual skill building. https://www.voiceflow.com/.

Cheongjae Lee, Sangkeun Jung, Minwoo Jeong, and Gary Geunbae Lee. 2006. Chat and goal-oriented dialog together: a unified example-based architecture for multi-domain dialog management. In *2006 IEEE Spoken Language Technology Workshop*, pages 194–197.

Pierre Lison and Casey Kennington. 2016. Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of ACL-2016 system demonstrations*, pages 67–72.

Gale M. Lucas, Jill Boberg, David Traum, Ron Artstein, Jonathan Gratch, Alesia Gainer, Emmanuel Johnson, Anton Leuski, and Mikio Nakano. 2018. Getting to know each other: The role of social dialogue in recovery from errors in social robots. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, page 344–351, New York, NY, USA. Association for Computing Machinery.

Gary Marcus and Ernest Davis. 2020. Experiments testing gpt-3's ability at commonsense reasoning: results. https://cs.nyu.edu/~davise/papers/GPT3CompleteTests.html.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational ai: The science behind the alexa prize.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M. Smith, Y-Lan Boureau, and Jason Weston. 2020. Recipes for building an open-domain chatbot.

Oleg Serikov and Dmitry Babadeev. 2021. Dd-idde alpha version 0.2.1. https://github.com/deepmipt/vscode-dff/releases/tag/0.2.1.

Statista. 2020. Number of digital voice assistants in use worldwide from 2019 to 2024 (in billions)*. https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/.

Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, et al. 2017. Pydial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78.

Google Books Ngram Viewer. 2021. Chatbot, natural language processing popularity. https://books.google.com/ngrams/graph?content=chatbot%2C+natural+language+processing&year_start=2000.

Richard Wallace. 2001. Artificial intelligence markup language (aiml). https://web.archive.org/web/20060505035935/http://aitools.org/aiml/spec/.

Richard S. Wallace. 2009. *The Anatomy of A.L.I.C.E.*, pages 181–210. Springer Netherlands, Dordrecht.

Dian Yu, Michelle Cohn, Yi Mang Yang, Chun-Yen Chen, Weiming Wen, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, Giritheja Sreenivasulu, Sam Davidson, Ashwin Bhandare, and Zhou Yu. 2019. Gunrock: A social bot for complex and engaging long conversations.

A User Interaction

A.1 Designing

Having installed the DD-IDDE VS Code Extension [1], user creates a new .drawio file in the `common/dff_markup_scenarios` with the name of the target skill scenario, e.g., `pet_scenario.drawio` - see 1.

User then proceeds with defining the Speech Function of the starting node by double-clicking on it - see 2.

Once the first node has been set up, user can use the *Speech Function Recommendations* to pick the next discourse moves by clicking on the "Show Suggestions" menu item - see 3.

User then continues adding nodes based on their scenario, with the aid of the Speech Function recommendations - see 4.

A.2 Developing

Once the designed skill is sufficiently outlined, user proceeds with the use of the Show Markup functionality that generates the code based on the designed skill - see 5.

With the generated DFF DSL code, user can now add the additional code-behind functionality such as the use of the regular expressions, custom intents, and the like - see 6.

A.3 Analyzing

In the Conversational Analysis dashboard user can observe the frequency of nodes' visits of the individual dialogue flows, rates of transitions within dialogues, as well as the frequency of the user responses. User can also collect statistics of annotations of user and bot utterances, custom data coming from other DFF extentions, e.g., classification of user utterances with the speech functions, user and bot emotions etc. The dashboard is described in more detail in appendix D. The dashboard allows users to select as many dialogues as one dialog and show user behavior on it. On this dialog graph is shown for one dialog 12 and on this one for many 8.

A.4 VS Code Extension Graphical User Interface Illustrations

VS Code Extension GUI is shown in figures 1, 2, 3, 4, 5, 6

[1] `https://github.com/deepmipt/vscode-dff/releases/tag/0.2.1`

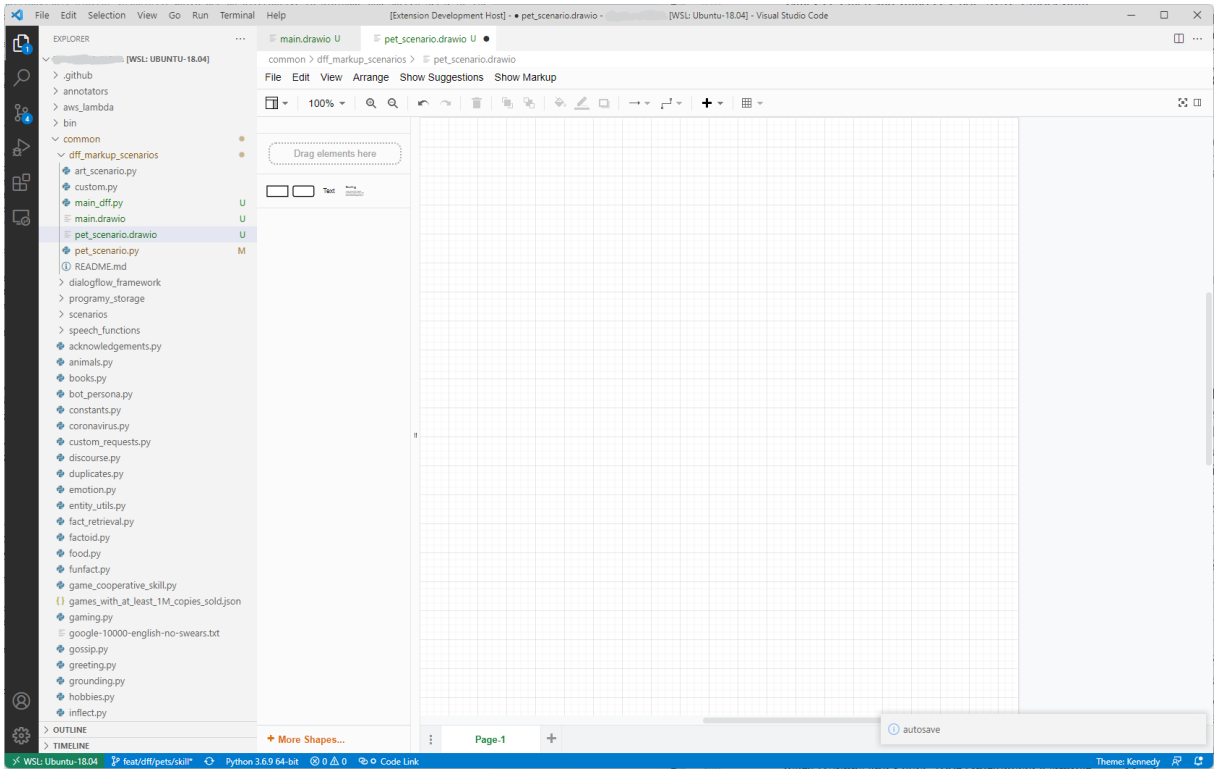


Figure 1: New .drawio file created in VS Code

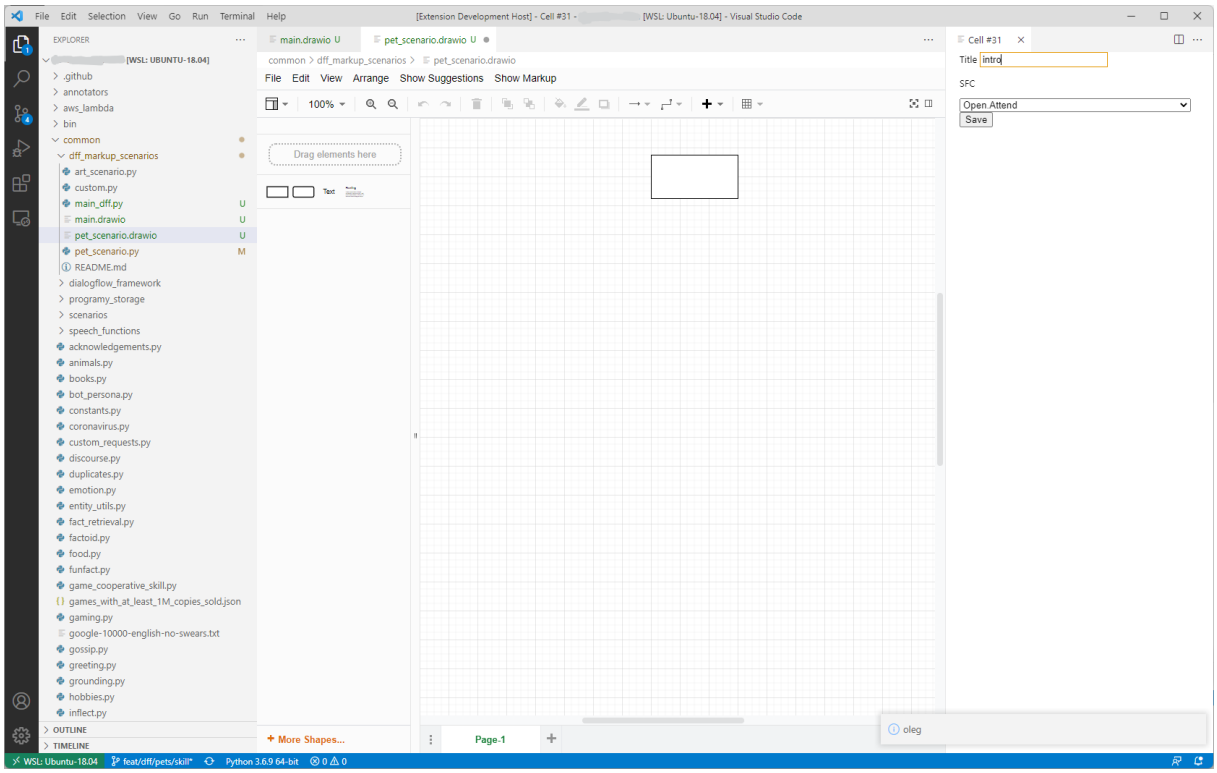


Figure 2: Setting Speech Function for the current node

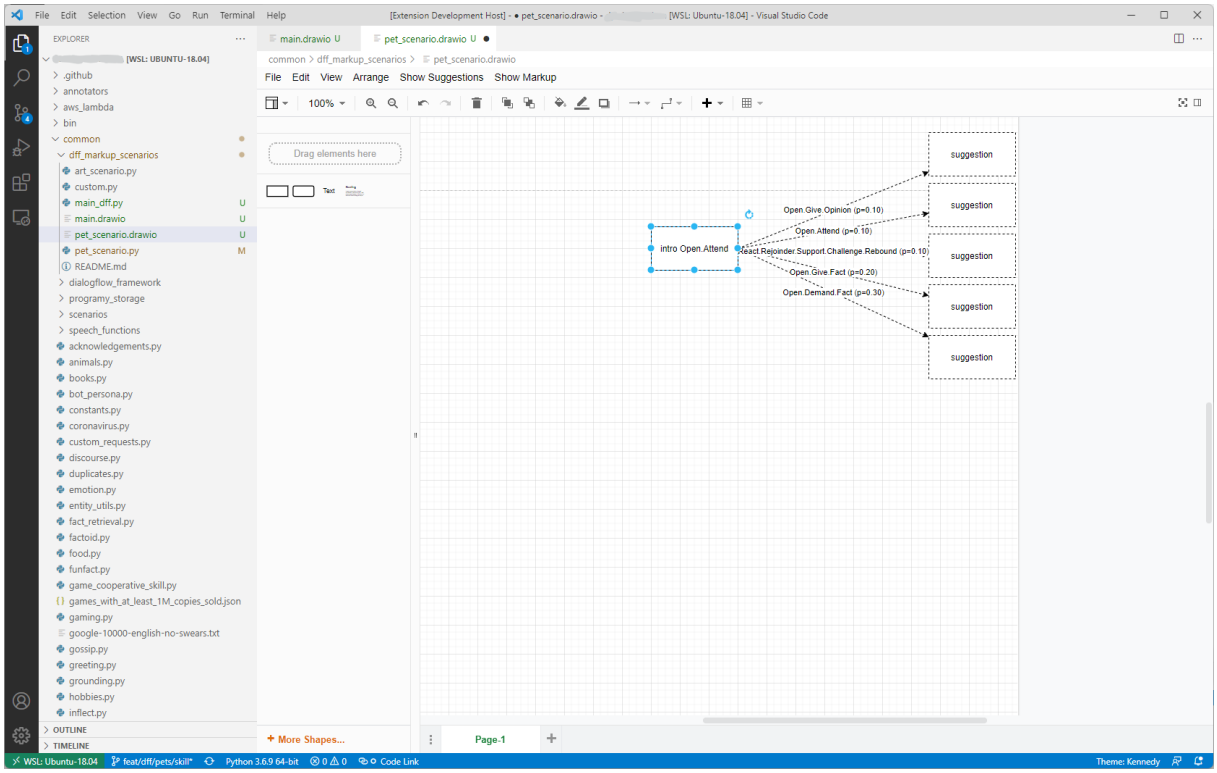


Figure 3: Using the *Speech Function Recommendations*

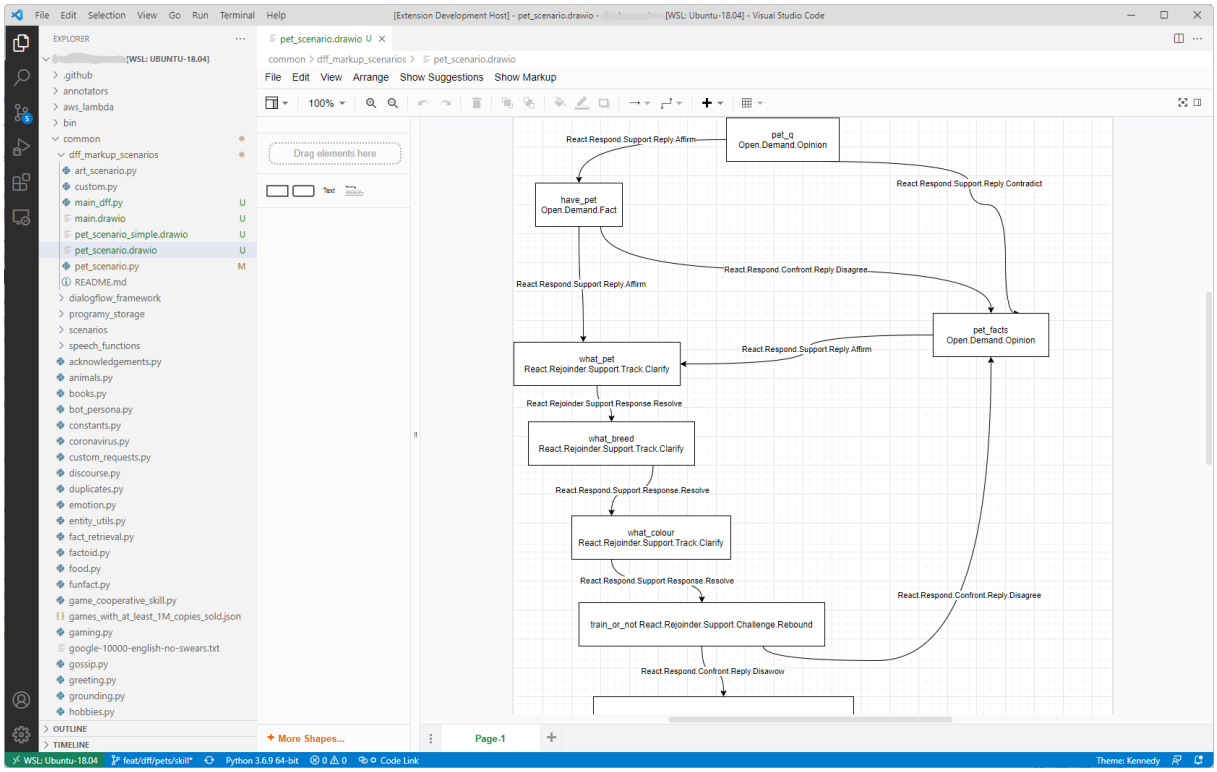


Figure 4: Outlining Scenario In VS Code Extension

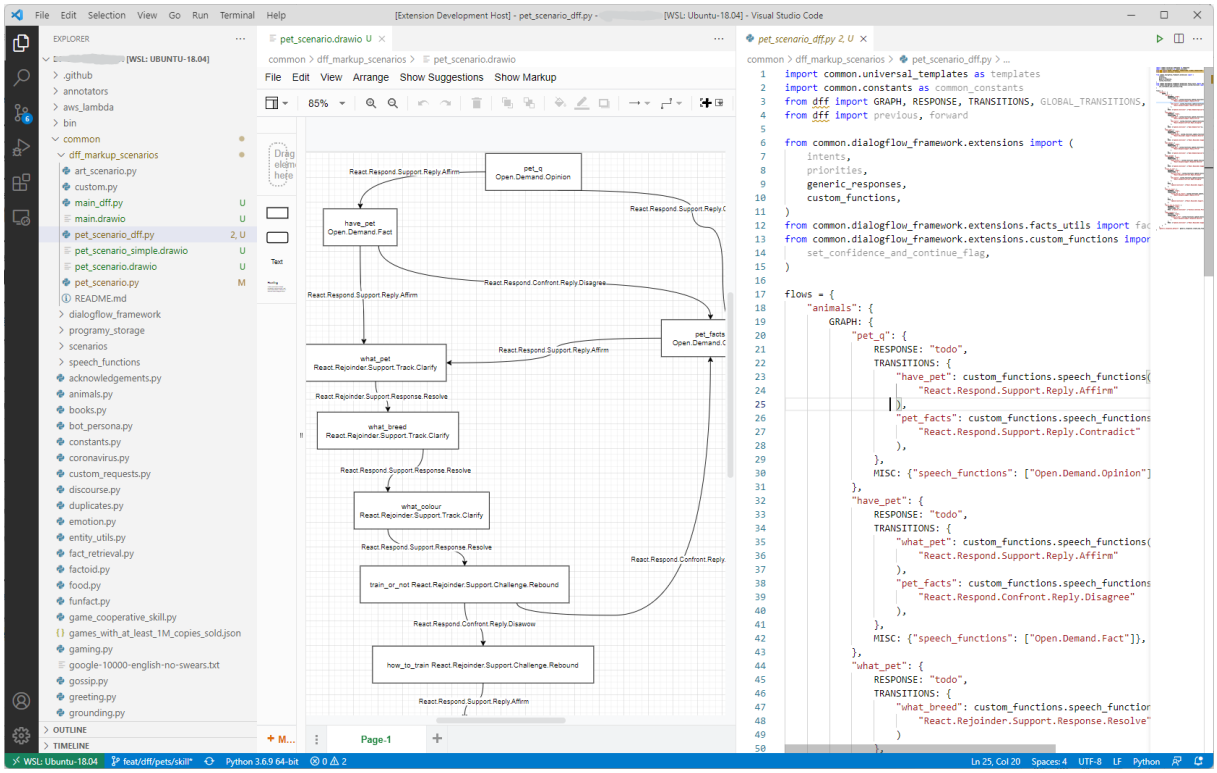


Figure 5: Using the Show Markup functionality to generate DFF Markup Code

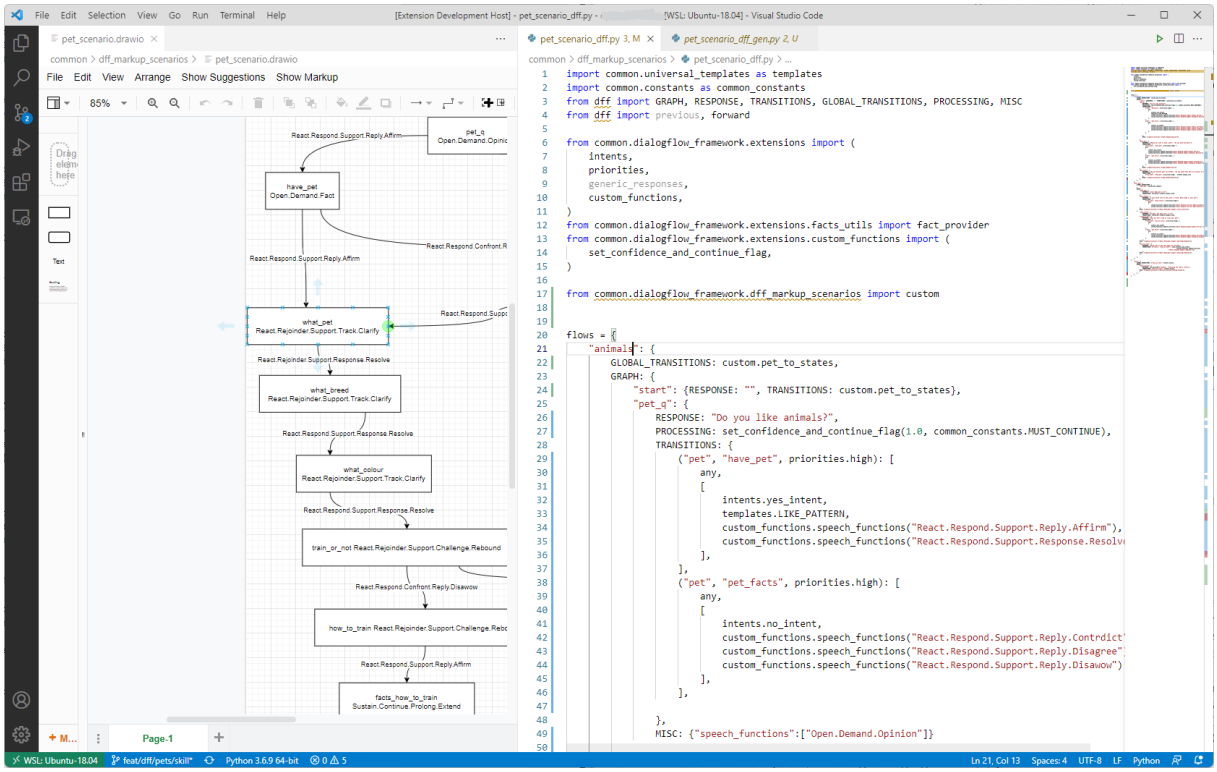


Figure 6: Adding Custom Code to the DFF Markup File

B DFF: Detailed Overview

B.1 Description

DFF as well as many extentions for it are distributed with an open source license, like many other frameworks (e.g., Emora STDM, AIML, RiveScript, ChatScript, botml, OpenDial, PyDial). DFF is distributed with Apache 2.0 license.

DFF DSL is a Python-based DSL for designing scenario-driven skills/chatbots. The main element of a skill in DFF DSL format is a dialogflow (a set of nodes and transitions between them, which refers to a particular subtopic). A dialogflow is a Python dictionary where keys are node names and elements describe response in this node and transitions to other nodes in the same dialogflow or in other dialogflows (Fig. 13). The main features of a dialogflow in DFF DSL format:

- a condition for transition between nodes can be one of predefined conditions in DFF DSL (for example, a regular expression or a function which checks the speech function in the user utterance, etc.)
- a response can be a string or a function
- processing of user utterance and response with custom functions (which can be used, for example, for slot extraction and slot filling (Fig. 14))
- extensions for using facts databases (for example, Wikipedia) for response generation (Fig. 16) or generic responses to user utterance speech function

The solution, like many others (e.g., AIML), is based on the state machine. However, unlike others, DFF aids in the development of the open-domain skills/chatbots with the help of the discourse moves recommendation making it easy to design these skills and chatbots. It also incorporates a mechanism to extract and use entities based on their Wikidata types, as well as to provide generic responses based on the speech functions ontology.

DFF was designed to be extensible as a modular system. With the external extentions, DFF can be significantly expanded as in complex frameworks (Finch and Choi, 2020), (Bocklisch et al., 2017). There is a set of community-created extentions that are well documented, and it is straightforward to add own extentions to the framework.

DFF DSL is automatically validated before starting the bot, making it easy to identify issues beforehand..

B.2 Transitions

The target node for transition can be:

- the node name from the current dialogflow;
- the name of another dialogflow and the node in this dialogflow;
- special keywords (“forward” - to the next node in the current dialogflow, “back” - to the previous node, “repeat” - to make cycle transition, “previous” - to the previous active node).

Transitions between nodes can be local (which can be used only in the dialogflow where they were defined) or global (can be used in any other dialogflow).

C DFF: Extensions

The approach of dividing DFF into modules allows users to incorporate both lightweight annotators, classifiers, and so on, as well as modules based on neural models that require a large amount of resources in their solutions.

In production ready systems, it is necessary to use keywords and a matching pattern for the agent to work, as well as the selection of entities, the presence of ontologies, the identification of intents, the classification of sentiments, and much more.

C.1 Generic Responses Module

Generic responses are short utterances, generated by the bot in response to a particular speech function in user utterance. Generic responses can be useful if the user is proactive and responds with long and detailed utterances, the bot in this case will say somewhat like “yes”, “I agree”. Examples of generic responses:

- “No” in response to "React.Respond.Confront.Reply.Contradict" and "React.Respond.Reply.Contradict" speech functions;
- “I don’t know.” to "React.Respond.Confront.Reply.Disawow" and "React.Respond.Reply.Disawow";
- “I don’t agree with you” to "React.Respond.Confront.Reply.Disagree" and "React.Respond.Reply.Disagree";

- “Yes” to "React.Respond.Support.Reply.Affirm", "React.Respond.Support.Reply.Agree" and "React.Respond.Reply.Agree".

Generic responses are implemented as an extention for DFF DSL with one cycled node of dialogflow and a response function which extracts a speech function from the user utterance and generates a suitable generic response.

C.2 Other Extentions

Entity Extraction Module is used to find entities of particular types in the user utterance and store them in shared memory. Entities can be extracted by:

- **Wikidata Entity Type**: user can specify a Wikidata ID for the given entity type to find entities of this type in the given utterance, e.g. to find a city user should specify "wiki:Q515" (which corresponds to “city” in Wikipedia)
- **Entity Tag** from the set of ("PERSON", "LOCATION", "ORGANIZATION", etc.): user can specify one of these tags to find entities of this type, e.g., to find a location user should specify "tag:LOCATION"

Slot Filling Module is used to fill extracted entities in the slots of the response.

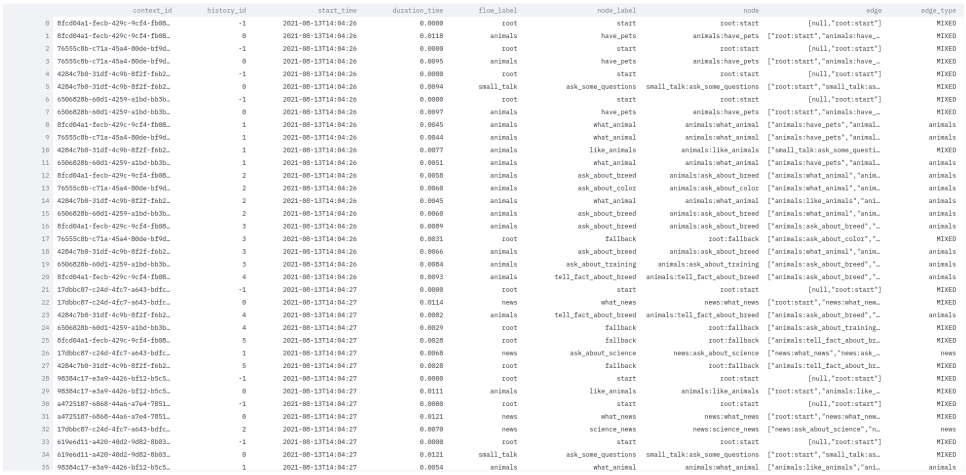
Factoid Response Generation Module provides content of the page from the databases like Wikipedia, wikiHow, etc. mentioned as the argument of the extension function.

Generic Responses Module returns short utterances (for example, “Yes”, “No”, “I agree” etc.) that represent selected speech functions as appropriate response to the selected user utterance’s speech functions.

D DFF: Conversational Analysis Dashboard

The statistics DFF extention [2] comes with a dashboard that displays various statistics. Using them, users can evaluate the user's use of script branches. The dashboard is shown in figures 7, 8, 9, 10, 12

[2] https://github.com/kudep/dff-node-stats/



	context_id	history_id	start_time	duration_time	flow_label	node_label	node	edge	edge_type
0	8fcd04a1-fecb-429c-9cf4-fb08…	-1	2021-08-13T14:04:26	0.0000	root	start	root:start	[null,"root:start"]	MIXED
1	8fcd04a1-fecb-429c-9cf4-fb08…	0	2021-08-13T14:04:26	0.0118	animals	have_pets	animals:have_pets	["root:start","animals:have_…	MIXED
2	76555c8b-c71a-45a4-80de-bf9d…	-1	2021-08-13T14:04:26	0.0000	root	start	root:start	[null,"root:start"]	MIXED
3	76555c8b-c71a-45a4-80de-bf9d…	0	2021-08-13T14:04:26	0.0095	animals	have_pets	animals:have_pets	["root:start","animals:have_…	MIXED
4	4284c7b0-31df-4c9b-8f2f-feb2…	-1	2021-08-13T14:04:26	0.0000	root	start	root:start	[null,"root:start"]	MIXED
5	4284c7b0-31df-4c9b-8f2f-feb2…	0	2021-08-13T14:04:26	0.0094	small_talk	ask_some_questions	small_talk:ask_some_questions	["root:start","small_talk:as…	MIXED
6	6506828b-60d1-4259-a1bd-bb3b…	-1	2021-08-13T14:04:26	0.0000	root	start	root:start	[null,"root:start"]	MIXED
7	6506828b-60d1-4259-a1bd-bb3b…	0	2021-08-13T14:04:26	0.0097	animals	have_pets	animals:have_pets	["root:start","animals:have_…	MIXED
8	8fcd04a1-fecb-429c-9cf4-fb08…	1	2021-08-13T14:04:26	0.0045	animals	what_animal	animals:what_animal	["animals:have_pets","animal…	animals
9	76555c8b-c71a-45a4-80de-bf9d…	1	2021-08-13T14:04:26	0.0044	animals	what_animal	animals:what_animal	["animals:have_pets","animal…	animals
10	4284c7b0-31df-4c9b-8f2f-feb2…	1	2021-08-13T14:04:26	0.0077	animals	like_animals	animals:like_animals	["small_talk:ask_some_questi…	MIXED
11	6506828b-60d1-4259-a1bd-bb3b…	1	2021-08-13T14:04:26	0.0051	animals	what_animal	animals:what_animal	["animals:have_pets","animal…	animals
12	8fcd04a1-fecb-429c-9cf4-fb08…	2	2021-08-13T14:04:26	0.0058	animals	ask_about_breed	animals:ask_about_breed	["animals:what_animal","anim…	animals
13	76555c8b-c71a-45a4-80de-bf9d…	2	2021-08-13T14:04:26	0.0060	animals	ask_about_color	animals:ask_about_color	["animals:what_animal","anim…	animals
14	4284c7b0-31df-4c9b-8f2f-feb2…	2	2021-08-13T14:04:26	0.0045	animals	what_animal	animals:what_animal	["animals:like_animals","ani…	animals
15	6506828b-60d1-4259-a1bd-bb3b…	2	2021-08-13T14:04:26	0.0060	animals	ask_about_breed	animals:ask_about_breed	["animals:what_animal","anim…	animals
16	8fcd04a1-fecb-429c-9cf4-fb08…	3	2021-08-13T14:04:26	0.0089	animals	ask_about_breed	animals:ask_about_breed	["animals:ask_about_breed","…	animals
17	76555c8b-c71a-45a4-80de-bf9d…	3	2021-08-13T14:04:26	0.0031	root	fallback	root:fallback	["animals:ask_about_color","…	MIXED
18	4284c7b0-31df-4c9b-8f2f-feb2…	3	2021-08-13T14:04:26	0.0066	animals	ask_about_breed	animals:ask_about_breed	["animals:what_animal","anim…	animals
19	6506828b-60d1-4259-a1bd-bb3b…	3	2021-08-13T14:04:26	0.0054	animals	ask_about_training	animals:ask_about_training	["animals:ask_about_breed","…	animals
20	8fcd04a1-fecb-429c-9cf4-fb08…	4	2021-08-13T14:04:26	0.0093	animals	tell_fact_about_breed	animals:tell_fact_about_breed	["animals:ask_about_breed","…	animals
21	17dbbc87-c24d-4fc7-a643-bdfc…	-1	2021-08-13T14:04:27	0.0000	root	start	root:start	[null,"root:start"]	MIXED
22	17dbbc87-c24d-4fc7-a643-bdfc…	0	2021-08-13T14:04:27	0.0114	news	what_news	news:what_news	["root:start","news:what_new…	MIXED
23	4284c7b0-31df-4c9b-8f2f-feb2…	4	2021-08-13T14:04:27	0.0082	animals	tell_fact_about_breed	animals:tell_fact_about_breed	["animals:ask_about_breed","…	animals
24	6506828b-60d1-4259-a1bd-bb3b…	4	2021-08-13T14:04:27	0.0029	root	fallback	root:fallback	["animals:ask_about_training…	MIXED
25	8fcd04a1-fecb-429c-9cf4-fb08…	5	2021-08-13T14:04:27	0.0028	root	fallback	root:fallback	["animals:tell_fact_about_br…	MIXED
26	17dbbc87-c24d-4fc7-a643-bdfc…	1	2021-08-13T14:04:27	0.0068	news	ask_about_science	news:ask_about_science	["news:what_news","news:ask_…	news
27	4284c7b0-31df-4c9b-8f2f-feb2…	5	2021-08-13T14:04:27	0.0028	root	fallback	root:fallback	["animals:tell_fact_about_br…	MIXED
28	9836dc17-e3a9-442e-bf12-b5c5…	-1	2021-08-13T14:04:27	0.0000	root	start	root:start	[null,"root:start"]	MIXED
29	9836dc17-e3a9-442e-bf12-b5c5…	0	2021-08-13T14:04:27	0.0111	animals	like_animals	animals:like_animals	["root:start","animals:like_…	MIXED
30	a4725187-6868-44a6-a7e4-7851…	-1	2021-08-13T14:04:27	0.0000	root	start	root:start	[null,"root:start"]	MIXED
31	a4725187-6868-44a6-a7e4-7851…	0	2021-08-13T14:04:27	0.0121	news	what_news	news:what_news	["root:start","news:what_new…	MIXED
32	17dbbc87-c24d-4fc7-a643-bdfc…	2	2021-08-13T14:04:27	0.0070	news	science_news	news:science_news	["news:ask_about_science","n…	news
33	619e6d11-a420-48d2-9d82-8b03…	-1	2021-08-13T14:04:27	0.0000	root	start	root:start	[null,"root:start"]	MIXED
34	619e6d11-a420-48d2-9d82-8b03…	0	2021-08-13T14:04:27	0.0121	small_talk	ask_some_questions	small_talk:ask_some_questions	["root:start","small_talk:as…	MIXED
35	9836dc17-e3a9-442e-bf12-b5c5…	1	2021-08-13T14:04:27	0.0054	animals	what_animal	animals:what_animal	["animals:like_animals","ani…	animals

Figure 7: Statistic data frame is shown in the Dashboard

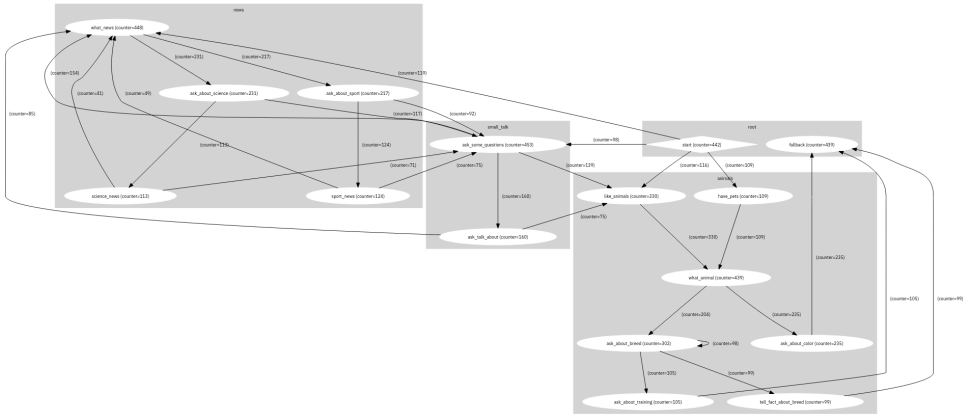


Figure 8: Transition graph with counters is built for all dialogues. Counters show numbers of users hitting these nodes or edges.

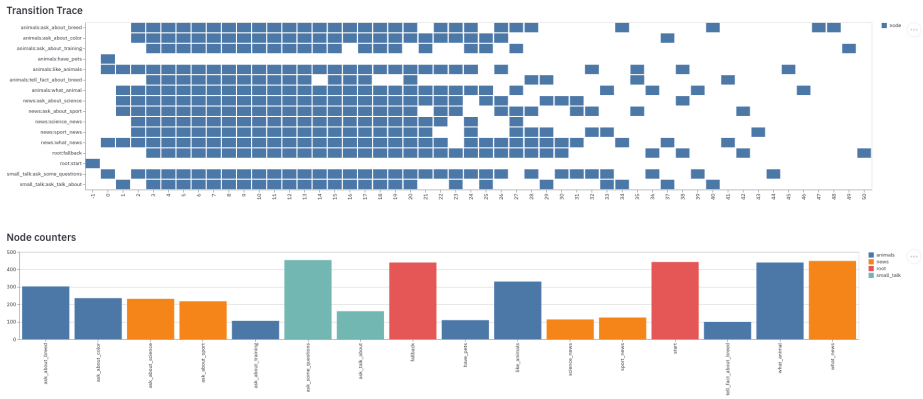


Figure 9: These figures are built in the dashboard. The top figure shows nodes that will be sent by all users at a certain step of the dialogue. The abscissa axis shows the order of a step in the dialog, the ordinate axis shows the node in which the user is located for this step. The bottom figure shows the frequency of users hitting each node, different colors are shown in different flows in which the nodes

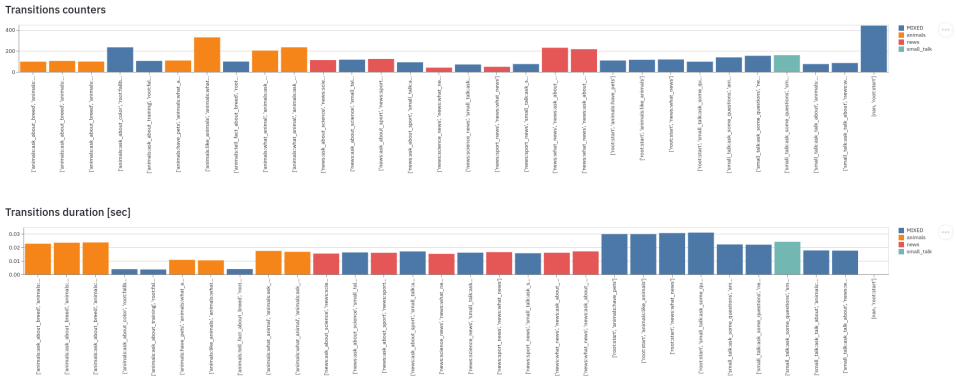


Figure 10: Statistic of transitions in dashboard. The top figure shows the frequency of users hitting each transition between nodes. The bottom figure shows duration of processing time for each transition between nodes. Different colors are shown in different flows in which the nodes.

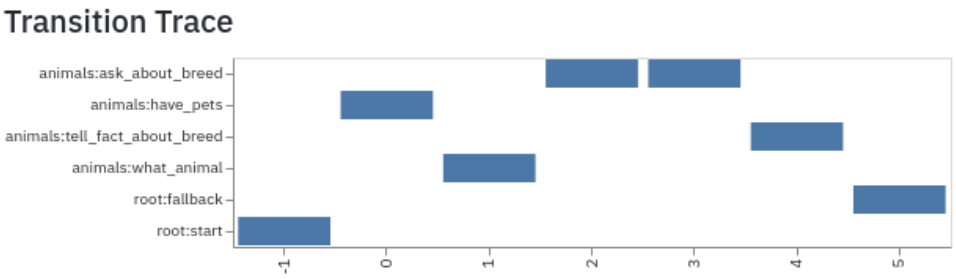


Figure 11: Statistic of transition in dashboard for one dialog. This figure shows the user transitions from node to node. The abscissa axis shows the order of a step in the dialog, the ordinate axis shows the node in which the user is located for this step.

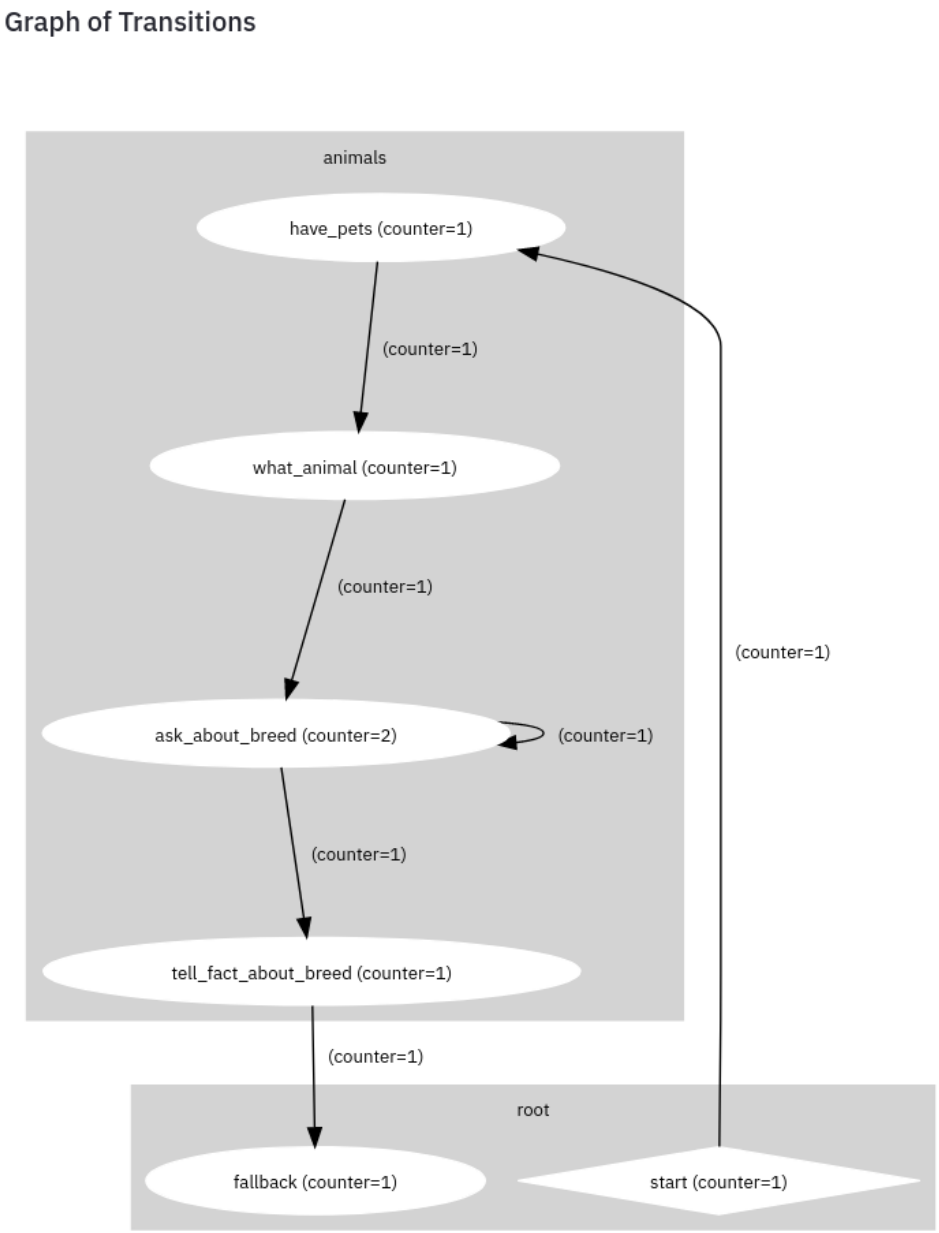


Figure 12: This figure is built in the dashboard. The graph shows user transitions through the script graph using one dialog as an example. Counters show numbers of users hitting these nodes or edges.

E Additional Code Examples

This section of the appendix contains the code and tables with descriptions

```
"drawing": {
    GLOBAL_TRANSITIONS: {
        "what_painter": custom_functions.drawing_request,
    },
    GRAPH: {
        "what_painter": {
            RESPONSE: "Pictures of what painters do you like?",
            TRANSITIONS: {forward(): intents.always_true}
        },
        "what_paintings": {
            RESPONSE: "I also like pictures of {user_fav_painter}. "
                      "What kind of paintings do you like to draw: "
                      "landscapes, portraits or something else?",
            PROCESSING: [custom_functions.entities(user_fav_painter="wiki:Q1028181"),
                         custom_functions.slot_filling],
            TRANSITIONS: {forward(): intents.always_true}
        },
        "how_to_draw": {
            RESPONSE: custom_functions.how_to_draw_response,
            TRANSITIONS: {
                ("facts", "how_to_draw"): intents.yes_intent
            }
        }
    },
},
```

Figure 13: Example of dialog flow in DFF DSL format

```
"what_paintings": {
    RESPONSE: "I also like pictures of {user_fav_painter}. "
              "What kind of paintings do you like to draw: "
              "landscapes, portraits or something else?",
    PROCESSING: [custom_functions.entities(user_fav_painter="wiki:Q1028181"),
                 custom_functions.slot_filling],
    TRANSITIONS: {forward(): intents.always_true}
},
```

Figure 14: DFF DSL elements for entity extraction from user utterance and slot filling in response

```
PROCESSING: [custom_functions.entities(user_fav_painter="tag:LOCATION"),
             custom_functions.slot_filling],
```

Figure 15: DFF DSL elements for entity extraction using Entity Detection annotator

```
"facts": {
    GLOBAL_TRANSITIONS: {"how_to_draw": intents.facts},
    GRAPH: {"how_to_draw": {
              RESPONSE: fact_provider("wikiHow", "Improve-Your-Drawing-Skills"),
              TRANSITIONS: {"how_to_draw": intents.facts}}
    }
},
```

Figure 16: Dialog Flow with "fact_provider" - a function for response generation using a wikiHow article

```
generic_responses flow = {
    GLOBAL_TRANSITIONS: {
        generic_response_state: sys_response_to_speech_function_request,
    },
    GRAPH: {
        "generic_response": {
            RESPONSE: usr_response_to_speech_function_response,
            TRANSITIONS: {repeat(): sys_response_to_speech_function_request}
        }
    }
}
```

Figure 17: Dialogflow for generic responses generation

```
if quarantine_end(last_utterance):
    logging.info('Quarantine end detected')
    reply, confidence = QUARANTINE_END_PHRASE, 0.95
elif last_bot_phrase in [WORK_AND_STAY_HOME_PHRASE, CDC_STAY_HOME_RECOMMENDATION]:
    if ' why ' in last_utterance_text or last_utterance_text[:3] == 'why':
        reply, confidence = EXPLAIN_PHRASE, 0.95
elif dontlike(last_utterance):
    reply, confidence = '', 0
elif asked_have(last_utterance):
    reply, confidence = BOT_CORONAVIRUS_PHRASE, 0.95
    reply = improve_phrase(reply)
elif vaccine_safety_request(last_utterance):
    reply, confidence = VACCINE_SAFETY_PHRASE, 0.95
elif emotion_detected(last_utterance, 'fear') or emotion_detected(last_utterance, 'anger'):
    r = random()
    if r < 0.5:
        reply, confidence = FEAR_HATE_REPLY1, 0.95
    else:
        reply, confidence = FEAR_HATE_REPLY2, 0.95
    reply = improve_phrase(reply)
elif get_chance_request(last_utterance):
    reply, confidence = "As I am not your family doctor, " \
                        "my knowledge about your resilience to coronavirus is limited.", 0.95
    reply = f'{reply} Please, check the CDC website for more information.'  # Daniil suggestion
elif 'to learn more' in last_bot_phrase:
    fear_prob = get_emotions(dialog['utterances'][-1], probs=True).get('fear', 0)
    logging.debug(f'Fear prob {fear_prob}')
    if is_no(last_utterance):
        logging.info('Another fact request detected, answer is NO')
        reply = corona_switch_skill_reply()
        confidence = 0.98
        human_attr["coronavirus_skill"]['stop'] = True
    elif fear_prob > 0.9:
        logging.info('Corona fear detected')
        reply = 'Just stay home, wash your hands and you will be fine. We will get over it.'
        confidence = 0.95
    elif is_yes(last_utterance):
        logging.info('Returning a fact')
        reply, confidence = return_fact(self.facts,
                                        human_attr['coronavirus_skill']['used_phrases'],
                                        asked_about_age, met_last), 1
    else:
        reply, confidence = '', 0
```

Figure 18: Coronavirus skill in pure Python

```
GRAPH: {
    "quarantine_end": {
        RESPONSE: "Although most American states are easing the restrictions, "
        "the Coronavirus pandemics in the majority of the states hasn't been reached yet. "
        "If you want to help ending it faster, please continue social distancing as much as you can.",
        PROCESSING: [int_prs.set_confidence(DEFAULT_CONFIDENCE)],
    },
    "uninteresting_topic": {RESPONSE: "", PROCESSING: [int_prs.set_confidence(ZERO_CONFIDENCE)]},
    "bot_has_covid": {
        RESPONSE: "As a socialbot, I don't have coronavirus. I hope you won't have it either.",
        PROCESSING: [int_prs.set_confidence(DEFAULT_CONFIDENCE)]
        # offer_more should be here by original idea, but it's useless due default function arguments
        # in legacy version of code (see coronavirus_skill.scenario: 554 and 375)
    },
    "vaccine_safety": {
        RESPONSE: "All CDC-approved vaccines are safe enough for you - "
        "of course, if your doctor does not mind against using them. "
        "I can't say the same about getting infected, however, "
        "so vaccines are necessary to prevent people from that..",
        PROCESSING: [int_prs.set_confidence(DEFAULT_CONFIDENCE)],
    },
    "user_feel_emotion": {
        RESPONSE: rsp.choice(
            [
                "Please, calm down. We are a strong nation, we are vaccinating people "
                "and we will overcome this disease one day.",
                "Please, chin up. We have already defeated a hell lot of diseases, "
                "and I am sure that coronavirus will be the next one.",
            ]
        ),
        PROCESSING: [int_prs.set_confidence(USER_FEEL_EMOTION_CONFIDENCE)],
    },
    "user_resilience_to_covid": {
        RESPONSE: "As I am not your family doctor, "
        "my knowledge about your resilience to coronavirus is limited. "
        "Please, check the CDC website for more information.",
        PROCESSING: [int_prs.set_confidence(DEFAULT_CONFIDENCE)],
    },
    "covid_symptoms": {
        RESPONSE: "According to the CDC website, "
        "The main warning signs of coronavirus are: "
        "difficulty breathing or shortness of breath, "
        "persistent pain or pressure in the chest, "
        "new confusion or inability to arouse, "
        "bluish lips or face. If you develop any of these signs, "
```

Figure 19: Coronavirus skill DFF DSL format

Speaker	Utterance	Speech Function
Bot	Do you like animals?	Open.Demand.Opinion
User	yeah	React.Respond.Support.Reply.Affirm
Bot	Would you like to have a pet?	Open.Demand.Opinion
User	yeah	React.Respond.Support.Reply.Affirm
Bot	Cool! What pet is it?	React.Rejoinder.Support.Track.Clarify
User	I have a dog. His name is Archie.	React.Respond. Confront.Reply.Resolve
Bot	I also would like to have such a friend. What breed is your pet?	React.Rejoinder.Support.Track.Clarify
User	He is a golden retriever.	React.Respond. Confront.Reply.Resolve
Bot	Alright. And what color is it?	React.Rejoinder.Support.Track.Clarify
User	He's golden one with white ears.	React.Respond. Confront.Reply.Resolve
Bot	Have you ever tried to train your pet?	React.Rejoinder. Support.Challenge.Rebound
User	Yeah, I train him every week in the park.	React.Respond.Support.Reply.Affirm
Bot	Would you like to know how dogs are trained?	React.Rejoinder. Support.Challenge.Rebound
User	okay	React.Respond.Support.Reply.Affirm
Bot	Some breeds are more independent and hard to train than others.	Sustain.Continue.Prolong.Extend
User	yeah	React.Respond.Support.Register

Table 1: Dialog Example

Speech Function	Discourse purpose
Open.Attend	attention seeking
Open.Demand.Fact	demand factual information
Open.Demand.Opinion	demand evaluative information
Open.Give.Fact	give factual information
Open.Give.Opinion	give evaluative information
Sustain.Continue.Prolong.Extend	offer additional or contrasting information
Sustain.Continue.Prolong.Elaborate	clarify and restate
Sustain.Continue.Prolong.Enhance	qualify previous move by giving details
Sustain.Continue.Monitor	check that audience is still engaged
React.Rejoinder.Confront.Challenge.Counter	dismiss addressee's right to his/her position
React.Rejoinder.Confront.Response.Re-challenge	question relevance of a prior move
React.Rejoinder.Support.Challenge.Rebound	dismiss addressee's right to his/her position
React.Rejoinder.Support.Response.Resolve	provide clarification
React.Rejoinder.Support.Track.Check	elicit repetition of a misheard element
React.Rejoinder.Support.Track.Clarify	verify information heard
React.Rejoinder.Support.Track.Confirm	confirm information heard
React.Rejoinder.Support.Track.Probe	volunteer further details
React.Respond.Confront.Disengage	show unwillingness to interact
React.Respond.Confront.Reply.Contradict	negate prior information
React.Respond.Confront.Reply.Disagree	provide negative respond to question
React.Respond.Confront.Reply.Disawow	deny acknowledgement of information
React.Respond.Support.Develop.Elaborate	clarify and restate a prior move
React.Respond.Support.Develop.Enhance	qualify previous move by giving details
React.Respond.Support.Develop.Extend	offer additional or contrasting information
React.Respond.Support.Engage	show willingness to interact
React.Respond.Support.Register	display attention to the speaker
React.Respond.Support.Reply.Acknowledge	indicate knowledge of information given
React.Respond.Support.Reply.Affirm	provide positive response to the question
React.Respond.Support.Develop.Enhance	deny acknowledgement of information
React.Respond.Support.Reply.Agree	indicate support of information given

Table 2: Discourse Purposes of Speech Functions