DREAM Technical Report for the Alexa Prize 4

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, Mikhail Burtsev

Neural Networks and Deep Learning Lab
Moscow Institute of Physics and Technology
`dilyara.baymurzina@phystech.edu`, `kuznetsov.dp@phystech.edu`,
`burtcev.ms@mipt.ru`

Abstract

In this report, we present the DREAM 2 Socialbot design and share scientific and technology contributions made towards developing a fluent and meaningful socialbot for Alexa Prize 4. Building on top of the last year's solution we added a rich plethora of the script-driven skills created with the help of the novel Dialogue Flow Framework. To lay down the foundation for the discourse-driven dialogue strategy management we introduced tag-based Response Selector and Speech Functions Classifier. We also began working on User and Bot Persona Knowledge Graphs as well as incorporated our work on World Knowledge Graph alongside with Entity Linking. The final version of DREAM 2 Socialbot is still a hybrid system that combines rule-based, deep learning, and knowledge based driven components, but it moves closer to a goal-aware system that can recognize users' and own goals and drive the dialogue strategically.

1 Introduction

In recent years, the field of conversational AI experienced rapid progress driven by the application of deep neural networks. On the one hand, transfer learning with pre-trained masked language models significantly improved natural language understanding [6, 28, 43] and made much better intent and entity recognition possible. On the other hand, the success of the end-to-end generative models in the machine translation had not been replicated yet for the open domain dialogue, despite considerable efforts to increase the size of the models and datasets [45, 33, 36, 2]. As a result, the state-of-the-art open-domain conversational systems such as XiaoIce [46] or Alexa Prize socialbots combine machine learning models for user input understanding with the hand-written script and template-based response generators [11].

Current open-domain conversational agents proved to be quite successful in the user engagement over the pre-scripted segments of the dialogue [11]. When a user allows such a system to drive dialogue, it creates a smooth flow by suggesting available relevant topical components and presenting them. This dialogue is like browsing over the prerecorded videos being translated on the TV channels or YouTube. In this case, the primary factor that contributes to user satisfaction is the quality of the precooked content. Conversely, when the user proactively interacts with the system, it usually fails to meet expectations. These failures are generally due to (1) insufficient information about the user; (2) lack of commonsense and dialogue state understanding; (3) absence of scripts for the requested domain.

4th Proceedings of Alexa Prize (Alexa Prize 2020).

In DREAM 2 – the next version of our original socialbot [23], we try to address the following challenges: user's preferences modeling, goal-aware dialogue management, and domain scaling. Our goal is to go beyond mere infotainment towards an engaging and thoughtful conversational partner.

2 DREAM 2 Socialbot System Design and Architecture

DREAM 2 socialbot is implemented and served with DeepPavlov Library[1] and DeepPavlov Agent[2].

DeepPavlov Library [3] includes a number of predefined pipelines for the most common NLP tasks. Any pipeline can be easily run in the REST API mode, making it a good choice for micro-service architecture.

DeepPavlov Agent is a framework for building production-ready multi-skill virtual assistants, complex dialogue systems, and chatbots. Key features of DeepPavlov Agent include (1) scalability and reliability in the high load environment due to micro-service architecture; (2) ease of adding and orchestrating conversational skills; (3) shared dialogue state memory and NLP annotations accessible to all skills. DeepPavlov Agent orchestrates the following types of services:

- `Annotator` is a service for NLP preprocessing of an utterance. It can implement some basic text processing like spelling correction, named entity recognition, etc.;
- `Skill` is a service producing a conversational response candidate for a current dialogue state;
- `Skill Selector` is a service that selects a subset of the available skills for producing candidate responses;
- `Response Selector` is a service that picks the best response out of the available candidates to be sent to the user;
- `Postprocessor` is a service that is responsible for the postprocessing of the response utterance. It can make some basic things like adding a user name, inserting emojis, etc.
- `Dialogue State` stores current dialogues between users and a conversational agent as well as annotations and other meta-data serialized in JSON format. The state supports sharing of stored information across the services.

DREAM processes user input in three main steps: (1) input annotation and context retrieval, (2) response generation, and (3) response selection (see Figure 1). First, multiple `Annotators` preprocess the user input serving a natural language understanding task. Also, annotators retrieve contextual information from external sources such as Wikipedia or news. Then, `Skill Selector` runs a subset of `Skills` based on the extracted information. Finally, `Response Selector` picks a response to be sent to `Response Annotators` and, eventually, to the user. All elements of the pipeline are running asynchronously with two points of synchronization: `Skill Selector` and `Response Selector`. Communication between different services goes through a shared memory stored in `Dialogue State`.

The architecture of DREAM socialbot and a list of all used components can be found in Figure 1. Most of the services are described in [23]. Detailed description of *new and changed components* of DREAM 2 can be found in Section D of the Appendix.

2.1 Dialogue Management

Dialogue management in the module-based DREAM architecture is driven by two main components: `Skill Selector` choosing the set of skills that generate response candidates for the current context, and `Response Selector` choosing the final reply which is given to the user. `Skill Selector` of the DREAM socialbot is described in detail in the previous year's DREAM Technical Report [23], and this year it has been modified to support the handling of the sensitive mode cases. Now, this mode is only used when users ask personal questions on restricted topics. This mode is not, however, involved for processing user utterances with obscene language.

[1] `https://deeppavlov.ai`
[2] `https://github.com/deepmipt/dp-agent`

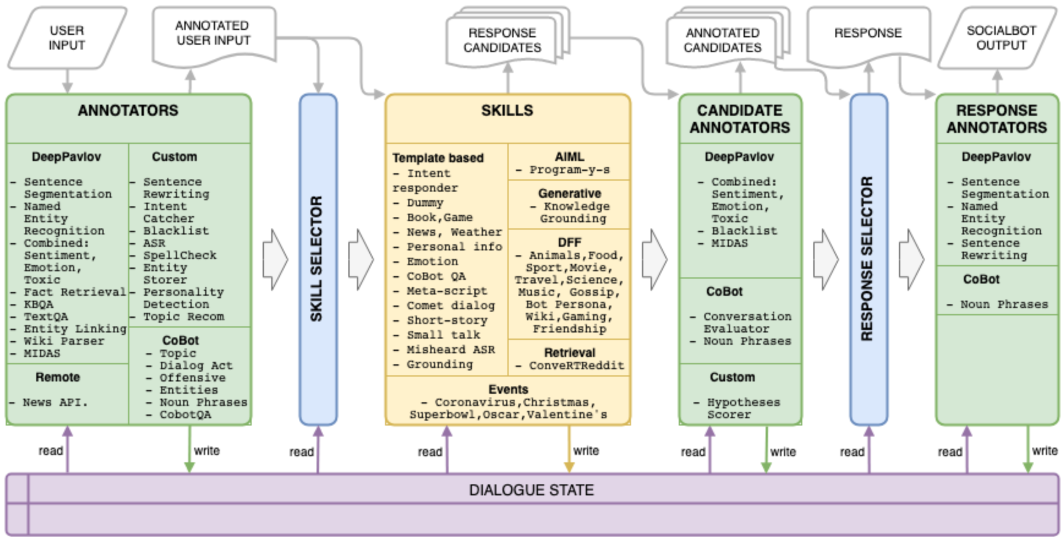

Figure 1: **DREAM socialbot architecture.** Multiple `Annotators` are used to extract information from the user input. `Skill Selector` defines a subset of active `Skills` based on the extracted information. Selected `Skills` propose their response candidates. Finally, `Response Selector` picks a response to be sent to `Response Annotators` and, eventually, to the user. All elements of the pipeline are running asynchronously with two points of synchronization: `Skill Selector` and `Response Selector`. `Dialogue State` serves as a shared memory.

Generally, all skills chosen by the `Skill Selector` propose response candidates. Then it is the task for the `Response Selector` to choose the responses that are most suitable to the current dialogue context. Taking into the account success of the script-based approach [10] and our focus on goal-aware dialogue management, we built a number of script-based skills on popular topics to provide users with the tightly-controlled conversational user experience. Some of these skills were described in [23], while updated and new skills are presented in Appendix D. One of the skills – `Wiki skill` is able to conduct the dialogue on the wide list of the popular topics that are not covered by specific scripted skills. `Response Selector` is also responsible for ensuring a smooth transition to the next topic by utilizing a number of different linking techniques.

Although some of the scripts can be connected by specific questions which smoothly move user from one topic to another one (for example, at the end of the discussion of animals, the socialbot may proceed with the question about animal movies user likes), it is important to provide some smoothness and coherence in the dialogue for all possible transitions between topics. DREAM 1[23] presented *linking questions approach* which was further developed for the current competition. For every topic covered by the appropriate script-based skill, there was created a curated list of questions designed to direct the dialogue towards the aforementioned skill, so called *linking questions*. There is a special component providing a linking question to the topic predicted by `Topic Recommendation` annotator (details in Section 3.7) on every turn. However, abusing this linking question technique can give a negative impression of uncontrolled switching between topics. Thus, for almost all supported pairs of topics we have a variety of citations, interesting facts, or thoughts that are related to both topics simultaneously. In case there is no accompanying connecting phrase that connects both current and new topic, we offer standalone introductions to that topic instead.

2.1.1 Response Candidate Annotations

The modular architecture of DREAM socialbot implies the combination of many different skills, including template-based, retrieval, and generative skills. The key ability of the socialbot to deeply discuss the most popular topics is reflected in the prioritization of components that can conduct a specific dialogue about those topics. However, response confidences alone can not justify their selection. Therefore, we propose two more tags for each response candidate: (1) continuation flag, (2) response parts. Both tags are assigned by the skill proposing it, so there could be response candidates with the same tags or tags with none of the response candidates assigned to it.

Continuation flag is intended to reflect the ability of the skill to continue the conversation on the next turn after the proposed response candidate, and it can have one of four different values:

- `must continue` – the current response candidate is perfectly suited to the context and should be returned to the user;
- `can continue script` – the current response candidate is a part of the script in progress, but there are no exact matches fitting the context, the response candidate should be returned to the user if no other response candidates with the exact match to the context exist (`must continue`);
- `can continue prompt` – the current response candidate is a prompt to start the conversation about a specific topic, and the skill itself is able to continue the dialogue over the next steps if the user keeps the conversation going;
- `can not continue` – the current response candidate is from the non-scripted skill or is the final response node in the script.

Continuation flag is conceived to prioritize skills with scripted conversations. There is no guaranteed method to provide an entirely coherent dialogue, although scripted skills could give an impression of coherency for at least a few turns. All linking questions are assigned to `can continue prompt` tag, and all skills except the script-based ones are annotated as `can not continue`. As for the scripted skills, we use the following rules to set the *continuation flag*:

- script beginning:
  - if user requests dialogue on a specific topic,
    - and trigger patterns or entities of specific types are extracted, then set `must continue`;
    - and a specific topic was detected, then set `can continue prompt`;
  - else if user was asked linking question leading to this skill,
    - and trigger patterns or entities of specific types are extracted from the user utterance, or expected yes/no intent detected, then set `must continue`;
    - otherwise, set `can continue prompt`;
  - otherwise,
    - if user mentions trigger patterns or entities of specific types, set `can continue prompt`;
    - otherwise, if specific topics were detected by an annotator, set `can continue prompt`;
- if in the middle of the script:
  - and expected patterns are in the user utterance, then set `must continue`;
  - and user was asked yes/no question, and yes/no intents detected, then set `must continue`;
  - and expected and found trigger patterns and entity types, then set `must continue`;
  - and for all other cases, set `can continue script`;
- end of the script:
  - if previous independent part of the script was finished, and the current line asks a question which can involve user into conversation with this skill again (for example, discussion of one specific movie is finished, and the skill asks one more question about movies), then set `can continue prompt`;
  - otherwise, set `can not continue`;

Response parts for each response candidate is a list indicating which response parts are present in the response candidate. This tag is needed to enact `Response Selector`'s understanding whether a response candidate already contains a phrase for further development of the dialogue, or that the candidate acknowledges what user have just said. This information is important for joining several response candidates to provide smoothness and coherence. Available response parts are:

- `acknowledgement` – statement intended to confirm socialbot's understanding of the user utterance;

- `body` – main part of the response intended to support a current conversation topic;
- `prompt` – statement or question which is designed to start a conversation on a new or user-requested topic.

Special template-based skill generates `acknowledgement` for some dialogue acts, other response candidates are considered as `body`, while linking questions are assigned to the `prompt`. We have a set of the hand-written heuristics to determine whether to combine `prompt` and `acknowledgement` with the best response candidate labeled as `body`.

2.1.2 Response Selection

The DREAM socialbot' Response Selector employs the following logic:

1. filters inappropriate response candidates;
2. penalizes response candidates for repetitions;
3. computes the final single-value score for each response candidate. The value is computed by the empirical formula combining confidences and CoBot Conversation Evaluation model predictions;
4. applies hand-written heuristics to prioritize special cases;
5. selects the final response candidate as one with the highest score.

The most important problem in this scheme is its heavy reliance on the confidences provided by the skills of different origin. For example, AIML skills have the same confidences for all responses, while rule-based skills have confidences manually assigned by developers for different cases, and finally retrieval skills return similarity scores as confidences. Another common problem faced by the system is latency and consequent timeouts of the remote services that lead to failures in the conversation evaluation process.

We figured out that last year's approach to response selection was mainly reactive and mostly relied on confidences, dialogue act classification, and intent classification of the last user's utterance. Thus, the system lacks a high-level understanding of the user's goals in the dialogue and fails to establish a solid common ground. Starting with the assumption that the user has some high-level goals in a dialogue, we developed and implemented basic goal-aware dialogue management in DREAM 2 as well as established a foundation for more advanced goal-aware dialogue management in the future versions. As trainable ranking models can not guarantee system's adherence to the user goals, we decided to use tag-based `Response Selector`. Generally, all response candidates based on the annotations and tags assigned by skills are divided into priority groups, and the final response candidate is chosen within the group with the highest priority as one with the highest score from the trainable ranking model. The single value score originally was computed using the empirical formula from Alexa Prize Challenge 3 but later replaced by trainable hypotheses (response candidates) ranking model described in Section 3.8.

All response candidates are also annotated with entities, topics, dialogue acts, intents. `Response Selector` has the following priorities:

1. if **special user intents** detected (for example, request to use an inaccessible Alexa command), **choose** the special component providing responses to those requests;
2. if user wants to **switch topic** to anything else or wants to stop discussing a given topic or wants the socialbot to switch to a new topic, **give** priority to linking questions (prompts) if available, otherwise to all available prompts;
3. if user wants to talk about some **particular topic**, **give** priority to prompts which include entities intersecting with requested by a user or which have `must continue` flag, otherwise to all available prompts;
4. if user's dialogue act requires specific action from the socialbot, **give** priority to response candidates containing at least one of the requested dialogue acts if available, otherwise to the next item;
5. otherwise **give** priority to response candidates in the following order:
   (a) with `must continue` flag;

(b) with `can continue script` or `can continue prompt` flags, and entities mentioned by a user;
(c) with `can not continue` flag, and entities mentioned by a user;
(d) with `can continue script` or `can continue prompt` flags, and entities not mentioned by a user;
(e) with `can not continue`, and entities not mentioned by a user.

To summarize, we give priority to scripted skills, while still having an opportunity to interrupt the script if the user requests something else. In the absence of response candidates provided by the scripted skills, we choose the final response using trainable `Hypotheses Scorer` (see Section 3.8), and push conversation to scripts by attaching linking questions according to the hand-written rules.

3 Selected Science and Technology Contributions

3.1 Dialogue Flow Framework

The `Dialogue Flow Framework` (`DFF`) is a dialogue systems development environment that supports both rapid prototyping and long-term team development workflows for dialogue systems. This framework is based on Emora STDM (E-STDM) [10]. A simple structure allows easily building and visualizing a dialogue graph, see Figure 7 in Appendix E.

`DFF` was designed during the process of E-STDM adaptation to DREAM 2 Socialbot architecture. E-STDM has a large set of modules that can be used out of the box, but these modules are not optional and are always loaded with the program which increases the resources consumption of the service. Using pre-built modules can be inconvenient when we need to include all of the related modules. For these cases, E-STDM suggests writing your own modules, but writing one such module may seem redundant, and if there are many such modules, then it becomes not easy to work with them.

Given all these disadvantages we decided to built our own framework `DFF` on the top of E-STDM. Development with the framework is organized in such a way that writing a dialogue script in `python` is as simple, fast, and flexible as possible, and the framework also consumes an order of magnitude less memory than E-STDM. A special extension was made for the framework, which accelerated the writing of the script in cases where the standard set of functions is sufficient.

Recently, a variety of frameworks for the development of dialogue flows have appeared to speed up the process of creating a dialogue system. They often allow developers to customize natural language understanding (NLU) modules and control dialogues using state machines. Other frameworks require more expertise but give a more fine-grained control by following the formulation of the dialogue control information state [38, 18, 22]. This information state-based design provides support for complex interactions but sacrifices the intuitiveness and speed of development [24].

Table 1 shows a comparison of existing frameworks. `DFF` is the most similar to E-STDM because it is derived from it. `DFF` has many similarities to PyOpen-Dial and botml, which support pattern matching for NLU and tight integration of external function calls. Likewise, `DFF` and E-STDM explicitly support both state machine and information state paradigms for dialogue management and also provide the ability to extend it easily by adding your own custom NLU that easily integrates pattern matching and custom modules.

`DFF` exists in the paradigm of creating a dialogue graph. Each graph has a set of states and edges also known as transitions connecting these states. Each individual user interaction with the bot is accompanied by a transition from state to state. Transitions can be global from a specific node to another specific node. Transitions can also be global transitions from any node to a specific node. For the transition to be triggered, the transition condition must be fulfilled. Since there can be many transitions from one node, the sequence of checking the transition condition is determined by the transition importance parameter.

When transitioning from one state to another, the function that is attached to this transition is executed and returns a text response to the user. Thus, during one transition, two functions are executed: one determines the condition of this transition, while another one determines the response returned to the user. These functions have access to the shared memory of the entire DREAM 2 Socialbot system, and the function returning the response can also modify the shared memory of the DREAM 2 Socialbot system.

If a graph of a dialogue flow becomes very large, then it's support becomes complex. To mitigate this issue, `DFF` allows one to create several graphs and combine them together by setting transitions between them.

Framework	License	SM	IS	PM	IC	EF	ON	ET	CM	EI
`DFF`	Apache 2.0	×	×	×		×		×	×	×
Emora STDM	Apache 2.0	×	×	×	×	×	×	×	×	
AIML	GNU 3.0			×						
RiveScript	MIT			×		×			×	
ChatScript	MIT	×		×		×	×			
botml	MIT	×		×		×				
OpenDial	MIT		×	×		×				
PyDial	Apache 2.0		×			×	×		×	
VOnDA	CC BY-NC 4.0		×	×		×	×			
Botpress	Commercial	×			×	×		×		
RASA	Apache 2.0	×			×	×		×		
DialogFlow	Commercial	×			×			×		

Table 1: Comparison of features supported by various dialogue system development frameworks. SM: state machine, IS: information state, PM: pattern matching for natural language, IC: developer-trained intent classification, EF: external function calls, ON: ontology, ET: error tracking, CM: combination of the independent dialogue systems, EI: ease of integration into other `Python`-based systems.

3.2 Knowledge Graphs and Text Databases

Many skills in DREAM 2 require factual knowledge to generate grounded responses. Knowledge Graph (KG) is a graph where vertices represent entities while edges represent relations between entities. Triplets (subject, relation, object) in KGs represent knowledge about entities that can be then rewritten to represent textual facts in the template-based manner suitable for the spoken language.

Wikidata KG[3] is integrated into the DREAM socialbot:

1. Every entity extracted with `CoBot Entities` from the user utterance is passed to `Entity Linking` annotator to find Wikidata entity identifiers for the entity substring.
2. Entity identifiers are passed from `Entity Linking` to `Wiki Parser` annotator. `Wiki Parser` finds triplets in Wikidata KG for each entity.

We use inverted index over unigrams (a dictionary where a key is a word and a value is a list of entities which contain that word in their titles) for the extraction of candidate entities. Candidate entities are ranked by:

- Levenshtein [25] distance between the candidate entity title as well as aliases and entity substring extracted from the user input;
- A number of edges leading to the candidate entity in Wikidata KG.

Following [29] and [41] entities are ranked by similarity of the context (i.e. utterance utt) and entity description d. We feed the sequence of the question tokens $utt = \{w_1, ..., w_n\}$, SEP-token, and the sequence of the description tokens $\{w_{d1}, ..., w_{dm}\}$ into the BERT. Output of the CLS-token is passed to a dense layer for classification into two classes: entity e is either relevant or irrelevant to the utterance utt. This softmax probability is used for entity ranking.

`Wiki Parser` retrieves triplets associated with the entity from the Wikidata. If one of the found triplets contains the relation «instance of» (for example, (soccer, instance of, type of sport)), this triplet is used to define the entity's type. An entity type can help to understand which topic the user wants to talk about. For example, if the user mentioned «heavy metal», the relation «heavy metal, instance of, music genre» is used to switch to `Music Skill`.

Other relations are specific for different entity types (for example, for a singer `Wiki Parser` outputs triplets which include the songs and albums of the singer, for a book — the author and publication

[3] `https://www.wikidata.org/`

year, etc.). These triplets are used in the topical skills for the template-based utterance generation. For example, if the user mentions an athlete, `Sport Skill` generates a response with the template «Oh, I kind of know him. He is a POSITION and plays in TEAM.», using the triplets with the relations «position played on team» and «member of sports team».

`Fact Retrieval` annotator takes entity identifiers from `Entity Linking` as an input. Wikidata entities have the corresponding Wikipedia pages. `Fact Retrieval` annotator extracts content of Wikipedia pages of the entities from the SQLite database. Page content is parsed to obtain a dictionary {heading1: [sentence1, sentence2, ...], heading2: [...]}, with headings and corresponding content of the Wikipedia page sections (Appendix C). Scripted skills can use these annotations to share a fact that is related to a particular property of an entity. For example, to tell the user where bears live, `Animals Skill` uses the text from the section with the heading «Distribution and habitat» from the Wikipedia page about bears. If the entity type is food, `Fact Retrieval` can also extract the «Recipes» section of the page from the wikiHow database with the title containing the mentioned food to return sentences from the page's introduction.

Although in [23] we reported a less engaging effect for responses with facts, we decided to continue using facts and knowledge but in a more controlled manner in Alexa Prize Challenge 4. In DREAM 2 Socialbot, we use facts in the following way:

- `Fact Retrieval` annotator retrieves facts for entities in user utterance from Wikipedia and wikiHow[4].
- `Fact Retrieval` annotations (along with `CoBotQA` facts) are used in the scripted skills to share non-trivial knowledge about the entities user is interested in.
- `Fact Retrieval` annotations are used in `Knowledge Grounding Skill` which uses retrieved fact as knowledge to generate response candidates for the current context.
- `Wiki Skill` uses a sequence of facts from the Wikipedia pages to discuss the entity mentioned in the user input.

`KBQA` (Knowledge-based Question Answering) annotator is intended to answer user factoid questions utilizing Wikidata KB. `KBQA` takes entity identifiers extracted from the user utterance with `Entity Linking` as an input. Then the system extracts triplets from Wikidata, which contain the entity. Described below, `Relation Ranking` component ranks the relations in the candidate triplets with a BERT-based model. The object which contains the relation with the highest score in the triplet is used as the answer.

The input to the BERT-based `Relation Ranking` component is the following: the question tokens, the [SEP] token, and the candidate relation title. Output representation of BERT [CLS] token is fed into a dense layer for binary classification into two classes: 1 if the relation is appropriate for the question (positive sample), or 0 otherwise (negative sample). The model was trained on LC-QUAD2.0 dataset [8] and achieved F1$=87.2$.

`Text QA` service answers the user's questions using Wikipedia pages. The service takes as input the paragraphs from Wikipedia extracted with `Fact Retrieval` and finds the spans of the answer. The model is based on R-NET [13]. It was trained on SQuAD v1.1 dataset [34] and achieved F1=80.3.

3.2.1 Wiki Skill

`Wiki Skill` has a list of supported entity types, and if the entity extracted from the user utterance belongs to one of these types, the skill starts the template-based conversation based on Wikipedia or wikiHow pages. `Wiki Skill` parses the extracted page to make a dictionary where keys are headings in the Wikipedia page, and values are the lists of the paragraphs that belong to the section with the aforementioned headings. The skill consequently offers the user to learn information generalizing it using the headings of the page. For example, if the user wants to talk about the smartphones, the skill produces the response «Would you like to know about the hardware of smartphones?» using the heading «Hardware». If the user continues this conversation, the skill then outputs a sentence from the section with the heading «Hardware», and proposes information under the next heading, e.g., «Are you interested in software of smartphones?».

[4] `https://www.wikihow.com/`

3.2.2 Wiki Extension of Dialogue Flow Framework

Wiki Skill was extended to facilitate the development of the small talk scripts that relied on pages from the Wikipedia. The Wiki Skill variant of the small talk script contains markup for entity extraction, slot filling, facts insertion, and for switching to the different branches of the dialogue. The main functionality of Wiki Skill is:

- extraction of entities and their types using CoBot Entities, Wiki Parser and regular expressions from the user utterances;
- dialogue branching based on the conditions of the different types: patterns, extracted entities, types of these entities, dialogue acts;
- filling slots with the extracted entities and Wikidata triplets;
- automatic integration of the facts from Wikipedia and wikiHow into the script;
- acknowledgments towards the user utterance based on checking of different conditions within the user utterance.

Each small talk script is a list of dictionaries of utterances. An example of single utterance is presented in Figure 2. The available parameters of the utterance are the following:

- «utt» contains the list of sentences which will be joined to compose the socialbot's response (sentence can be skipped if it contains slots but entities for filling the slots were not extracted);
- «subtopic» refers to the script branch that the utterance belongs to;
- «expected_entities» is an optional list of entities which are expected to appear in the next user utterance (entities can be defined by one of the tags from CoBot Entities, Wikidata entity type from Wiki Parser or a regular expression). In Figure 2) the extracted entity from the next user utterance will be saved to the shared memory under the key «user_hobby»;
- «facts» is an optional parameter containing the list of knowledge sources (Wikipedia or wikiHow page) the socialbot can discuss with the user;
- «cond» defines the condition which is checked to move into the discussion of the knowledge source ([[«is_yes», «user», True]] defines checking user agreement within dialogue acts). The conditions can include matching a regular expression pattern in the user utterance, user dialogue acts, or checking the existence of entities of specific CoBot Entities and Wiki Parser types within the user utterance. Parameter «cond» is used for switching between branches in the different parts of the script.

```
{"utt": ["Recently I found some life hacks and I think "
         "collecting them may become one of my hobbies.",
         "Just for your interest, I can share with you how to use life hacks.",
         "Do you agree?"],
 "subtopic": "user_doesnt_like_hobbies",
 "expected_entities": [{"name": "user_hobby", "cobot_entities_type": "misc"}],
 "facts": [{"wikihow_page": "Use-Life-Hacks",
            "cond": [["is_yes", "user", True]]}]}
```

Figure 2: Utterance sample in Wiki Skill small talk implementation in Python. Sentences from «utt» is a next socialbot response in the dialogue branch «subtopic». If user response to this utterance contains agreement, the socialbot will share information from «wikihow_page». Parameter «expected_entities» determines that if user response contains any entity of a given type it will be stored as a variable to be used further in the dialogue for slot filling.

An example of a conversation with Wiki Skill small talk script is presented in Figure 6 in Appendix B. Wiki Skill small talk mode is used to promptly implement the topic-specific scripts to cover popular topics that are not supported by any of the DFF skills. Wiki Skill covers the following topics (in alphabetical order): anime, art, bitcoins, cars, chill, dinosaurs, family, friends, Harry Potter, hiking, hobby, love, politics, robots, school, sleep, smartphones, space, TikTok, work. All small talk scripts contain several turns and utilize knowledge sources if possible.

3.3 Knowledge-Grounded Response Generation

Different methods of knowledge integration presented in Section 3.2 have two main disadvantages. Firstly, these methods are using direct statements coming from the encyclopedic-like or generic written articles, and make socialbot's speech more robotic. Secondly, Amazon Alexa receives transcribed human speech which brings speech recognition errors and more importantly, colloquial expression and phrasing that makes factual questions answering difficult. In theory, both disadvantages could be resolved by the generative skill based on this knowledge.

Knowledge Grounding is an approach to generate a response containing new information from the provided knowledge relevant to the context of the conversation. Knowledge-Grounded Response Generation is implemented in `Knowledge Grounding Skill` that uses a ParlAI Blender 90M model [35] fine-tuned on Topical Chat Enriched dataset [16] as its core. The model input consists of the current user utterance, conversation history, and a paragraph of knowledge.

To find the best length for the grounding knowledge, we fine-tuned ParlAI Blender 90M model on the data grounded with one sentence and three sentences of knowledge. Scores before and after fine-tuning for the socialbot setups are presented in Table 2. The number of the fine-tuning epochs is determined by running fine-tuning until the validation perplexity stopped getting better.

Context length	**Epochs**	**Before fine-tuning**		**After fine-tuning**	
		PPL	**Token acc.**	**PPL**	**Token acc.**
One knowledge sentence	47.85	18.92	0.41	10.97	0.49
Three knowledge sentences	61.42	19.81	0.41	11.00	0.49

Table 2: Perplexity and token accuracy scores before and after fine-tuning ParlAI Blender 90M model on Topical Chat Enriched dataset for one knowledge sentence grounding and three knowledge sentences grounding. Scores provided for validation rare set that contains entities that were infrequently or never seen in the training set.

Table 8 in Appendix A lists examples of knowledge-grounded conversation with true labels and responses generated by the fine-tuned ParlAI Blender 90M model for conversational data from the Topical Chat Enriched test set.

`Knowledge Grounding Skill` aims to develop a conversation on a given topic grounded on all available knowledge sources. It uses news descriptions from `News Skill` to continue news discussion or as a source of knowledge about recently mentioned entities. The skill utilizes facts from `CoBotQA` and `Fact Retrieval` if available for currently discussed topic or entity. `Knowledge Grounding Skill` is also used in a case when a user wants to change the topic. If the user does not specify the subject of the conversation, the skill generates a prompt based on the hand-crafted facts on one of the popular conversation topics (games, movies, sports, science, music, food, emotions, relationships, weather, activities, celebrities, children, travel, art, jokes).

3.4 Goal-Aware Dialogue Management

One of our original tenets, Goal-Aware Dialogue Management, was proposed to enable dialogue management based on the understanding of the user's and socialbot's goals. However, our existing mostly single-turn dialogue management lacked the functionality to transform these high-level goals into the turn-specific activities. Understanding of the user's and socialbot's goals, in turn, required laying out the foundation necessary for modeling them and using them in the conversation. During Amazon Alexa Prize Socialbot Challenge 4, we decided to begin work in all of these directions.

One of the past Alexa Prize teams, Gunrock [27], observed different types of user conversation styles in their socialbot. Submissive users tended to follow the dialogue flow initiated by the system, whereas dominant users liked to direct the conversation. They calculated a ratio of utterances classified as «QUESTION», «COMMAND», «OPINION», and «STATEMENT» dialogue acts to identify user's type. In our socialbot, closer to the end of the Semifinals, we added a similar mechanism to detect user's type by adding Big5-based [5] Personality Detection module [19]. While there are 5 personality traits, we picked only introversion/extroversion to aid in detecting extrovert users and enabling them to lead the conversation with the socialbot providing short reactive responses. The system calculates the median of the user's utterance classifications across 5 last turns to classify the user as either an introvert or an extrovert. Unfortunately, this approach, introduced within the `Generic Responses Skill`, alongside with `Speech Function Classifier` and `Speech Function Predictor` has only been tested with the internal users.

We introduced the concept where entities mentioned by the user or the socialbot are stored together with the recognized user's attitude. This attitude can be positive, neutral, or negative.

In one of the skills, `Gossip Skill`, socialbot's attitude is randomly generated during the first interaction with the entity, and is later used to express socialbot's opinion within the dialogue. The user's relation to an entity is extracted based on the sentiment classification of the user utterance that mentioned the given entity. While our original DREAM 1.0 socialbot from Alexa Prize Challenge 3 used generic `Dialogue State` to represent all annotations across the system, we expanded the aforementioned concept into a shared component to more prominently store entities and attitudes to them within the dialogue.

In another skill, `Bot Persona Skill`, we initially created a list of 20 most popular things with short explanations expressing our socialbot's opinion towards them. Then, to collect the top 20 popular items appearing in the conversations, we analysed 81408 dialogues where users told about their favorite things or asked the socialbot about its preferences. As a result, we selected the 20 most favored objects: breakfast, movie, book, game, color, song, food, animal, TV show, thing, kind of sport, singer, actor, day, series, book genre, music, number, pet, sports team, anime.

In the next step, we derived higher-level categories primarily based on topics covered by our scripted skills (movies, TV shows, sports, sports teams, music). For each category, `Bot Persona Skill` can explain why it prefers the particular item from the corresponding category. Using the `Wiki Parser`, the skill can express its opinion about entities related to the given category.

3.5 Speech Function Classifier and Predictor

One of the past Alexa Prize teams, Slugbot [1], proposed a DRDM dialogue model to control the coherence of the open-domain dialogue using discourse relations. Their approach introduced a combination of dialogue acts and four discourse relations from the Penn Discourse TreeBank [32] as means to model interaction within individual turns and at a higher level. However, 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 casual conversation analysis. Instead, Eggins and Slade in their work [9] introduced a similar connection between individual turns and cross-turn discourse structure patterns specific for spoken language as the higher-level abstraction that operates across multiple turns, enabling interactive and sequential conversational experience. At turn level, they extended Halliday's concept of Speech Functions [14, 15] which are an alternative to Dialogue and Speech Acts. At the higher level, they introduced a concept of Discourse Moves that are directly connected to the Speech Functions.

Speech Functions and Discourse Moves have been originally used by Mattar and Wachsmuth in 2012 [30] in the virtual museum agent as a mechanism that enables small talk. However, their speech function classifier's taxonomy was greatly reduced to support just a small talk within the mostly goal-oriented dialogue system. We needed a broader taxonomy that would focus first on the open-domain dialogue.

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. Two versions of the `Speech Function Classifier` were developed. 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 second version 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` labels each phrase in the user and socialbot utterance with the Speech Function. This classification enables the socialbot to predict the socialbot's response Speech Functions most expected by the user for a given last user's phrase Speech Function. While we ran an experiment with the internal users, unfortunately, we did not have time to test this concept on real users during the Semifinals.

3.6 Dialogue Acts Driven Skill Generation

Our work on the Goal-aware dialogue management led us to understand that even short small talks imply setting and achieving different conversational goals, e.g. greeting, requesting an opinion, or sharing information. While script-based skills use goals incorporated by the developers, we can only wish that both retrieval and generative skills can aid in conducting a goal-aware dialogue. On the other hand, script-based skills have a substantial disadvantage, which is a development cost. Being encouraged by the success of the script-based approach [10] for the Alexa Prize Challenge 3, we introduce a method for the automatic skill generation. So, the proposed idea is to create a goal-oriented skill on top of the conversational data. Goal-oriented skills usually utilize custom intents and entity detection as a natural language understanding module and next action prediction alongside with the slot-filling as a natural language generation module. Therefore, a dataset annotated with the dialogue acts as abstract intents and detected entity types can be used for goal-oriented skill construction.

We decided to use the Topical Chat Dataset [12] which contains short dialogues focused on the single-topic conversations between real people for the first version. We annotated every utterance in the dataset with dialogue acts using `MIDAS Classifier` and mentioned entities alongside with their types using `CoBot Entities` annotator. We consider *system action* as a combination of MIDAS dialogue act and types of entities extracted from the utterance.

The architecture proposed in [40] was used to involve dialogue acts and entity types into the dialogue management. RNN model learns to predict the next system action based on the full dialogue history. Vector representation for each utterance is composed of utterance embedding and one-hot encoding representation of both the involved entity types and its dialogue act. The response of the skill is a random utterance from a subset of the system utterances labeled with the predicted system action. Then an optional slot in the system utterance is filled with the user-mentioned entity of the same type. This can be further improved by filling system utterances with entities connected to the user-mentioned ones, for example, using knowledge graphs.

For proof-of-concept experiment, system utterances could contain more than one entity which led to the huge number of system actions. Therefore, we considered only a subsample of the Topical Chat consisted of the dialogues related to the Sports discussions. The next action prediction accuracy of the trained model was 0.9242. Close-reading analysis revealed that system predictions were quite irrelevant for the user most of the time.

Getting rid of topic-specific restrictions, we utilize only dialogue turns where system utterance contains no more than one entity to limit number of system actions. While the proposed approach was not good enough to be integrated into the general DREAM pipeline, there are several promising directions to improve the system. Possible enhancements are described in Section 6.

3.7 Recommendation Models

Increasing number of the scripted topics leads us to the necessity to control offered topics. For that, we can utilize not only the dialogue history but also the structured user personality — starting from the main information, like age group, to the user's preferences extracted by `Entity Storer`. Therefore, in this section we present `Topic Recommendation` annotator which offers a topic for further conversation using the information about the discussed topics and user's preferences. `Dummy Skill` generates a linking question to the scripted skill supporting the recommended topic. Then `Response Selector` can either choose response candidate as a final response or join linking prompt to another response candidate. `Topic Recommendation` annotator aims to recommend topics that the user is likely to support. It is important that we can proactively suggest the next topic to the user when the user wants to change the topic but does not specify which one, and when the conversation with specific scripted skill is coming to the end.

There were several experiments with an applied model for recommendations, including Logistic Regression, TF-IDF, and ConveRT. Originally, we assumed that topic recommendation implies the presence of a dialogue dataset marked with the considered topics. As we did not have a good enough classification model for the topics we covered with scripted skills, we decided to utilized `CoBot Entities` annotator, and manually map different entity types to the considered topics.

3.7.1 Entity Recommendation

The first version of the Entity Recommendation model used Logistic Regression to predict entities and entities labels which could be mentioned in the conversation. For every utterance in Topical Chat, `CoBot Entities` extracted mentioned entities and tagged them by type. The dataset contains 4152 samples with 157776 utterances. We excluded some non-informative entity types («misc», «anaphor», «number», «duration», «year», «date»), so the final number of the entity types was 22. The number of the unique occurred entities was equal to 10061. 200 top frequent entities were chosen to be predicted.

For the model recommending N objects, feature vector consists of 3 vectors of dimension N. The first vector contains at the corresponding position of each object the portion of its occurrences in the utterance history. The second one includes 1 for mentioned in the last utterance objects, and 0 – for others. And the last one includes 1 only for the candidate object. Then these 3 vectors are concatenated, and the final feature vector with dimension 600 is given to the model as an input. If the candidate object was mentioned in the dialogue sample, the feature vector is labeled by 1. If the candidate object is chosen randomly, the feature vector is labeled by 0.

Eventually, 40000 samples were prepared for training and 4214 samples for testing. The results of label and entity recommendations can be found in Table 3. After an entity or an entity label is recommended, the predicted value needs to be mapped to a specific scripted skill. Accidentally, we did not estimated its quality on real users due to integration bugs.

Type	Accuracy, %	Train	Test
Entity	66.6	40000	4214
Label	76.3	95000	10286

Table 3: Accuracy scores of Logistic Regression Model for recommendation models of the next mentioned entity and entity type. Train and test dataset sizes are shown in corresponding columns.

3.7.2 ConveRT-based Topic Recommendation

The second approach was based on the ConveRT model that is a model for ranking responses for the given context. The key idea is to replace response ranking with *recommendations ranking*. Assume a recommendation is a sentence proposing the next topic for the conversation. Then ConveRT could be used to choose topic proposal the most suitable to the current dialogue context. The first set of possible proposals was created as a set of template-based sentences «let's chat about TOPIC» where «TOPIC» values generalize topics of particular scripted skills. The second method to create possible proposals was to use linking questions. Unlike the previous method, in this case every topic had several corresponding proposals. If the question was ranked in top-k, the corresponding topic was labeled by 1, and 0 otherwise. One more method to create proposals set was to unite both proposals sets.

The results of the ConveRT model evaluation are presented in Table 10 in Appendix F. At that moment, only 9 topic scripted skills were available. These results can be considered sufficient but could be improved by taking into account the user's personality. Unfortunately, Topical Chat dataset contains coincidence topics but does not collect user preferences.

3.7.3 Topic Recommendation based on Reddit Personality

The main idea of the third approach is to build representations of user personalities from Reddit and then to find the most similar to the current user. So, topic recommendation is produced by the use of information about similar Reddit users. We collected information about subreddits in which the user submitted the last 10 posts and left the last 10 comments. 2878 subreddits with 31578 submissions were received for 665 users. Then these subreddits were classified by 12 topics by keywords and language model BART [26]. As a result, each user is represented by a vector of dimension 12 where each element of the vector is equal to the portion of occurrences of the corresponding topic among all user's posts and comments. The representation of the current user persona is created as portions of scripted topic-specific skills responses in the dialogue. Similarity scores are obtained with cosine similarity between vector representation of the current user and all considered Reddit users.

The evaluation process was performed in the same way as for the ConveRT model. The results are presented in Table 11 in Appendix F.

The recommendation model was tested on the real users. The percentage of the user's agreement to talk on the recommended topic was counted. The agreement was identified by the positive sentiment or agreement intent. The final results can be found in Table 4. There is a slightly improved agreement percentage with recommendation model that should be improved to be more useful for the dialogue management.

Approach	Agreement, %	Samples
Without recommendation (random choice)	59.6	334
Topic recommendations with TF-IDF	61.2	778

Table 4: Accuracy scores of user's consent prediction with and without topic recommendation based on Reddit users' personalities.

3.8 Trainable Hypotheses Ranking Model

There are two main approaches for ranking candidate responses. The first method is to obtain an independent representation of the candidate response and context, and then compare these representations to determine their relevance [37, 17]. The second approach is to determine relevance based on the combined representation of both candidate response and context [42, 47, 39]. Our solution consists of two stages. At the first stage, we extract features from our candidate responses. For this we use ranking models that assess intermediate relevance, which is added to the rest of the features received from annotators. At the second stage, the obtained features are used to obtain the final relevance.

For fine-tuning, we used TopicalChat dataset [12]. For the final task, we have mapped out 30 selected dialogues with real users. The length of the dialogue is about 30-40 turns. At every step of the dialogue, 6-12 candidate responses were marked out. Thus, in total, we collected about 10 thousand triplets consisting of a context, candidate response, and candidate response label.

Table 5 shows the results of the first stage models. The baseline model chooses the candidate response with the highest confidence value. The ConveRT model [17] trained on the Reddit dataset generates an independent representation for each utterance, after which the relevance is assessed through the pre-trained function. Another model is UMS-ResSel model [39], that is based on BERT, but it uses additional strategies for training process. That strategies extend the loss function of vanilla BERT by adding special operations (insertion, deletion, search) for improvement order understanding of utterances of a dialogue.

Model	P@1	R@1	R@3	R@5	R@10
Baseline	51.2	50.2	71.6	85.3	99.0
ConveRT	47.9	46.8	74.0	89.0	99.1
ConveRT (finetuned)	52.6	49.9	71.6	88.9	99.6
UMS-ResSel	47.6	47.6	69.7	84.5	99.0
UMS-ResSel (finetuned)	48.2	48.0	69.5	85.3	99.1

Table 5: The results of response selection models without gradient boosting performed on our dialogue dataset. Baseline – top@ based on internal confidence of a skill. ConveRT – response selection by Transformer-based model pre-trained on Reddit. UMS-ResSel – response selection by BERT-like model. Models were fine-tuned on the TopicalChat dataset.

For the second stage, we used additional annotators – Dialogue Breakdown (DB) [31] and MIDAS [44] classifier. In this stage, models based on the gradient boosting performed the best. Two implementations of this algorithm were considered: XGBoost [4] and CatBoost [7]. Hyperparameters were selected by the grid search method. Table 6 shows that the model based on CatBoostClassifier received the best result in comparison with other gradient boosting algorithms by using the ConveRT relevance score, the markup with dialogue acts from MIDAS, and the feature from Dialogue

Breakdown and the XGBRanker model achieved the best result with all annotations and confidence added.

Model	Features	P@1	R@1	R@3	R@5	R@10
CatBoostClassifier	ConveRT, MIDAS, DB	63.5	57.0	80.9	93.3	99.7
XGBRanker	ConveRT, MIDAS, DB, C	65.7	61.3	83.1	94.3	99.8
XGBRanker	ConveRT, MIDAS, DB, C, A	68.6	63.1	84.3	94.4	99.8

Table 6: Comparison of second stage models with an extended set of features. Features column denotes which additional features were utilized, particularly, DB denotes Dialogue breakdown labels, C – confidence, A – annotations of CoBot Conversation Evaluation.

3.9 Multi-Task Classifier

Multiple annotators and skills of the DREAM social bot use the pre-trained neural models that consume enormous computational resources. However, the computational cost available to us is limited. Moreover, the work of `Response Selector` requires use of CoBot API services for all candidate response annotations, while the number of queries to API is also limited. These restrictions led us to the idea of «squeezing» several classification models into one to lower down the computational costs. These models are `CoBot DialogAct`, `CoBot Topics`, `Sentiment Classifier`, `Emotion Classifier` and `Toxic Classifier` which description is given in [23].

We compressed the functionality of the models into the single BERT-base model. We used samples from the dialogues with real users from the Alexa Prize 3 which were labelled by all considered models. Although we researched over different pseudo-labeling approaches [20], in our task the number of the examples that already had labels from all these models was so high that there was no need in using this approach. Specifically, the train set contains 468237 samples, the test set – 10597 samples. We used the original raw utterances without history as a sample truncating the phrase length to 32 tokens. Apart from the unification of all 6 models, we experimented with the unification of only CoBot models and only non-CoBot models. In all settings, we considered all labels to be independent from each other.

The results on the test set for the models we received and the original models are presented in Table 7. We should note that «accurate» labels for CoBot tasks those ones obtained by original CoBot API services. For other tasks, the train, validation and test sets were the same as in [23]. We also tried to add history (3 utterances) to the `Combined Classifier` that yielded increase in accuracy (by about 9%) and F1-score (by about 10%) for `CoBot DialogAct` tasks (original API service utilizes history). However, we did not integrated this model as it required increasing the input size from 32 up to 64 tokens. For the sake of achieving the best balance between latency and accuracy, we chose the variant of the **Combined (6 in 1)** model as a final variant to be used in the socialbot.

4 DREAM Socialbot Evaluation Results

After the Alexa Prize Challenge 3, our team continued work on the development of the DREAM socialbot. We used the final version of the original DREAM socialbot [23] as the starting point. Given that we no longer had an access to the `CoBotQA` remote service that was used for factoid question answering and knowledge retrieval in the original DREAM socialbot, we had to implement our own solution. For that, we have integrated basic versions of the following knowledge graph components: open-domain question answering model (`ODQA`), knowledge base question answering (`KBQA`), `Entity Linking`, factoid questions detection (`Factoid Classification`), as well as factoid questions answering skill (`Factoid-QA`).

We used Docker Swarm for deployment in our original DREAM socialbot. However, given the growing complexity of the solution we decided to move on to more advanced orchestration system. We migrated to Kubernetes on AWS EKS to get much needed flexibility. Although we kept other tracking and analytical components of the last year, transition to Kubernetes turned out to be a

Model name	Source models	6 in 1	CoBot – 3 in 1	Non-CoBot – 3 in 1
Custom CoBot Topics	—	0.84 (0.83)	0.82 (0.84)	—
Custom CoBot DialogAct Topics	—	0.76 (0.64)	0.78 (0.66)	—
Custom CoBot DialogAct Intents	—	0.69 (0.65)	0.70 (0.67)	—
Emotion Classification	0.92 (0.75)	0.82 (0.60)	—	0.85 (0.67)
Sentiment Classification	0.72 (0.68)	0.60 (0.57)	—	0.66 (0.62)
Toxic Classification	0.92 (0.60)	0.92 (0.59)	—	0.93 (0.60)

Table 7: Combined classification: accuracy and F1-score in brackets on the test sets for 6 tasks for different models. Source models denote separate models, CoBot – 3 in 1 denotes model trained on `CoBot Topics` and `CoBot DialogAct` annotaions, Non-CoBot – 3 in 1 denotes model trained on emotion, sentiment, and toxic classification tasks.

significant challenge for our small team. Only by the end of January, we managed to get deployment under control.

In Alexa Prize Challenge 4, we participated in 3 official competition phases: Initial Feedback (January 18 – March 1), Quarterfinals (March 2 – April 30), and Semifinals (May 4 – June 25) periods. However, for analysis purposes, we identify shorter periods in the DREAM socialbot development. In Figure 3 daily, moving average and stage average ratings of DREAM socialbot are presented.

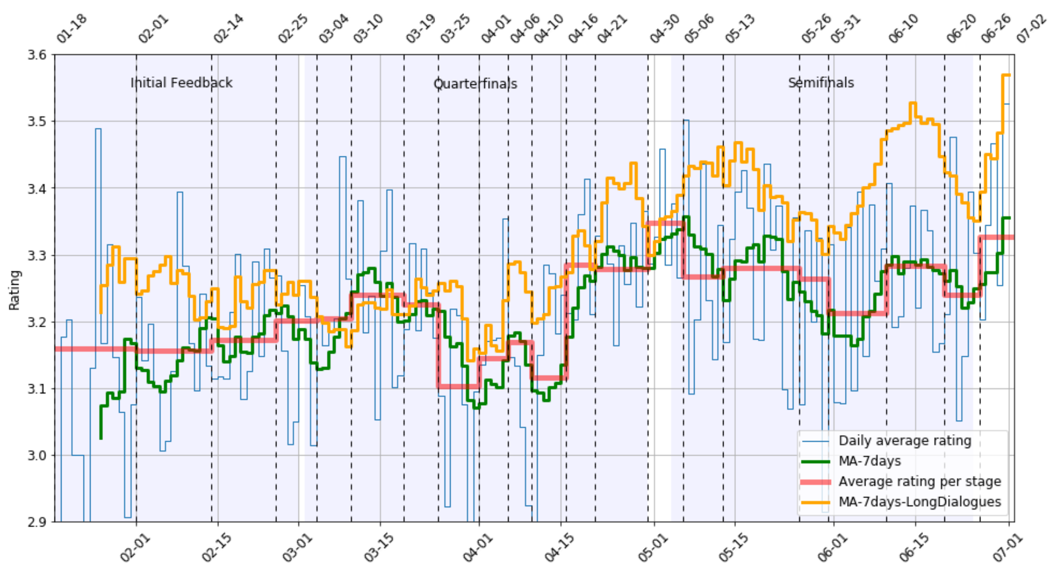

Figure 3: Average daily DREAM Socialbot rating. Daily rating is in blue. Thicker green line is a moving average of daily ratings for the last 7-days. Vertical dotted lines separate different stages of DREAM socialbot development. The thickest red line shows average rating during the stage. Shaded area corresponds to different official phases of the competition. Orange line corresponds to moving average of daily ratings for dialogues of 20 and more utterances.

Initial Feedback period officially started on **January 18**. By that time, DREAM 2.0 socialbot changed significantly from DREAM 1.0 socialbot: it not only used our own knowledge graph-driven components such as `KBQA`, `ODQA`, and `Wiki Parser`, but also got an updated `ConveRT Reddit`. Between January 18 and February 1 DREAM 2.0 socialbot was still unstable due to continuing issues with deployment, and we had to resort to the original DREAM 1.0 socialbot for the half of the entire user traffic to maintain uninterrupted user experience. On **February 1**, we put an end to DREAM 1.0

socialbot. Average rating of DREAM socialbot during the first half of February is similar to the average DREAM rating during January, so DREAM 2.0 and DREAM 1.0 have similar quality.

By mid-February, our observations showed that the updated `ConveRT Reddit` provides worse responses compared to its version from the original DREAM 1.0, and we reverted to the original version on **February 14**. At the same time, though, we released our new generative `Knowledge Grounding Skill` programmed to be invoked rarely, with low confidences. This helped us to accurately measure user's reaction and fine-tune this skill. These changes along with the updates to the existing components slightly improved our ratings.

During the **second half of the February** we have enabled `Wikidata Dial Skill` couple times for a few days but generated responses were of insufficient quality, so we removed the skill. As we did not have an access to `CoBot Dialogue Acts and Topics` after the Alexa Prize Challenge 3, we used their annotations collected during the last year to build our own versions of the CoBot-based classifiers. However, as the resulting resource consumption has grown significantly, we integrated them alongside with the existing classifiers for toxicity, emotion, and sentiment detection into the `Combined Classification` described in Section 3.9. However, with the restored access to the original CoBot classifiers, and with their increased performance, we reverted to the Amazon-provided CoBot classifiers for the duration of the entire Challenge. This decision made a positive impact on the average daily ratings.

On **February 25**, we have released `Dialogue Flow Framework` and decided to double down on the content of the socialbot. `Travel Skill` was first released on **February 27**, and, after a few days, we added the first versions of `Animal Skill` and `Food Skill` on **March 4**. By **March 10**, we have already had three improved topic-based `DFF`-skills, and released `Sport Skill` and `Music Skill`. All these gradually improved and refined skills had a positive effect on the average rating. At the same time, pre-trained original MIDAS [44] model was integrated into DREAM as `MIDAS Classifier`. On **March 10**, we disabled `Knowledge Grounding Skill` as its responses demonstrated that the skill still has to be vastly improved. Two days later, we also changed the beginning of the dialogue from offering users just three most popular topics (movies, books, and games) to an option to choose between two random topics covered by our skills. These substantial changes helped us to improve our daily ratings significantly, confirming our assumptions about the importance of scripted content in the socialbot.

On **March 19**, `Knowledge Grounding Skill` was enabled once again, this time enhanced with the fine-tuned model utilizing knowledge up to 3 sentences. Seeing a decrease in rating, we continued our work on balancing of the `Knowledge Grounding Skill` confidences and conditioning for turning it on or off. On **March 25** we released a new greeting script with questions about weekends. Unfortunately, most of the users did not want to share the details about their weekend time, and when they did the mentioned activities rarely if ever led to the scripted skills. This made it harder for the socialbot to continue a coherent and engaging conversation after this scenario which negatively affected the daily ratings.

Originally, we have extracted conversational subjects using `NER` for named entities and `CoBot Noun Phrases`. On **April 1**, we transitioned subjects extraction from noun phrases to entities using the `CoBot Entities` service provided by Amazon. We also added sentiment-based filtration of negative news in `News Skill` and negative predictions of commonsense aspects in both `Activities Discussion Skill` and `Personal Events Discussion Skill`. Simultaneously, `Entity Linking` algorithm was significantly improved with the use of context. In the beginning of April we finally turned off `Alice`, all topic-based `TFIDF-Retrieval Skills`, and event-based skills. Discussion of human activities obtained from user utterance in `Activities Discussion Skill` was also disabled. These improvements increased our ratings.

On **April 6**, we released the first version of the new `Response Selector` described in details in Section 2.1.2. It used our empirical formula for ranking responses inside the same priority group. `Knowledge Grounding Skill` began generating response candidates based on the `CoBotQA` knowledge. We also started to utilize GNews API that improved quality of `News Skill`. Two days later we introduced «disliked skills» approach – if user refuses to discuss, shows negative or toxic reaction to linking question to the scripted skill, this skill is marked as «disliked» and will never be offered to the user again. Around **April 10** we fixed substantial bugs with the scripted skills activation, facts formatting in `Fact Retrieval`, and with timeouts across the socialbot's pipeline. We also decreased frequency of the universal dialogue act-based responses from `Grounding Skill` that interrupted the dialogue too often. These fixes had a significant positive impact on the DREAM daily rating in mid-April.

In **mid-April** we noticed that some of the other competing socialbots do not interrupt when user requests something and simply continue following the script. In cases when user dialogue act requires action from the socialbot (for example, if user requests opinion, socialbot's dialogue act should be opinion expression), current version of `Response Selector` removed priority from the scripts, choosing a final response among response candidates containing corresponding dialogue acts. Due to a significant focus on fixes and new features during the competition, we have not had an opportunity to compare these two strategies, so we just switched between them trying to manually estimate user reaction several times during the challenge.

`MIDAS Classifier` provides very useful information about dialogue acts in user and socialbot utterances, so we also concentrated on its improvement and on **April 21** replaced the original model with the BERT-based classification model trained on the semantic classes subset of the MIDAS dataset [44]. Classification quality improved which had a positive impact on universal responses by `Grounding Skill`, script-based skills and `Response Selector` in general.

On **April 30** `Knowledge Grounding Skill` started to generate response candidates based on the news descriptions from `News Skill`. At the same time, we turned on interrupting scripted skills if user dialogue act requires a corresponding response dialogue act. One more important feature released in the end of April is `Wiki Skill` information-based dialogue about different subjects and concepts. We also shipped `Wiki Extension` of `Dialogue Flow Framework` for conducting small talks about some popular topics for which we did not have script-based skills. These fast but considerable improvements increased our daily ratings to almost the best values during the competition.

An important dialogue management feature, linking questions, was widely used to lead the user to some scripted skill. However, random use of these linking questions might not be the best choice as it can give the impression of an unreasonable topic change. So, on **May 6** we added pre-linking connection phrases which take into account the recent topic if available, and topic to be linked to. These phrases are citations, short interesting facts, personal opinions, and simulated thoughts of our socialbot.

In **mid-May**, we deprioritized for response candidates with the same entities as in the user utterance, and also ramped up the scripted skills priority even if user dialogue act requires some particular dialogue act. On **May-26**, we decided to stop offering topic choice in the beginning of the dialogue. A few days later we also made socialbot's sensitive mode to be enabled only for selected dialogue acts for sensitive topics (e.g. opinion request on politics), so now user toxic utterances were processed like the rest of the utterances. In the **end of May**, new ranking model for the response candidates was finally integrated to the `Response Selector`. Experiments with `Response Selector` as well as removing topic suggestion in the dialogue beginning could lead to ratings decrease.

We tried out several different strategies for dialogue beginning: asking how user is doing, asking for the user's name, asking about hobbies, asking about daily life, immediate switching to scripted topics. More than half of the dialogues are finished right after invocation, half of the rest of the dialogues contains less than 2 dozen utterances, so the dialogue beginning strategy could significantly impact on ratings of short dialogues. Therefore, our focus on the content at the cost of the more careful study of the dialogue beginning could be the key reason our total rating didn't grow well enough. However, the average rating of the long dialogues increased significantly during the Challenge (see Figure 3). The **first half of June** was finally dedicated to dialogue beginning fixes, including improvements to the scripts discussing work and school as the most popular weekday activities. On **June 10**, we fixed response time previously accidentally increased by reducing information stored in the dialogue state. Careful reading of the dialogues demonstrated that while asking for weekdays activities in the beginning of the dialogue can potentially help to separate users into age groups (child or adult), it is not easy to provide the user with some engaging feedback on their daily life. So, on **June 20**, we changed the greeting part to the «how are you» exchange followed by linking to one of the scripted skills. At the same time, we fixed a bug with enabling `Topic Recommendations`. We also integrated compliment acknowledgements to user opinion expressions to please users.

On **June 26**, we continued our experiments with `Response Selector` disabling priorities for the scripted skills when the user's dialogue act expects a given action. A few days later we also conducted

AB-test to compare two similar strategies for user dialogue acts which require some actions: (1) choose among response candidates with the expected dialogue acts with ranking model, (2) join this candidate of the expected dialogue act response with the next script line using special «Let us get back» connections. This AB-test showed no statistically significant differences between these two versions of the socialbot.

In Figure 4 one can find dialogue lengths in utterances which decreased significantly during Semifinals period (shorter by about 10 utterances). Simultaneously, we were increased socialbot utterance length adding acknowledgement to show user our understanding and sympathy, pre-linking connection phrases to make topic switching more coherent, discussions based on information sharing by `Wiki Skill`. Therefore, increasing average bot utterance length potentially could negatively affect dialogue length. In Figure 5 we demonstrate daily portion of dialogues with returning users among all dialogues per this day. During Semifinals about 8% of the dialogues per day were conducted with users having 5 and more conversations per month (user identifiers are reset every month), and about 4% of the dialogues – with users having 20 and more conversations per month in total. Although `DeepPavlov Agent` allows to store information about previous user conversations, we do not utilize it properly.

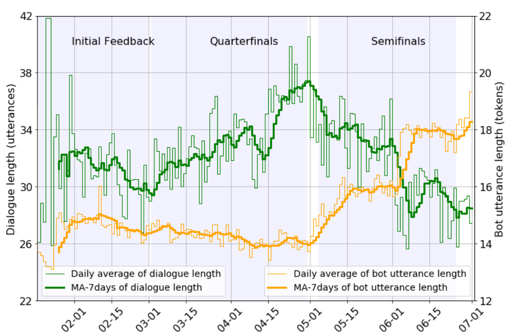

Figure 4: Daily average of **dialogue lengths** in utterances (on the left y-axis) and daily average of the **socialbot utterance lengths** in tokens (on the right y-axis). The dialogues containing only invocation and stop commands by user are not included. Dialogue length significantly decreased during semifinals while we were working on increasing socialbot utterance length.

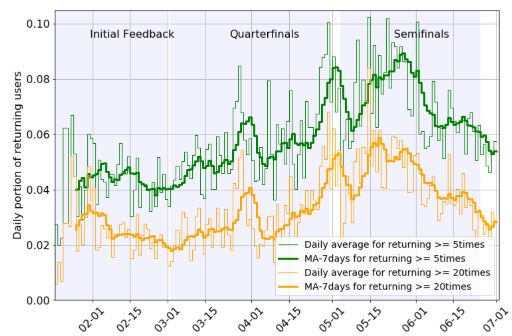

Figure 5: Daily **fraction (among all dialogues this day)** of conversations with users returning **5 and more** and **20 and more** times in total. User identifiers are being reset every month. Average portion of returning users increased almost twice during semifinals period.

5 Conclusions

One of the major tenets of this Challenge for us was proper integration of the Knowledge Graph information into the dialogue. We use KGs for natural language understanding (NLU) and natural language generation (NLG) across both slot-filling and neural generative models. KGs are also used to direct or even change the dialogue flow. Although commonsense completion models sometimes provide inadequate predictions, we actively utilize them for NLU, and with further improvements of the commonsense generation models we plan to expand their use for NLG.

We also paid a lot of attention to the scenario-driven content of the dialogue to provide a coherent multi-turn dialogue flow. We introduced a `Dialogue Flow Framework` for the scenario-driven skills development, presented its extension for fast knowledge- and annotation-based scripts, and significantly expanded the number of the scenario-covered topics. Moreover, we implemented the universal knowledge-based skill capable of conducting consistent albeit limited subject-specific conversations.

Having a lot of content requires an even bigger focus on the dialogue management. We introduced a rather sophisticated response selection algorithm taking into account dialogue acts, topics, as well as entities of the conversation. Although it prioritizes scenario-driven skills and user-expected dialogue acts, this is only one of the first steps towards the goal-aware dialogue management.

Taking into account that half of the dialogues are finished immediately, and another quarter of all dialogues is shorter than 20 utterances, user impression from the dialogue beginning could play one of the most important roles. We tested several different strategies including direct offering of a particular topic by the socialbot or complete freedom of topic selection by the user. However, a lack of the socialbot's flexibility in an open topic discussion spoils the user impression in the beginning of the conversation.

Another aspect of the Alexa Prize Challenge 4 is an increasing number of the dialogues with returning users that leads to an importance of using previously gathered information about the user to establish and maintain closer relations. Currently, we work only with the high-level user information, but we could further improve the relationship between user and bot by diving deeper into the history of interaction.

6 Future Directions

6.1 Dialogue Flow Framework and DFF Markup

While introduction of the `DFF` allowed our team to introduce a number of scenario-driven skills enriched with the information coming from a plethora of the annotators rapidly, development of the `DFF`-based skills can be challenging for the newcomers. Our plan is to evolve our Wiki Extension of `DFF` further by transforming it into a Python-based DSL to significantly simplify development of the scenario-driven skills using `DFF` and DeepPavlov Agent's annotators.

6.2 Dialogue Act Driven Skill Generation

Our experiments showed an importance of dialogue acts for natural language generation. While dialogue acts provide discourse-level utterance classes, one could not derive enough pragmatical knowledge from them. That is, even if the proper response dialogue act is known, one could not simply pick a random utterance labeled with this act from some corpus. A straightforward source of additional information here is the information kept in the Dialogue State, such as the slot-filling information. Anyway, the more advanced method of constructing the response text could help. Finally, MIDAS dialogue acts could simply be not the best choice for discourse-level utterances labeling. The use of Speech Functions (see section 3.4) seems to be the promising alternative here.

6.3 Goal-Aware Dialogue Management

By the end of June 2021, we have made major contributions to our Goal-Aware Dialogue Management tenet, with work spanning response selection approach, discourse management and speech functions classification, basic bot persona modeling, and user modeling, as well as the integration of the Big5 personality detection. However, when comparing our progress with our original plans, it is clear that we have not reached our own goals yet. There is a lot of work lying in front of us with the focus on integrating and properly using these technology components towards building an engaging and thoughtful conversational partner.

We introduced `Entity Storer` as a mechanism to store user's and socialbot's attitude to the entity mentioned in the conversation, and to maintain coherent socialbot's relation to the discussed entities across the entire dialogue. In addition to that, we have introduced `Bot Persona Skill` as a mechanism to tell about socialbot's favorites and share opinions towards mentioned entities based on the relations to the top of the hand-picked categories. One of our future directions is building Knowledge Graphs for storing users' and socialbot's relations to the mentioned entities mapped to the world's KG. This mapping will enable our socialbot to calculate and predict user's relation to the higher-level categories beyond individual entities. Finally, we defined several personas based on the different preferences towards these categories. Our vision is to further extend our socialbot's KG by adding several hand-crafted personas and then adding a mechanism for generating a believable bot persona based on these personas and first user utterances in the conversation.

In the current version of the `Bot Persona Skill`, we only explored a socialbot ability to express and explain its opinions as well as telling the backstory about their favorite things. In its future version, we plan to expand this skill with the functionality towards driving the conversation based on the socialbot goals. It will also include functionality imitating socialbot emotions, making it believable that user actions influence socialbot emotional level. We expect this approach to make conversations more engaging and personal and to build an emotional connection between the interlocutors.

6.4 Discourse-Driven Dialogue Strategy Management

We plan to integrate `Speech Function Classifier` and `Predictor` at multiple levels, including `Skill and Response Selectors`, as well as in our `Dialogue Flow Framework`. With `Speech Function Classifier`, we will be able to identify multi-turn conversation fragments by mapping Speech Functions to the higher-level concept of discourse. We envision this Discourse element to be represented as a sequence of speech functions used by the interlocutors within the dialogue, `CoBot Dialogue Act Topics`, mentioned entities, active goal, and both user and socialbot relations to those entities. This component will further extend our Dialogue State with the structured information about the discourses within the conversation.

Finally, with the `Speech Function Classifier`, `Discourse Detector`, as well as with the understanding of user and socialbot goals, we will be able to dive deep into enabling our socialbot to strategically control the conversation. Our vision is to enable 3 levels of dialogue planning: (1) Goals, including helping a user to reach their goals as well as following socialbot's interests, (2) Discourse, with the focus on enabling the bot to continue a current Discourse or begin a new one, and (3) Speech Functions, with the focus on enabling bot to pick the next Speech Function based on the Discourse and the key Goal picked by the socialbot at the previous dialogue planning levels.

Acknowledgements

DREAM team is deeply grateful to the Alexa Prize organizers for their feedback and advice during the competition. DREAM team also thanks all members of Neural Networks and Deep Learning Lab for their support and making participation in the competition highly productive.

References

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

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, et al. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, 2018.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] Michael S. Chmielewski and Theresa A. Morgan. *Five-Factor Model of Personality*. Springer New York, New York, NY, 2013.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[8] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *International Semantic Web Conference*, pages 69–78. Springer, 2019.

[9] S. Eggins and D. Slade. Analysing casual conversation. 1996.

[10] James D Finch and Jinho D Choi. Emora stdm: A versatile framework for innovative dialogue system development. *arXiv preprint arXiv:2006.06143*, 2020.

[11] Raefer Gabriel, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, et al. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. *Alexa Prize Proceedings*, 2020.

[12] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, Dilek Hakkani-Tür, and Amazon Alexa AI. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*, pages 1891–1895, 2019.

[13] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. May 2017.

[14] M. A. K. Halliday. *Language as code and language as behaviour: a systemic-functional interpretation of the nature and ontogenesis of dialogue*, pages 3–36. Linguistics: Bloomsbury Academic Collections. Bloomsbury Academic, London, 1 edition, 2015/11/27/ 1984.

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

[16] Behnam Hedayatnia, Seokhwan Kim, Yang Liu, Karthik Gopalakrishnan, Mihail Eric, and Dilek Hakkani-Tur. Policy-driven neural response generation for knowledge-grounded dialogue systems. *arXiv preprint arXiv:2005.12529*, 2020.

[17] Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Ivan Vulić, et al. Convert: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688*, 2019.

[18] Youngsoo Jang, Jongmin Lee, Jaeyoung Park, Kyeng-Hun Lee, Pierre Lison, and Kee-Eung Kim. Pyopendial: a python-based domain-independent toolkit for developing spoken dialogue systems with probabilistic rules. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP): system demonstrations*, pages 187–192, 2019.

[19] Wieser Johannes. Personality prediction from text. `https://github.com/jkwieser/personality-detection-text`, 2020.

[20] Dmitry Karpov and Michail Burtsev. Data pseudo-labeling while adapting bert for multitask approaches. In *Proceedings of the International Conference "Dialogue 2021"*, 2021.

[21] Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qihan Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, Ming Cheng, Qinglang Chen, Lauren Stubel, Karthik Gopalakrishnan, Kate Bland, Raefer Gabriel, Arindam Mandal, Dilek Z. Hakkani-Tür, Gene Hwang, Nate Michel, Eric King, and Rohit Prasad. Advancing the state of the art in open domain dialog systems through the alexa prize. *ArXiv*, abs/1812.10757, 2018.

[22] Bernd Kiefer, Anna Welker, and Christophe Biwer. Vonda: A framework for ontology-based dialogue management. In *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction: 10th International Workshop on Spoken Dialogue Systems*, pages 93–105. Springer Singapore, 2021.

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

[24] Staffan Larsson and David R Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3 & 4):323–340, 2000.

[25] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, feb 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[26] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[27] Kaihui Liang, Austin Chau, Yu Li, Xueyuan Lu, Dian Yu, Mingyang Zhou, Ishan Jain, Sam Davidson, Josh Arnold, Minh Nguyen, and Zhou Yu. Gunrock 2.0: A user adaptive social conversational system, 2020.

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[29] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy, July 2019. Association for Computational Linguistics.

[30] Nikita Mattar and Ipke Wachsmuth. Small talk is more than chit-chat. In Birte Glimm and Antonio Krüger, editors, *KI 2012: Advances in Artificial Intelligence*, pages 119–130, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[31] Nathan Ng, Marzyeh Ghassemi, Narendran Thangarajan, Jiacheng Pan, and Qi Guo. Improving dialogue breakdown detection with semi-supervised learning. *arXiv preprint arXiv:2011.00136*, 2020.

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

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[35] 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. Recipes for building an open-domain chatbot, 2020.

[36] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[37] Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 695–704, 2017.

[38] 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. Pydial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78, 2017.

[39] Taesun Whang, Dongyub Lee, Dongsuk Oh, Chanhee Lee, Kijong Han, Dong-hun Lee, and Saebyeok Lee. Do response selection models really know what's next? utterance manipulation strategies for multi-turn response selection. *arXiv preprint arXiv:2009.04703*, 2020.

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

[41] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online, November 2020. Association for Computational Linguistics.

[42] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*, 2016.

[43] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc., 2019.

[44] Dian Yu and Zhou Yu. Midas: A dialog act annotation scheme for open domain human machine spoken conversations. *arXiv preprint arXiv:1908.10023*, 2019.

[45] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*, 2020.

[46] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93, 2020.

[47] Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127, 2018.

A Knowledge-Grounded Response Generation Example

<table>
<tr><td>Target response from the dataset</td><td>Fine-tuned Blender90M that uses the first knowledge sentence</td><td>Fine-tuned Blender90M that uses three knowledge sentences</td></tr>
<tr><td colspan="3">Knowledge:
There is a 274-line poem featuring roughly 800 irregularly pronounced English words called The Chaos.
The real-life soap opera behind the publication of Emily Dickinson’s poems.
So did critic Thomas Wentworth Higginson, with whom Emily had corresponded: despite many arresting phrases, he pointed out, nothing scanned or rhymed properly.
Utterance: Bringing back so many memories, i think i might listen to that soon. Demetri Martin wrote a 224 word palindrome poem.</td></tr>
<tr><td>Funny! You really can do so much with poetry! There’s also a poem that’s 274 lines, and features about 800 irregularly pronounced English words. It’s called The Chaos. I’d like hear that one read out loud!</td><td>That is probably an episode of the show. The rest of the show is not so good, i think. The episodes without the chaos, are called a chaos.</td><td>Yep. I didn’t know that. Good, I have a thing for giving back. Take care!</td></tr>
<tr><td colspan="3">Knowledge:
One of the first typewriters was initially called a «literary piano».
When Mabel visited their home to play the piano, she occasionally glimpsed a flitting swirl of white in the next room, sometimes even received a note of thanks for calling, but she never actually spoke with the reclusive, almost spectral Emily.
There is a 274-line poem featuring roughly 800 irregularly pronounced English words called The Chaos.
Utterance: Yeah that seems like the right name for it. Do you listen to or play the piano?</td></tr>
<tr><td>Funny you should ask! I was just going to ask if you knew that they used to call typewriters «literary pianos»! Isn’t that funny? I like to bang on the keyboard every now and then!</td><td>Yeah , I love piano! One of the first typewriters was called a «literary piano.» So I kind of feel sorry for piano owners.</td><td>No, but I learned to play piano. I like to play the piano, but not the flute. How about you?</td></tr>
<tr><td colspan="3">Knowledge:
Steam pianos used to exist, and they could be audible for miles.
When Mabel visited their home to play the piano, she occasionally glimpsed a flitting swirl of white in the next room, sometimes even received a note of thanks for calling, but she never actually spoke with the reclusive, almost spectral Emily.
There is a 274-line poem featuring roughly 800 irregularly pronounced English words called The Chaos.
Utterance: That’s all I do is bang on them while missing the keys I’m aiming for, lol! Nice chatting with you.</td></tr>
<tr><td>Good thing you’re not playing on an old steam piano! They’d hear you for miles around! Goodbye!</td><td>Oh wow, I think I’m missing the point. Thank you! I love chatting with you, and it’s great chatting with you as well! Have a nice day!</td><td>Same! Do you have a steam pianos? I love them! Apparently, you can buy them online!</td></tr>
</table>

Table 8: Example conversation with real targets from the dataset and predictions of two fine-tuned models with different lengths of grounding knowledge. The text highlighted in bold shows the matches between grounding knowledge information and phrases in the predicted model responses.

B Example of dialogue with `Wiki Skill` small talk script

Part from Art scenario, Photography subtopic:
B: Art is a good way to express feelings. Would you like to talk about art? [*An introductory phrase which suggests talking about art*]
U: yes, I like **photography**
B: In our increasingly busy lives it's difficult to always be in the moment. Taking pictures helps you to hang on to those memories a little longer. Do you have any funny photos of your family or pets? [*in our art scenario we have three branches for drawing, photography and memes and we move to photography branch after matching of the corresponding pattern in the user utterance*]
U: yes
B: I can tell you some tips about photography. Would you like to know how to take picture of pets?
U: ok
B: Letting your pet get settled. Hold the camera out to the animal for a good sniff. Anytime an animal sees something new, it's going to be curious, especially cats. Would you like to learn more? [*We checked affirmative user answer and began to give the steps from the wikiHow page "Photograph-Pets"*]
U: no [*We stopped giving content from the wikiHow page because the user gave a negative answer to the question "Would you like to learn more". If the user had given the affirmative answer, we would have continued to tell the next paragraphs from the wikiHow page*]
B: Do you take photos on an SLR camera or on your cell phone?
U: on a **camera**
B: SLR camera gives better quality of photos. [*Acknowledgement after extraction of "camera" pattern from the user utterance, another acknowledgement could be if the user answered "phone" or "mobile"*] Do you share your photos in Instagram or Flickr?

Part from music scenario:
B: Which music do you like?
U: ***scorpions*** [*We extract the band "Scorpions", and switch to the branch of the scenario for discussion of bands*]
B: I like ***Scorpions*** too. I like their songs ***Wind of Changes*** and ***Fly to the Rainbow***. What is your favourite song of ***Scorpions***? [*We fill in the slots the extracted band. Also we extract from Wikidata the triplets (<song_title, performer, Scorpions>) and fill in the slots the songs titles.*]

Part of Art scenario, Painting subtopic
B: I like Shishkin paintings. Pictures of what painters do you like? [*We expect the next user utterance to contain the entity with Wikidata triplet <entity, P106, Q1028181> (P106 is the relation "occupation", "Q1028181" is "painter")*]
U: ***picasso***
B: Would you like to know about style and technique of Picasso? [*We extracted from the base a Wikipedia page about Picasso using the correspondence between Wikidata entities and Wikipedia pages and suggested the user one of the headings from the page*]

Figure 6: Examples of `Wiki Skill` utterances. Bold - which words were used for checking of conditions, bold italics - extracted entities and filled slots, bold underlined - facts from Wikipedia and wikiHow. The dialogue is not with a real user.

C Parsing of Wikipedia pages

Entity type	Wikidata types	Wikipedia page headings
animal	Q16521, Q55983715	distribution, relationship with humans, behavior, in popular culture
athlete	Q2066131, Q18536342	club career, international career, player profile, records
team	Q20639856, Q847017	support, stadium, colors and mascot, club rivalries, records
musician	Q488205, Q36834, Q177220, Q753110	compositional style, musical style, vocal style, music career
band	Q215380, Q105756498	early years, breakthrough success, band split-up, new line-up, musical style, development
author	Q36180, Q49757, Q214917, Q6625963, Q28389	fictional works, critics by other authors, life and career
book	Q571, Q277759, Q8261, Q47461344, Q7725634, Q1667921	composition history, principal characters, background, film
game	Q7889	game modes, multiplayer, customization, films, virtual reality
film	Q11424, Q29168811, Q24869, Q202866	plot, production, development, filming locations, music, casting, special effects

Table 9: Examples of headings for paragraphs from Wikipedia pages for different entity types, extracted with `Fact Retrieval`.

D DREAM Socialbot Components

All used in DREAM 2 components are presented in Figure 1. All components not described in this Appendix have not been changed from the original DREAM [23].

D.1 Annotators

D.1.1 User Input Annotators

All annotators except `ASR Processor` accept raw ASR texts composed by ASR hypotheses with the highest probabilities.

SpellCheck is a pattern-based component to rewrite different colloquial expressions to a more formal style of conversation. All components in the pipeline accept preprocessed user utterances.

Combined classifier is a BERT-based model, that was built on top of the following models: `Custom CoBot Topics classifier`, `Custom CoBot DialogAct Topics classifier`,`Custom CoBot DialogAct Intents classifier`, `Sentiment Classifier`, `Emotion Classifier` and `Toxic Classifier`. Specifically, we utilized the dialogue data from the Alexa Prize Challenge 3 with annotations by all six models (CoBot models as API) and trained the BERT-base model on these annotations (without history), treating them as «gold» labels. All labels were considered to be independent from each other.

MIDAS Classifier is a BERT-based model trained on a semantic classes subset of MIDAS dataset [44]. This classifier takes as an input sentence with previous socialbot utterance and returns probabilities of different semantic dialogue acts according to the MIDAS annotation scheme.

CoBot Entities is built as API service on top of the Amazon Conversational socialbot Toolkit (CoBot) [21]. `CoBot Entities` returns list of detected entities labelled with types, e.g. «person», «videoname», «sport», «misc».

Entity Linking finds Wikidata entity ids for the entities detected with `CoBot Entities` annotator. For each entity substring, candidate entities are ranked using Wikidata entity descriptions to find the entity which is the best fit for the context.

Wiki Parser extracts Wikidata triplets for the entities detected with `Entity Linking`.

Fact Retrieval extracts facts from Wikipedia and wikiHow, which are used in topic `DFF`-based skills, in `Knowledge Grounding Skill`, and in `Text QA` for answering factoid questions. The annotator has a set of Wikidata entity types for each topic (for example, «athlete» and «team» for the topic «Sport») and extracts facts from Wikipedia pages for these entities, marked with the corresponding headings (for example, for an athlete the annotator will look for Wikipedia paragraphs with the headings «club career», «international career», «player profile», etc. and return the list of paragraphs with these headings). These annotations are made so that a topic skill could use a fact of a specific subtopic, for example, tell about the club career of an athlete.

KBQA answers user's factoid questions based on Wikidata KB. The annotator takes as input entity ids, detected with `Entity Linking`, extracts triplets from Wikidata, which contain the entity, and ranks these triplets to find the one which gives the answer to the question.

Text QA answers user's factoid questions using textual facts extracted with `Fact Retrieval`. The service uses the model which detected the spans of the answer in the text and gives as output the sentence which contains the answer.

Entity Storer is a rule-based component, which extracts from the user's and socialbot's utterances entities if opinion expression is detected with patterns or MIDAS Classifier and saves them along with the detected attitude to dialogue state.

Speech Function Classifier is a hierarchical algorithm based on several linear models and a rule-based approach for the prediction of speech functions described by Eggins and Slade. Classifier takes the user's current and previous utterances as an input and outputs one label, i.e. a speech function, for a current utterance.

Speech Function Predictor yields probabilities of speech functions that can follow a speech function predicted by Speech Function Classifier.

Personality Detection uses a Big5-based mechanism to identify the psychometric profile of the user.

D.1.2 Candidate and Response Annotators

Response Candidate Annotators include `Combined Classifier`, `MIDAS Classifier`, `CoBot NounPhrases`, `Blacklist Words Detector`, `Speech Function Classifier` and `Speech Function Predictor` described in D.1.1 as well as CoBot `Conversation Evaluator` and `Hypotheses Scorer`.

As soon as the final response has been selected by `Response Selector`, we further process it with `Sentence Segmentation`, `NER`, `CoBot NounPhrases` and `Sentence Rewriting Response Annotators`. The final response annotations allow us to work with the outputs from the heterogeneous skills such as template-based ones with punctuation, retrieval, or generative skills in the same way.

D.2 Skill Selector

Skill Selector has few changes from [23]. The clue difference is the constriction of sensitive mode cases. Now the sensitive mode is used for personal questions on restricted topics and is not used for user utterances with obscene language.

D.3 Conversational Skills

D.3.1 Linking Skills

Appropriate transitions from one skill to another create the smooth user experience. Skills can add templated triggers to enable other skills on the next dialogue turn. When the script is short, or the user declines offered topics, the dialogue could be seen as incoherent because the socialbot quickly changes the topic. Therefore, we also added a special list of connection phrases between different topics, which are covered by scripted skills. For almost all pairs of topics, we have a variety of citations, interesting facts, or thoughts that are related to both topics simultaneously. For every topic, we also have a list of interesting previews to be used in case of not presented connection phrases for the previous and current topics or in case of no recently discussed scripted topic.

D.3.2 Template-based skills

Movie Skill is implemented using `Dialogue Flow Framework` and takes care of the conversations related to movies. It is inherited from the previous `Movie Skill` version and almost repeats the structure of the conversation. In DREAM 2 we use another way to extract movie title from user utterance based on `CoBot Entities`. We collected a list of movies with high rating and low enough number of reviews, so `Movie Skill` is able to recommend movies including of specific genres.

Book skill detects book titles and authors mentioned in the user's utterance with the help of the `Wiki parser` and `Entity linking` and recommends books by leveraging information from the GoodReads database[5]. The skill can discuss genres of books, their storylines, and their publication years. It can also suggest books by author or by genre to the user. Apart from that, this skill has several other scripted lines of dialogue. Overall, the skill is inherited from the previous `Book skill` version, but now it uses different sources of information (WikiData instead of Evi), which allows for broader and more interesting dialogues.

Small Talk Skill asks questions using the hand-written scripts for 25 topics, including but not limited to love, sports, work, pets, etc. Most of the topics are now covered by specific scripted skills or by `Wiki Skill`, so the `Small Talk Skill` can be considered as a simple dummy for cases of failing scripted skills.

Food Skill is constructed with `Dialogue Flow Framework` to encourage food-related conversation. It can talk about interesting food facts and discuss different world cuisines, healthy meals that are easy to cook, and favorite food.

Generic Responses Skill, a yet another `Dialogue Flow Framework` skill, is designed to support extrovert users' desire to talk with the socialbot in a dominant fashion. It utilizes `Personality`

[5]`https://www.goodreads.com/`

Detector to identify extrovert users, then uses Speech Function Classifier to identify utterances with the Speech Functions that can get generic responses as the reply, and then provides these generic responses back to the user.

Gossip Skill is implemented with Dialogue Flow Framework to encourage conversation about celebrities. It was our early exploration in the Gossip conversation genre modeling. Initially designed around Speech Functions, it was developed without them given that our Speech Function Classifier was going through the second iteration of development after the first one didn't provide acceptable accuracy levels. Based on the output of News Api Annotator and Wiki Parser, this skill can discuss different celebrities: their basic and non-basic occupations, news about them, their creative works, and their sports teams. This skill also introduced a basic mechanism for reflecting and maintaining the socialbot's opinion towards the discussed entities as well as remembering the user's opinion towards them. This mechanism was later expanded and introduced as the Entity Storer. Discussion of non-basic occupations was previously included in Celebrity Skill.

Bot Persona Skill aims to discuss user favorites and 20 most popular things with short stories expressing the socialbot's opinion towards them.

Animals Skill is created using Dialogue Flow Framework and has three branches of conversation about animals: user's pets, pets of the socialbot, and wild animals. The script about the user's pet asks about the name and breed of the user's pet, asks how the user plays with his pet, whether the user loves his pet etc. The script about pets of the socialbot tells about a cat or a dog and asks the user's opinion. The script about wild animals extracts the animal entity from the user utterance, asks some questions about it, and then tells facts about the animal.

Wiki Skill is created using Dialogue Flow Framework. The skill is used for making scenarios with the extraction of entities, slot filling, facts insertion, and acknowledgments.

- **Anime** script shares the information about popular animations and offers the user to learn how to «Make-an-Anime» from wikiHow.
- **Art** talks with the user about drawing, photography or about memes. The scripts use Wikipedia and wikiHow facts. Drawing script can suggest the user tips on how to improve drawing skills based on the wikiHow article The script also can tell facts about the user's favourite painter.
- **Bitcoin** script shares information how to mine and buy bitcoins from wikiHow pages «Mine-Bitcoin» and «Buy-Bitcoins».
- **Cars** script asks the user which he has a car, then asks different questions and comments on the user's answers. The script suggests the user the tips from wikiHow about reducing the cost of car maintenance and how to keep warm in a car in cold weather.
- **Chill** script asks the user how he spent his free time (listened to music, played games) and has links to music and gaming skills if the user mentions one of these activities.
- **Dinosaurs** script tells the user about pre-scientific history, early dinosaur research, and discoveries of dinosaurs based on Wikipedia content.
- **Family** script comments on the first user utterance where one of the family keywords was mentioned based on detection of different patterns . For example, if the user said that he played with his brother, the script asks what games did he play) and then asks some general questions about the user's family, followed by acknowledgments.
- **Friends** script recommends how to «Make-a-Friend-Laugh» and «Maintain-a-Friendship» to users who have friends, and shares info how to «Make-Friends» for users who have no friends. It also asks user questions about his friends.
- **Harry Potter** script asks the user different funny questions based on Harry Potter films, for example, «Do you think that using magic outside of Hogwarts is fine?», suggests the user tips for making a potion from Severus Snape's lab etc.
- **Hiking** script can advise the user on how to «Choose-a-Hiking-Vacation-Destination» and «Choose-a-Good-Hiking-Dog» using wikiHow.
- **Hobby** is aimed to talk about user hobbies. The skill suggests ways to find a hobby from wikiHow if the user does not have one, discusses popular life hacks and tells how to keep hobby costs down.

- **Love** script discusses relationships with the user using several wikiHow pages. If the user is in relationships, the script suggests how to be more romantic. If the user is not in relationships but loves someone, the script tells how to catch the crush's attention and make someone fall in love. Otherwise, the script offers advice on how to find love.
- **Politics** script helps interested in politics users to understand politics itself and how to discuss it with other people. For those not interested in politics, the script suggests how to friendly avoid talking about politics.
- **School** script is turned on if the user answers «school», «study» or «homework» to the question «What do you do on weekdays» or mentions these keywords. The script contains questions about different aspects of the user's school life (favorite subject, school sports activities) and suggests trying several pranks on teachers and classmates.
- **Sleep** script tells the user different tips for better sleeping, for example, listening to sounds of the rain or relaxing music.
- **Space** script uses parsed Wikipedia page about space exploration to tell about first outer space flights, space station, and future of space exploration.
- **Smartphones** script asks the user about his smartphone OS (IOS or Android) and tells tips from wikiHow pages how to speed up an Android smartphone or transfer files from iPhone to iPad.
- **Robots** script is based on Wikipedia pages «Robot» and «Unmanned aerial vehicle» and also suggests the user building a simple robot from the wikiHow page.
- **TikTok** script tells the user how to become popular in TikTok (from wikiHow page).
- **Work** script is turned on of the user answers «work» to the question «What do you do on weekdays». The script contains several questions about the user's job (his occupation, how he relaxes after work etc.).

D.3.3 Template-based Skills with External Services

News Skill presents the top-rated latest news about entities or topics using the GNews API[6]. The skill supports the functionality from DREAM 1.0. This skill also offers news about extracted entities in a high-confident manner if the user asked to talk about this entity and low-confident manner if the user just mentioned the entity.

Gaming Skill also provides video games discussion. While `Game Skill` focuses mainly on game charts, `Gaming Skill` is for more general talk about video games. The skill collects information about video games via IGDB API[7]. Since the API responds not fast enough, we had to store locally information about approximately 150 most popular video games. Apart from general comments about games, `Gaming Skill` can lead specialized discussion about video game Minecraft[8]. Unlike `Game Skill`, `Gaming Skill` was created using `Dialogue Flow Framework`.

D.3.4 Generative Skills

Knowledge Grounding Skill generates a response based on the dialogue history and provided knowledge related to the current conversation topic. It uses a ParlAI Blender 90M model fine-tuned on the Topical Chat Enriched dataset.

Wikidata Dial Skill generates an utterance using Wikidata triplets. The skill extracts triplets which contain an entity from the user utterance. The BERT-based ranker finds the most relevant triplet, DialoGPT takes as input this triplet, dialogue history and generates the utterance. The models were trained on OpenDialKG dataset.

D.4 Response Selector

`Response Selector` is a DREAM agent component that makes the final decision about the content of the response to be surfaced to the user. `Response Selector` reads from the `Dialogue State`

[6] `https://gnews.io`
[7] `https://www.igdb.com/api`
[8] `https://www.minecraft.net/en-us`

candidate responses generated by the active conversational skills and annotated by the `Response Annotators`. `Response Selector` is not restricted to select the final response only from the response candidates but can also generate a final response as a combination of available candidate responses.

E Dialogue Flow Framework

This section contains an example of visualization of a skill based on DFF, the visualization is built automatically and helps to visualize the dialogue graph, which can greatly simplify the development.

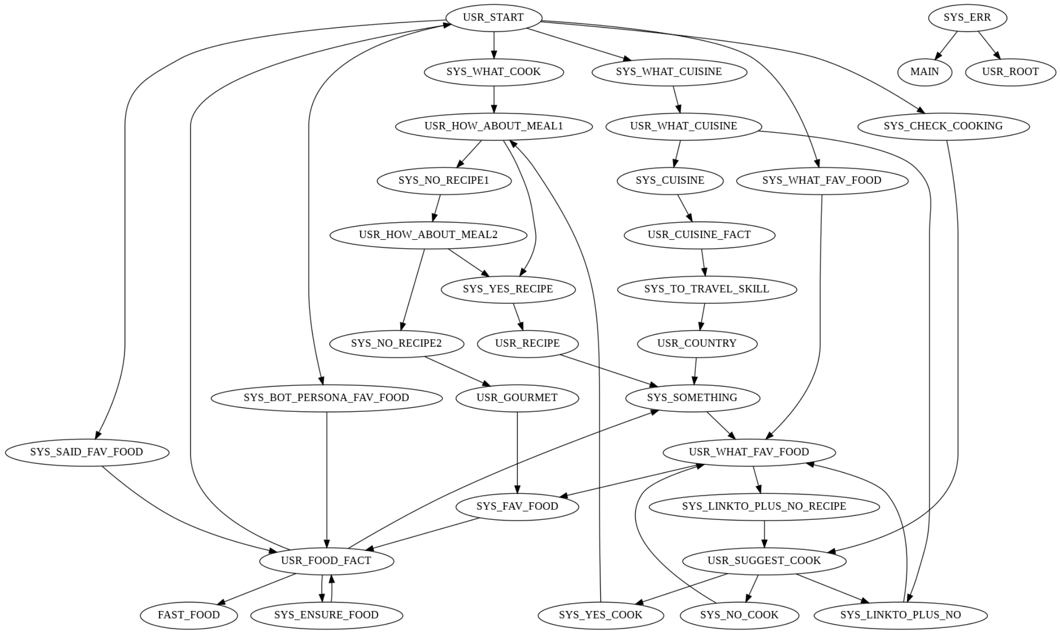

Figure 7: Visualisation of `DFF Food Skill` scenario. The graph shows the nodes that the user passes through during the dialogue. The transition checks the condition and returns the corresponding response.

F Topic Recommendation Results

Topic	F1-score, %			Positive Samples	Negative Samples	All Samples
	«let's chat»	link questions	«let's chat» & link questions			
Book	39.8	50.7	54.2	171	144	315
Movie	49.4	54.7	54.4	334	244	578
Animals	62.6	62.0	61.2	609	223	832
Food	28.1	54.1	48.8	318	195	513
Travels	25.8	52.2	50.7	286	187	473
News	44.8	41.5	49.1	212	224	436
Sports	45.9	40.0	42.2	223	264	487
Music	33.7	46.7	45.8	271	159	430
Games	37.1	51.9	48.2	346	245	591

Table 10: F1-weighted scores of the ConveRT model predictions for different topics and for different responses sets. The evaluation is conducted on the dialogues with real users, negative samples are generated by random assignment of predicted topic. The number of samples with positive and negative labels can be found in corresponding columns. The ConveRT model is used as pre-trained without any fine-tuning.

Topic	F1-score, %		Positive Samples	Negative Samples	All Samples
	Keywords	**BART**			
Book	37.1	45.9	111	51	162
Movie	49.4	48.3	106	57	163
Animals	45.3	47.6	138	63	201
Food	34.9	40.1	176	80	256
Travels	39.9	50.1	177	83	260
News	47.5	47.0	120	126	246
Sports	49.8	47.8	106	135	241
Music	46.8	46.5	16	8	24
Games	34.5	41.1	221	129	350
Science	47.8	49.1	195	141	336
Gossips	51.1	55.8	222	145	367

Table 11: F1-weighted scores of the TF-IDF Model for different topics and for different methods of subreddits classification. The number of samples with positive and negative labels can be found in respective columns.