# Alquist 3.0: Alexa Prize Bot Using Conversational Knowledge Graph

**Jan Pichl, Petr Marek, Jakub Konrád, Petr Lorenc, Van Duy Ta**
Faculty of Electrical Engineering, CTU Prague
Prague, Czech Republic
`{pichljan, marekp17, konrajak, lorenpe2, tavanduy}@fel.cvut.cz`


**Jan Šedivý**
CIIRC, CTU Prague
Prague, Czech Republic
`jan.sedivy@cvut.cz`

## Abstract

The third version of the open-domain dialogue system Alquist developed within the Alexa Prize 2020 competition is designed to conduct coherent and engaging conversations on popular topics. The main novel contribution is the introduction of a system leveraging an innovative approach based on a conversational knowledge graph and adjacency pairs. The conversational knowledge graph allows the system to utilize knowledge expressed during the dialogue in consequent turns and across conversations. Dialogue adjacency pairs divide the conversation into small conversational structures, which can be combined and allow the system to react to a wide range of user inputs flexibly. We discuss and describe Alquist's pipeline, data acquisition and processing, dialogue manager, NLG, knowledge aggregation, and a hierarchy of adjacency pairs. We present the experimental results of the individual parts of the system.

## 1 Introduction

This paper describes the third version of the socialbot "Alquist", a conversational system designed to converse coherently and engagingly with humans on popular topics.

To provide users with an informative and fluent experience, we enhance a novel approach based on creating the dialogue using a conversational knowledge graph. This allows Alquist to work with smaller template-based sub-dialogues, and be more reactive to the broad range of user responses present in natural dialogue. The conversational knowledge graph also allows Alquist to store the knowledge expressed during the conversation and utilize it in following interactions with the user. The system is more modular and provides a unique experience in each session. It can have a seamless conversation on popular topics like pop culture, sport, or technology. We also made a significant increase in the performance of our NLU module using new state-of-the-art technologies, and focused on joining more models into one to maintain accuracy and decrease memory requirements. Additionally, Alquist 3.0 can be enhanced with the approach presented in the paper about the second version of Alquist [11]. Such a combination offers more vibrant conversations to the users.

The Alquist socialbot builds on the experience and knowledge gained from Amazon Alexa Prize '17 and '18. Previous results showed that users could have a meaningful conversation with a socialbot, but it needs to be more reactive and accurate in the answer. These facts motivated the changes which we introduced into the third version of the socialbot Alquist.

The structure of this paper is divided into several sections. First of all, we focus on our novel approach based on the knowledge graph. After this, we dedicate a section to the overall system architecture, highlighting essential parts in subsections. Finally, we provide example conversations, we describe the experiments we performed during the creation of the system, and we summarize our findings.

## 2 Motivation

Let us start by explaining why we decided to use the architecture described later in the paper. The architecture of the system has always been designed to address two contradictory requirements. The first requirement is to be in control of the dialogue flow during the development process. The second one is to create such dialogues that can respond to various and unexpected user inputs.

The main part of the first version of our system [12] developed during the Alexa Prize 2017 was created using a complex dialogue structure, which we called "dialogue automata". These structures were supposed to deliver engaging conversations thanks to a rich dialogue structure, which can present all the relevant information to the users as they traverse through it. While this approach works very well for cooperative users as the structure guides them through the conversation, it can be easily broken by the users asking additional questions or trying to change the direction of the dialogue. Additionally, many of the dialogue nodes could remain unvisited as the unexpected input messages did not let the bot finish the dialogue.

To solve the problems mentioned above, we introduced the second version of the system [11] during the Alexa Prize 2018. We introduced smaller structures called sub-dialogues, which can be chained dynamically, allowing the users to have different dialogue experiences. Each of the sub-dialogues has its Dialogue Management model, and the high-level decision of which one should be triggered is made by a combination of the results of the Topic Switch module and the Intent module. This new approach reduced the problem, yet it was far from a complete solution.

We run analyses of the errors produced by the system. We manually went through 100 randomly selected conversations and marked the turns which we found incoherent (the engagement of the system was not evaluated). Note that the evaluation was done on a subjective basis. For each error turn, we marked which of the system components caused the erroneous response (some turns can contain multiple errors). The distribution of the errors is shown in Table 1. An error is marked only for those turns that produced a response violating the coherence of the dialogue. There can be, for example, a turn with a punctuation error but the responses are still generated correctly—such a turn is not marked as an error.

Table 1: Error distribution in the second version of Alquist. The analysis was done manually on 100 conversations. Some of the analyzed turns contained multiple errors, the most typical combination was an error in the Topic Switch module and in the Intent module. Conversation examples are synthesized by our team based on observed patterns in the data.

|  | Occurrence | Example |
|---|---|---|
| Topic Switch | 44 % | **Bot:** ... How is it going?<br>**User:** it's going good what happened today in history<br>**Bot:** Well, I'm always busy as people keep chatting with me. |
| Intent | 28 % | **User:** how old are you (intent: how_are_you) |
| Punctuation | 19 % | **User:** good I'm <SEP> good you're really good |
| Dialogue Act | 6 % | **User:** who's boyfriend (DA: Other, correct Wh-question) |
| Dialogue Manager | 4 % | **Bot:** ... Who do I have the pleasure of speaking with?<br>**User:** why are fire trucks red<br>**Bot:** Hey Red, I'm happy to meet you! |

In pursuit of mitigating the errors above, we came up with the third version of our system, which we describe in this paper. We present a concept of adjacency pairs that can be chained flexibly. Thanks to the fact that dialogues built using adjacency pairs are short, we do not need the Topic Switch component, which caused most of the errors. Moreover, we use a conversational knowledge graph that stores the information available for use as well as the information expressed by the users in previous

sessions. The previous versions of Alquist were also able to store users' preferences; however, only in predetermined parts of dialogues. With the new approach, the system can remember facts expressed at any point. The utilization of the knowledge graph allows us to remember a significantly large number of facts extracted during the dialogue, which can be subsequently used in the interaction by Alquist, or asked about by users.

# 3  System architecture

The Alquist 3.0 system architecture is conceptually similar to the architecture of previous versions of the system. It is based on the Knowledge Graph and the main component Pipeline, to which additional micro-services are connected.

## 3.1  Conversational Knowledge Graph

To be able to handle more complex interactions with the user, it is necessary to have access to real-world knowledge and objective factual information as well as personalized (subjective) knowledge, such as opinions, preferences, and others. To represent the knowledge and opinions of our system about the world, we have created an RDF graph database where we used a dump of the Wikidata database[1] as its base.

We have also designed a custom ontology that is to be partially mapped to the ontology used by Wikidata. Our ontology is inspired by other common ontologies such as Schema.org[2] and EVI. However, we expanded it with objects representing the user and the bot. We also added properties modeling personalized relationships between the user or bot and various other objects in the database. Examples of such properties are *likes, hates, hasFavorite, hasOpinion*.

As shown in Figure 1, we also created custom annotation properties in the ontology (e.g. *doYou*). These properties are tied to object properties. Their range is commonly a string (e.g. *"Do DOMAIN like RANGE"*), and they represent delexicalized examples of sentence structures that represent the domain object property. This allows us to do two things. We can automatically generate and update datasets for dialogue acts detection directly from the knowledge base (KB). Furthermore, we can use the sentence structures assigned to properties in adjacency pairs templates to generate system responses regarding relationships between objects in the database.
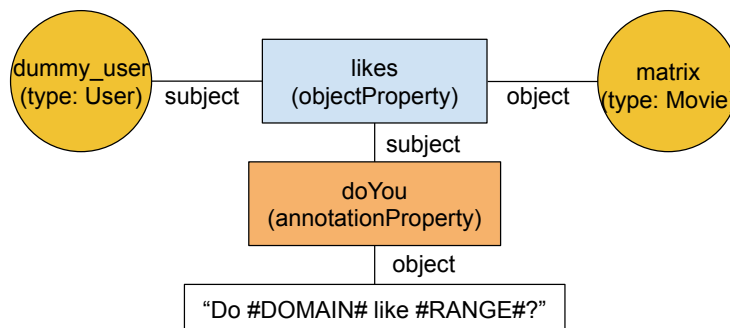


Figure 1: Illustration of sentence structure examples as annotation properties

Additional examples of sentence structures, such as:

- *ProvideInformation_Negative* - #DOM#'s name is not #RAN#.

- *OpenQuestion_Object_Positive* - What is #DOM#'s name?

- *YesNoQuestion_Positive* - Is #DOM#'s name #RAN#?

---

[1]https://www.wikidata.org/wiki/Wikidata:Database_download
[2]https://schema.org/

The new approach to representing knowledge allows the system to create a profile of the user directly in the knowledge database, based on the information the user shares with the system. The system is then able to reference the profile during the conversation, and this leads to more variability and higher personalization. However, this personalized engagement needs to work in both directions – and so the profile of the bot is represented in the knowledge base in the same way. The database contains a representation of the bot, complete with its preferences, personal profile, likes, and dislikes. The profile is adjusted and anonymized for the purposes of the competition, but the system can draw information from it in real-time and include it in the conversation.

## 3.2 Pipeline

The fundamental part of the system responsible for message processing is the Pipeline, which handles communication between the Natural Language Understanding (NLU), Knowledge Graph, Dialogue Management, and Natural Language Generation (NLG) components during the process of response creation.

The input to the pipeline is the message with the highest confidence score produced by the Alexa ASR. The message is sent to the NLU component, which segments the message. A segment is a part of the message representing one semantic unit. The NLU component also provides entity and property annotations for each segment (see section 3.3). Alquist generates a set of actions for every segment, based on the NLU annotations in the next step (see section 3.4). Dialogue management creates a response for each action based on the adjacency pair, which is assigned to the action (see section 3.5). The NLG component adds entities extracted from the knowledge graph into the responses and converts them into syntactically correct forms (see section 3.6). After this step, Alquist selects one action for each segment, which will produce a response based on NLU confidence scores and additional criteria (see section 3.7). Additionally, to have a more engaging and coherent conversation with the user, the system also enhances the response by a fun fact and a question based on the annotations of the response (see section 3.8). After this step, the final response is presented to the user.

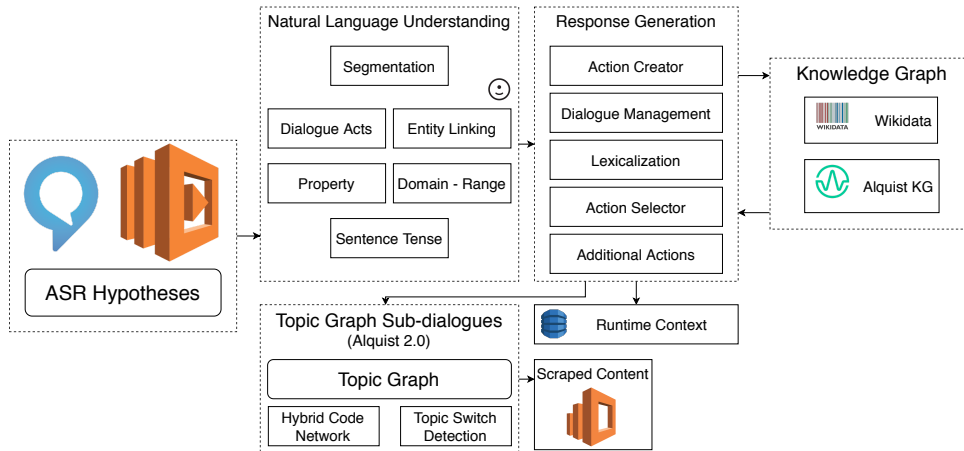We summarize the pipeline in Figure 2.



Figure 2: The system architecture schema

## 3.3 Natural Language Understanding

The Natural Language Understanding component is a set of analysis machine learning models producing the annotations essential for further processing in the pipeline. These models can be triggered in an arbitrary order. An exception is the sentence segmentation, which needs to be triggered first, as the remaining models annotated each segment separately. The user message is processed as follows. The input for the NLU consists of $n$ speech recognition hypotheses, each hypothesis is processed with the segmentation model first, and each segment from a hypothesis is used as the input for the additional models.

### 3.3.1 Sentence Segmentation

The goal of this model is to split the sentences into clauses. The standard way how to tackle this task is to decide based on punctuation marks. However, our input consists of speech hypotheses without punctuation. Therefore, we start by using punctuation restoration, and afterward, we split the sentences based on the restored punctuation marks.

Compared to the former Alquist system where we used a bidirectional neural network (TBRNN [15]), we decided to use a BERT based model [4]. Our model consists of a pre-trained BertForMaskedLM layer and a linear layer that predicts the punctuation mark after the middle word of a fixed size input sequence.

Since the model predicts only the punctuation after the word in the middle, we need to add padding to the beginning and the end of the input speech hypothesis. Afterward, we feed the model with sequences of a fixed segment size based on the moving window, and the model returns one of the possible punctuation characters for each word in the original speech hypothesis.

The results of the new model are shown in Table 2. We can see a significant improvement in all metrics compared to the older model. Moreover, since the BERT model is more computationally demanding, we also analyzed the performance of various BERT distillations [13, 10]. Based on the observed results, we can conclude that all tested models outperform the former bidirectional one.

Table 2: Results of the experiments on IWSLT 2012 English dataset.

| Model | Comma | | | Period | | | Question | | | Overall | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pr. | Re. | $F_1$ | Pr. | Re. | $F_1$ | Pr. | Re. | $F_1$ | Pr. | Re. | $F_1$ |
| TBRNN [15] | 65.5 | 47.1 | 54.8 | 73.3 | 72.5 | 72.9 | 70.7 | 63.0 | 66.7 | 70.0 | 59.0 | 64.4 |
| BERT [4] | **74.9** | **66.8** | **70.7** | **81.8** | 85.5 | **83.6** | 66.7 | **73.9** | **70.1** | 74.4 | **75.4** | **74.8** |
| DISTILBERT [13] | 70.3 | 58.6 | 63.9 | 81.4 | 76.3 | 78.8 | **78.4** | 63.0 | 69.9 | **76.7** | 66.0 | 70.9 |
| ALBERT [10] | 84.0 | 44.1 | 57.8 | 77.0 | **87.6** | 82.0 | 64.6 | 67.4 | 66.0 | 75.2 | 66.4 | 68.6 |

### 3.3.2 Dialogue Act detection

Since one of the key concepts in the pipeline is the processing of adjacency pairs (as described later in this paper), we need to identify the dialogue act in the message of a user. We created a hierarchical structure of the dialogue acts. We treat this problem as a simple text classification task where each class is defined by the combination of the tags on the path from the tree root to a leaf node. You can find a description of the classes in Appendix A. We defined in total 37 leaf classes, each of the class can be mapped to one or more dialogue act class presented in the Switchboard dataset[3] [6].

The data for the dialogue act detection model were created manually through a template mechanism which we have been using since the previous version of our system. The created data are stored in the KG. We adopt the model architecture which we have been using for intent detection and dialogue act detection in previous version of Alquist [11], which is inspired by [9].

### 3.3.3 Entity Linking

As our new approach is based on working with a knowledge graph (KG), we need to identify each entity with its unique ID to be able to generate a proper response. This task is commonly referred to as entity linking.

Entity linking, as stated in [3], consists of three parts: mention detection (MD), candidate generation (CG) and entity disambiguation (ED). We tackle each problem as a separate unit to be able to leverage Cobot KG.

To generate mentions (text spans of potential entity occurrence), we used neural network based on bidirectional LSTM model, where the training set was manually labeled from a small subset of the user's messages. We test our model on the sequence labeling task, see the result in Table 3.

After retrieving the possible text spans, we combine the candidate generation with entity disambiguation. First, we have to decide if the entity is from our "private" knowledge graph or from the "general"

---

[3]`https://web.stanford.edu/~jurafsky/ws97/manual.august1.html`

knowledge graph (for example, Wikidata). We trained our neural network model to classify into two label types. If the model returns the label classifying the entity candidate as one from our "private" KG, then we have a set of manually designed rules to decide which possible entity should be assigned to that specific text span. The other case is that the candidate is from the "general" KG: then we query an Evi knowledge graph service to make the mention resolution. We collect all candidates for possible text span that the system returns, and query Evi knowledge graph service again to get their IDs (more specifically, Wikidata ID).

Further processing is coupled with property detection (see section 3.3.4), where we map the text span to two classes – domain or range. We have to match predictions from property detection and entity linking together to be able to decide which entity (linked to the knowledge graph) is the domain and which one is the range. Our further processing of user queries is principally dependent on these annotations (see 3.4).

Table 3: Results for sequence labeling

| Model | Dataset | Precision | Recall | F1 |
|---|---|---|---|---|
| Bi-LSTM | CoNLL 2003 | 90.0 | 89.4 | 89.8 |
| Bi-LSTM | Own manually labeled data | 93.6 | 92.0 | 92.6 |

### 3.3.4 Property detection

The knowledge graphs stores facts as triples, with each triple containing two entities and one relation. The user utterance can be mapped to this triplet – the entities are called domain and range; the relation between them is called property. To properly handle the response, we need to retrieve the property out of the user's message.

We tackle the problem of property detection as a multi-task model. These models are showing promising results on a task like joint slot-filling and intent classification [1].

Based on [16], we design our neural network with the novel approach based on the multi-task loss function. The model is trained on two tasks – sequence labeling and sentence classification.

To find the most suitable word embeddings on our proposed system, we used three types of word representation as a vector of floats:

- GloVe [8] - 300d
- Fasttext [2] - 300d
- Bert [4] - 768d - without fine-tuning

We measure our metrics on the publicly available ATIS dataset provided by Microsoft[4]. The data contain spoken utterances classified into one of 26 intents. Each token in a query utterance is aligned with IOB labels. Primarily, the dataset is used for intent recognition and slot filling, but it is sufficient for our testing purpose. The results are shown in the Table 4.

Table 4: Results on ATIS dataset

| Embeddings | Intent F1 accuracy | Slot-filling F1 accuracy |
|---|---|---|
| BERT | 92.45 | 95.78 |
| fasttext | 90.70 | 96.83 |
| GloVe | **95.74** | **98.44** |

From the result, we can see that GloVe embeddings outperform the others. Unfortunately, they have the disadvantage of the limited possibility of dealing with out-of-vocabulary words, so their usage is limited. We will do more research about fine-tuning the BERT, which also shows promising results.

---

[4]Available at github.com/Microsoft /CNTK/tree/master/Examples/LanguageUnderstanding/ATIS

To create a more robust model, we used the WebNLG dataset [5] as an additional source of data for training. It consists of a set of triples extracted from DBpedia, where the text is the verbalization of these triples. We mapped the properties to follow the Wikidata taxonomy.

The final model of property detection has two outputs. The sequence labeling output will classify each token into three categories – domain, range, or outside. The text classification output predicts the label of property (mapped to our KG) which is mentioned in the user's message.

### 3.3.5 Grammatical tense classification

Because the user is speaking naturally and we need to understand what is the time-span of the event which he might be talking about, therefore we classify an utterance into the correct tense. This sub-component uses the morphological structure of the word and part-of-speech tagging designed in spaCy [7]. SpaCy's implementation is based on rules and can predict the present and past tense. For the future tense, we develop a set of own rules.

### 3.4 Action Creator

Alquist has to react to the user's message appropriately. The input can be noisy either due to the ASR or because of the user over- or under-specifies the facts in the message (we observed over-specification of facts being a bigger problem for socialbots). For these reasons, sentence segmentation divides the input message into several segments, and each segment is processed by NLU independently. The NLU component outputs several classification hypotheses for each segment. This leaves the system with a large amount of NLU classification hypotheses, from which it has to choose the relevant ones for the creation of the response. We define *action* as a possible way, how to react on the user query, so the Action creator is the first component of the pipeline contributing to the process of response creation. Its task is to select all combinations of NLU classification hypotheses (entity types, properties, and dialogue acts), which are proposed for further processing.

There is only a finite number of combinations of NLU classifications meaningful for processing. The meaningful combinations are closely tied to the structure of the knowledge graph. We create actions only for those combinations of NLU classifications that contain an entity of a certain type, a certain property, and a certain dialogue act. For example, we create an action containing the entity type $chatbot$, the property $name$ and the dialogue act $open\ question$. This combination represents the question "What is your name?"

The Action Creator creates several actions for each sentence segment, yet it does not create actions that do not respond to the structure of the Knowledge Graph. This fact allows Alquist to take advantage of more than just the Top-1 NLU classification. Thus, it processes potentially noisy messages in a computationally feasible way.

### 3.5 Dialogue Management

Alquist has to decide what response it will produce for each action. This decision is made by the Dialogue Manager. The Dialogue Manager creates a response based on actions created by the Action Creator, and existing adjacency pairs. Adjacency pairs [14] are a linguistic concept according to which adjacent utterances in dialogue often form predetermined pairs. We utilize the concept of adjacency pairs in the form of simple dialogue structures, which are small and generic. Thanks to their size, adjacency pairs can be chained together and thus can flexibly react to unexpected input messages. Because of their generality, they can cover large parts of the knowledge graph with a small number of universal dialogue structures. An example of adjacency pairs handling open questions is shown in Figure 3.

It would be very time-consuming to create specific responses for all adjacency pairs. Often, the same adjacency pair will deal with very similar responses concerning different properties – in such situations, the responses usually differ in a single word. For this reason, we do not create the specific adjacency pairs directly. Instead, we introduce three levels which are utilized during the creation: a word level, a sentence structure level, and an adjacency pair level.

The first level operates with words. We selected a set of word categories, which describe the properties. The categories, for example, are Verb, Subject, or Pronoun. It is the task of the person working on certain properties to select the appropriate word for each category.
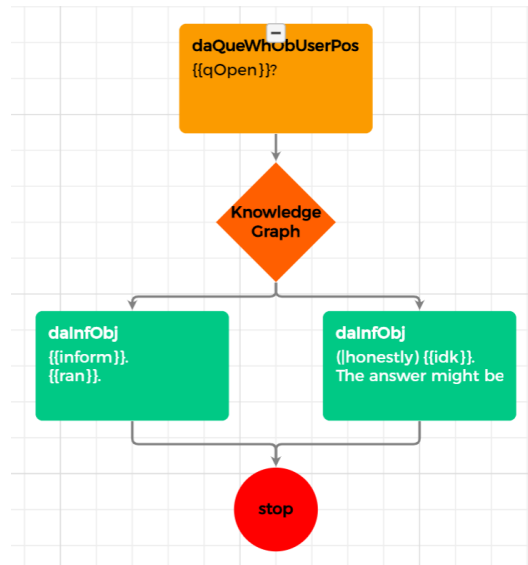
Figure 3: Adjacency pair representing an open question. The pair is started by the user with the *open question* dialogue act. The adjacency pair queries the knowledge graph first. Based on the result of the query, it informs about the result (in the left branch), or it apologizes for not knowing the answer (in the right branch).

The selected words are used as input for the level of sentence structures. Each structure is represented by a pattern of words. These patterns constitute sentences that express a certain meaning. The sentence structures can, for example, inform about the value of the property or ask about the value of a property.

The sentence structures are finally utilized in the adjacency pairs. An adjacency pair in our implementation consists of nodes and edges. The nodes represent the responses, and edges represent transitions between them. The response contains types of sentence structures, which are substituted by their real values depending on the property for which the adjacency pair is being utilized.

As shown, the word, sentence, and adjacency pair levels decrease the amount of work necessary for the design of responses.

## 3.6 Lexicalization

The knowledge graph contains a large amount of knowledge which relates to many different entities, individuals, or events happening at different points in time. If we want to describe such knowledge in natural language, we have to express all of these variables with the correct word morphology and sentence syntax.

However, it is intractable to create responses using all possible word forms and sentence syntaxes to encapsulate all entities or events contained in the knowledge graph. For this reason, Alquist uses the Lexicalization module, responsible for converting responses into the correct form.

The Lexicalization module has four tasks. It fills the values of entities in place of delexicalized tokens, it switches pronouns based on the gender of entities, it converts nouns into singular or plural form, and it converts the verb into the appropriate tense.

The response created by the dialogue manager contains delexicalized tokens $Domain$ and $Range$. We can see the example in the response *"DOMAIN loves to talk about RANGE"*. Actual entity values extracted from the knowledge graph replace $Domain$ and $Range$ tokens, creating the response *"KAREL ČAPEK loves to talk about ROBOTS"* for example. It often happens that the substituting entities are User or Alquist. In such cases, Lexicalization applies an additional rule: if the $Domain$ or $Range$ is equal to User or Alquist, the system replaces the delexicalized entities by pronouns $I$ or

*you.* The part of Lexicalitzation responsible for entity substitution allows Alquist to talk about any pair of entities using a single response.

When the same entity is referenced multiple times, it is natural to reference it by a pronoun and not its name. However, it is not trivial to select the correct pronoun because it depends on the gender and the grammatical number of the entity. This is the task of the second part of lexicalization. The first step determines the grammatical number of an entity based on the value stored in the knowledge graph. If the number is plural, Alquist uses pronoun $they$. If the number is singular, Alquist determines the gender of the entity from the knowledge graph. Based on the result, it selects the pronoun $he$, $she$, or $it$. The process of pronoun lexicalization is possible thanks to the utilization of the knowledge graph, which stores the grammatical number and the gender of entities.

The knowledge graph also stores the quantity of entities. An example could be that the user has $two$ sisters. The delexicalized response for such a fact looks like this: *"DOMAIN have RANGE sister."* The form of the noun sister depends on the value of a range. If the range is larger than one, the system creates a plural form of the word according to lexical rules. This part of lexicalization allows Alquist to talk about amounts stored in the knowledge graph.

Some of the knowledge stored in the knowledge graph or mentioned during the conversation can have a certain temporal relation. It also influences the response, mainly its verb. The fact that the user *went* to a university is an example of knowledge having a temporal relation. If Alquist wants to continue in a deeper conversation about such a fact, it has to take into account the temporal information. The user used past tense, meaning that he most probably does not study anymore. If this is true, the follow-up question *"What university do you study?"* is not natural. Instead, lexicalization works with the tense used in the user's message and changes the verb tense in the response accordingly, using lexical rules. Thanks to this lexicalization system, Alquist can talk about past, recent, and future events as well.

The main innovation of Alquist 3.0 is its ability to carry flexible conversation grounded by the facts stored in the knowledge graph. This puts a high demand on the syntax of the responses. The syntax needs to encapsulate relations between entities, their amounts, or the time of events. It is labor-wise challenging to create all syntactic variants for responses, considering the large number of all possibilities. Thus it is evident that the lexicalization module, which modifies the syntax dynamically, is essential for a socialbot of Alquist's scale.

## 3.7 Action Selector

Due to the expected noisy input message, the NLU part of the system segments the message into several parts, for which NLU produces several hypotheses of entity recognition and dialogue act detection. In the following step, the Action Creator creates actions out of these hypotheses. This leads to several actions for each segment. Dialogue Manager and Lexicalization discard some of the actions; however, Alquist has to select the most appropriate combination out of the rest. This is the task of the Action Selector.

The main criteria for the selection of actions are combined confidence scores of the NLU classification contained in action. The goal is to select one action for each segment, which maximizes the total score of all selected actions. These criteria select the actions that Alquist is the most confident of, given the noisy input message.

There are also additional constraints to the selection. The selected combination cannot contain actions that lead to the same adjacency pair. The combination also cannot include actions leading to multiple adjacency pairs whose responses consist of questions. The first constraint makes sense, given the desired naturalness of the conversation. It would be unnatural to repeat the same sentence multiple times during a single response. The second constraint improves the coherence of the conversation. We observed in the previous versions of the system [12, 11] that asking the user a single question leads to a more prolonged conversation. Thus Action Selector improves the naturalness and coherence of the conversation even when Alquist receives noisy input messages.

## 3.8 Generating additional actions

In previous sections, we described how we generate actions that directly respond to the user's message. We call this action type the *handle action*. However, in order to keep the conversation engaging and

exciting for the user, it is necessary to enrich it by injecting additional information, thoughts, and opinions, as well as prompting the user to continue the conversation with additional questions.

We define two additional types of actions that are generated for each segment of the user's message, *fun-fact actions* and *forward actions*. The fun-fact action is the part of the system's response that contains a statement with information related to the entities in the user's message. Forward actions are questions that are again related to the entities in the user's messages, and they ask users for more information or their opinions and feelings about the conversational topic.

Both types of actions are generated using the entities recognized in the segment and the Conversational Knowledge Graph. Properties and entities in the database had a popularity metric assigned to them, starting at 1 with specific properties and entities having their popularity increased manually to make their appearance more common during the conversation. The popularity is then further increased whenever the property or entity is mentioned in the conversation by the user or system. When generating the fun-fact and forward actions, the system generates a large number of possible actions. The popularity of each action is computed as the average of popularities of contained entities and properties. The action with the highest popularity is selected. This system is designed to prevent the bot from introducing conversational topics based on properties and entities that would not provide an exciting conversation.

## 4   Latency improvements

In addition to the coherence and engagement criteria, system latency is another important aspect. It is essential to keep latency below a reasonable level in order to maintain a seamless conversation experience. During development, the NLU component showed to be the most challenging in keeping its latency under the desired level. As we want to keep the overall latency under 500ms (on average), we needed to apply several improvements of the NLU models to comply with this level.

Adding a new model to the NLU module, especially the BERT-based segmentation model, started to degrade the response time of the whole module. We selected constant input to see the performance changes during development. The input was selected to contain multiple segments (*"hello how are you what's your name"*). The measurements were done on the same instance type as the production system uses.

The NLU version used in Alexa Prize 2018 had the average latency (over 1000 measurements) 153 ms. As we implemented models required for the new architecture, including BERT-based segmentation, the latency grew to 835 ms on average. The first improvement was to change the instance type from the CPU instance to the GPU instance, improving the average latency to 421 ms. However, the GPU instance introduced a latency caused by transferring data from the system memory to the GPU memory – the models process the input immediately, there is no queue to form a bigger batch by combining several requests. Therefore we implemented the following process:

1. Run segmentation model on a batch of $n$ speech hypotheses
2. Create input batch from all segment of all hypotheses
3. Combine all the additional NLU models into one computational graph-—allowing parallel inference across the models
4. Run the inference and assign the result to corresponding segments and hypotheses

The combination of batching segments and creating one computational graph from the individual trained models allowed us to decrease the overall NLU latency to 204 ms.

## 5   Examples of conversational exchanges

In this section, we present several dialogue fragments typical for the new aspects of the system. We try to demonstrate multiple parts of the responses as well as the ability to work with knowledge.

**User:** Do you like music?
**Bot:** Yes, I love music! What music genre is your favorite?
**User:** I really like funk.

**Bot:** I see!

**User:** Alfred Nobel was born in France, right?
**Bot:** No, Alfred Nobel was born in Sweden. Did you know that the synthetic element nobelium is named after him?
**User:** No, I didn't! Thanks for telling me.
**Bot:** No problem.

**User:** What movie is your favorite? And... do you like Tom Hanks?
**Bot:** My favorite movie is Matrix. Yes, I like Tom Hanks.

**User:** Hey, how many siblings do you have?
**Bot:** I don't have any siblings. You have three siblings, right?
**User:** Yes, that's right.
(...)
**User:** I have two siblings.
**Bot:** Oh, really? I remembered something else.

**User:** Where was Jara Cimrman born?
**Bot:** I'm sorry but I don't know that...
**User:** Ah, don't worry about it.
**Bot:** Thanks.

## 6   Conclusion

Alquist 3.0 brings the innovative usage of conversational knowledge graphs and adjacency pairs. We described how conversational knowledge graphs are essential for open-domain socialbots: it is thanks to the knowledge graph's ability to store the information mentioned during the conversation, as well as to serve as the foundations for constructing dialogues.

We showed that insufficient flexibility of the open-domain dialogue system could lead to lower satisfaction of the users. The utilization of smaller and more flexible dialogue structures can help overcome this issue. We demonstrated how we took advantage of dialogue adjacency pairs, which are small flexible parts of dialogues, how we efficiently create them out of word and sentence structure levels, and how we chain them in order to create flexible, highly coherent, and engaging open-domain dialogues.

## 7   Future Work

We developed a system that is highly connected to a conversational knowledge graph. The KG contains the information that the system is capable of having a conversation about, as well as relevant information about the user's preferences. The small segments of natural responses allow the system to be more adaptive in the response generation. However, during the development, we started experimenting with the generative models, which take a selected information from the KG, the corresponding dialogue act, and a context window as an input. Based on the information, the model is supposed to generate a natural language response. The models used in our experiments are not yet ready for a production-level system, but we want to continue with the experiments, and we believe that it helps the system to handle the "long-tail" cases in combination with template-based responses.

## References

[1]   Ian Lane Bing Liu. *Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling*. 2016. URL: https://arxiv.org/abs/1609.01454.

[2]   Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[3]     Samuel Broscheit. "Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking". In: *arXiv preprint arXiv:2003.05473* (2020).

[4]     Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5]     Claire Gardent et al. "Creating Training Corpora for NLG Micro-Planners". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 179–188. DOI: `10.18653/v1/P17-1017`. URL: `https://www.aclweb.org/anthology/P17-1017.pdf`.

[6]     John J Godfrey, Edward C Holliman, and Jane McDaniel. "SWITCHBOARD: Telephone speech corpus for research and development". In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 1. IEEE. 1992, pp. 517–520.

[7]     Matthew Honnibal and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear. 2017.

[8]     Christopher D. Manning Jeffrey Pennington Richard Socher. *GloVe: Global Vectors for Word Representation*. 2014. URL: `https://nlp.stanford.edu/pubs/glove.pdf`.

[9]     Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[10]    Zhenzhong Lan et al. "Albert: A lite bert for self-supervised learning of language representations". In: *arXiv preprint arXiv:1909.11942* (2019).

[11]    Jan Pichl et al. "Alquist 2.0: Alexa Prize Socialbot Based on Sub-Dialogue Models". In: *2nd Proceedings of the Alexa Prize (AP 2018)*. (2018).

[12]    Jan Pichl et al. "Alquist: The alexa prize socialbot". In: *1st Proceedings of the Alexa Prize (AP 2017)*. (2017).

[13]    Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[14]    Emanuel A Schegloff and Harvey Sacks. "Opening up closings". In: *Semiotica* 8.4 (1973), pp. 289–327.

[15]    Ottokar Tilk and Tanel Alumäe. "Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration." In: *Interspeech*. 2016, pp. 3047–3051.

[16]    Hongxia Jin Yu Wang Yilin Shen. *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling*. 2018. URL: `https://arxiv.org/abs/1812.10235`.

# A  Dialogue Acts

| T1 | T2 | T3 | Description | Corresponding SWBD tags | Example |
|---|---|---|---|---|---|
| da | | | | | |
| | Que | | request for information, "question" | | |
| | | Yesno | request for either confirmation or denial | qy, qyd̂, d̂ | Do you like music? |
| | | Wh | request for specific information | qw, qwd̂ | |
| | | WhOb | question on gramm. object | | Where were you born? |
| | | WhSub | question on gramm. subject | | Who starred in Matrix? |
| | | Choice | request for choosing between a set options | qrr | |
| | | Howabout | contextual request for information | qo | |
| | Ans | | Answer/reaction | | |
| | | Affirm | affirmative reaction | ny, na, aa, aap_am | Yes, I do. |
| | | Deny | negative reaction | nn, ng, ar, arp_nd | No, he wasn't. |
| | | Agree | agreeing reaction | aa, aap_am | |
| | | Refuse | refusal, disagreeing reaction | ar, arp_nd | I'd rather not. |
| | | Suspend | suspended reaction | ĥ | I'll have to think about it |
| | | Tosummons | indication of presence / conversation can go on | | |
| | | Clash | "oh really?" "wow!" | | Oh really? |
| | Inf | | Providing information | | |
| | | Obj | objective information | sd, no | John has three siblings. |
| | | Subj | subjective information | sv, q̂, no | I really like sports. |
| | | Repeat | repeating previous information | bm̂ | |
| | | Clarif | clarification, further explanation, correction | sd, no, sv, q̂, bm̂ | |
| | Act | | Non-verbal action, stopping, volume up etc. | | |
| | Req | | Request for verbal/non-verbal action | | |
| | | Repeat | request for repeat | br | |
| | | Clarif | request for clarification | | |
| | | Summons | summons | | |
| | | Action | command, request for a non-verbal action | ad | |
| | | Verif | req for verification (via summary/reformulation) | bf | |
| | | Sugg | suggestion, proposal | ad | |
| | Cont | | Continuators | | |
| | | Rhet | rhetorical question | qh | |
| | | Hm | "uh-huh" | b, ba | Uh-huh? |
| | | Que | "like really?!!" | bh | |
| | | Ackn | response aknowledgement | bk | Oh, okay. |
| | | Collab | collaborative completion | 2̂ | |
| | Form | | Formalities | | |
| | | Hello | greeting | fp | |
| | | Bye | farewell | fc | |
| | | Open | conventional opening | fp | |
| | | Close | conventional closing | fc | |
| | | Thx | thanking | ft | Thanks a lot! |
| | | Sorry | apology | fa | I am sorry… |
| | | Nw | downplaying response | bd | Nah, it's no problem. |
| | Inv | | Invalid act | | |
| | | Inv | abandoned, uniterpretable, nonverbal | %, x | |
| | | Other | other | fo_o_fw_by_bc, t1, t3 | |