The BoardBot Project

Konstantinos Pattakos

LT1200019

1. System Functional Specifications

The "BoardBot" chat bot was designed to meet the specifications of the "M913: Dialogue Systems and Voice assistants" class of the Language Technology MSc, in which the Rasa 2.0 and 3.0 Conversational AI platforms were introduced. The bot's function is to suggest to the user which board game to buy or to play next, based on the extracted information obtained through an automated dialogue sequence. The user is asked to give information concerning his preferences in board games, like game mechanics, play time, etc., which the system then uses to find an appropriate suggestion from an automatically assembled board game database. When a match is found, the user is presented with the board game's title, image, and short description, as well as a link to the game's Board Game Geek page.

This project was primarily based on the tools provided by the Rasa Open-Source Conversational AI. This was utilized via the Visual Studio Code editor. Moreover, all data was scraped by the Board Game Geek website, considered to be the most detailed and comprehensive source of information in the board games community.

2. Architecture

2.1. Filling out the slots

BoardBot lets the user initiate the conversation by extending a greeting, or by asking to find a board game. Regardless of those, the bot takes for granted that the user knows is main function, and so it takes the initiative in the conversation to ask for information. Its goal is to fill the 4 main slots in the `boardgame_form`, which remains as an active loop throughout the conversation. The 4 slots required are the following:

- The **Type**, determined by the question *"What type of game would you like it to be?"*, followed by examples. Types of games include Strategy games, Family games, Abstract games etc., and are broader characterizations based on the categories they include. While it is not easy to explicitly distinguish them from the categories themselves, boardgame fans (and by association, the users of this chat bot) can easily understand the difference.
- The **Category**, which is prompted by the chat by the question *"Would you prefer a specific theme or category?"*, followed by a number of examples. This is based upon the Board Game Geek categories, which distinguishes games based on a number of factors, like activity (puzzle, educational, racing, etc.), components (cards, dice, miniatures, etc.), or skills (dexterity, economic, memory, etc.).

- **Play time**, which is approximately how long the game lasts. For this, the user is asked *"How long do you want to play more or less?"*, and his answer is compared to the games' average play time, within a margin of 16%; this means that, if the user replies that they would like to play for about 60 minutes, the system considers games that last from 50 to 70 minutes.
- **Number of players,** which is determined by the question *"How many of you will be playing?"*. Depending on the answer, the system keeps the games that can include it in their range of the number of players.

The system contains 4 different custom stories, all of which direct the user to linearly answer questions that fill these slots. The different stories include 4 series of questions, in order for the user to be able to start the conversation by submitting a piece of information himself.

2.2. Submitting the form

Once all slots are filled. The system is led to the `submit form` rule, which utilizes the `action_suggest_game` action to find an answer.

The `action_suggest_game` action uses the information gathered in the **games_db_1000.csv** database, which contains information about 22.899 board games from the first 1000 Board Game Geek ranking pages. This was done by using the url "https://boardgamegeek.com/browse/boardgame/page/#", with # being a number from 1 to 1.000. Each of these pages contains urls to game pages, by ranking order. With this url we then can use the `parse_game` function, also found in extract_top_games.py, which scrapes each individual game's page and retrieves the following information about the game:

- Game title
- Game type
- Rating
- Average playing time
- Minimum and maximum number of players
- Categories
- Weight (how difficult it is to learn and play)
- Description
- Image url
- Game url

After the first 20 pages scraped the website required a login, so the `cookies` and `headers` parameters were used for the `requests` python library in **extract_top_games.py**. Four of these, namely game type, average playing time, number of players, and categories, are used to find the optimal suggestion. Once it is found, the `action_suggest_game` action dispatches a message, utilizing the game title, the image url, the description, and finally the game url.

2.3. Finishing the conversation

So once the slots are filled, the system suggests the answer retrieved. If no answer is retrieved, the system states that it can't seem to find a game matching all characteristics and prompts the user to try again. Otherwise, if the user replies that they were happy with the result, the system utters a greeting and ends the conversation.

3. System Components

3.1. Pipeline

The system's pipeline is found in the **config.yml** file and is necessary to determine the hyperparameters used to predict answers based on the user's intents. The pipeline does that by handling the system's NLU predictions. It includes the following basic components: the `WhitespaceTokenizer` is used to tokenize word sequences and to remove some symbols or characters from the text. The `RegexFeaturizer` utilizes regular expressions to extract vector representations of the user's intents, in order to achieve entity extractions and classify the user's intents. The `LexicalSyntacticFeaturizer` is used to extract lexical and syntactic features to recognize entities in the user's inputs. The `CountVectorsFeaturizer` uses intents to extract bag of words representations based on sklearn's `CountVectorizer`, in order to classify and finally choose an appropriate response. The `DIETClassifier` is an intent and entity Transformer that classifies the user's inputs and recognizes its entities, and then connects these intents with pretrained BERT or GloVe embeddings. The `EntitySynonymMapper` classifies all entities that have a common meaning to a unique value. The `ResponseSelector` outputs a confidence level value for the system's possible responses, and thus selects the most appropriate response. Finally, the `FallbackClassifier` classifies intents, if their classification proves ambiguous, by using a threshold value set here at 0.3.

3.2. Dialogue policies

Policies are used by the system to decide the order of the actions taken at each part of the conversation. In our case, two different policies were used to check if they would have an alternate impact on the conversation. As they did not make a significant impact to the conversation, default settings were chosen. These policies are:

Policy 1:

```
- name: MemoizationPolicy
- name: RulePolicy
- name: UnexpecTEDIntentPolicy
- name: TEDPolicy
```

Policy 2:

```
- MemoizationPolicy
- TEDPolicy
- RulePolicy
```

4. Evaluation

In order to evaluate the Boardbot's implementation, the chat bot was handed out to be tested by potentially actual users of the final product in terms of user experience. Since this chat bot is targeted at people who are board game enthusiasts, all testers were part of the specific community: Board game collectors, game designers, and play testers. The evaluation process consisted of two phases: In phase one, the bot was tested by 5 people, who then filled in a questionnaire, which was then considered to correct parts of the dialogue. In phase two the chat bot was now tested by 10 people, who again filled in a second questionnaire, allowing for new changes to the system. Links to the 2 questionnaire forms can be found below:

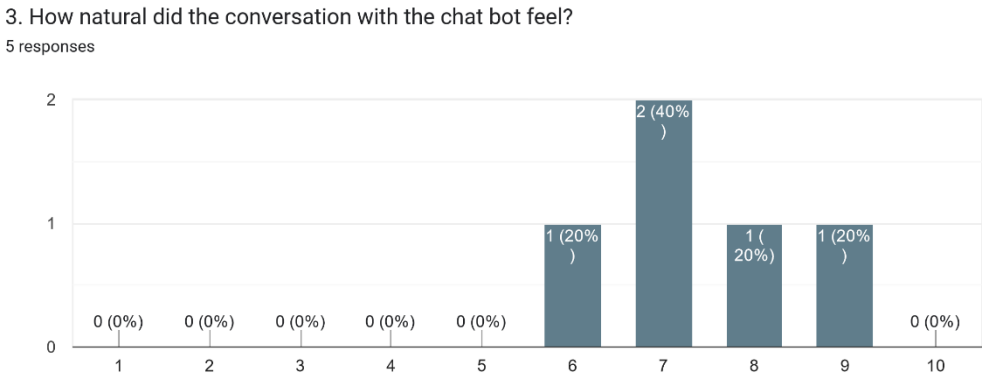
- Questionnaire form phase 1: https://forms.gle/mQNokadErBwSnvUh9

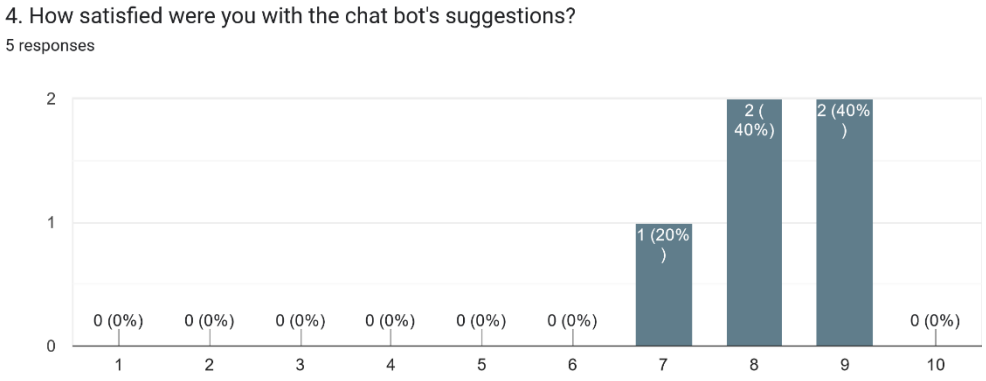
- Questionnaire form phase 2: https://forms.gle/QS4q2EwULMgPH6qo9

It's important to note that, during testing, the users were asked to answer the question concerning playtime not by just typing in a number, but by adding 'minutes' to their answer. This was because if for example a user would type in '40' instead of '40 minutes', the system would register this reply both in the 'preferred playtime' and the 'number of players' slots. Because this was explained to the users verbally and beforehand, it does not heavily reflect in their comments.

4.1. Evaluation Phase 1

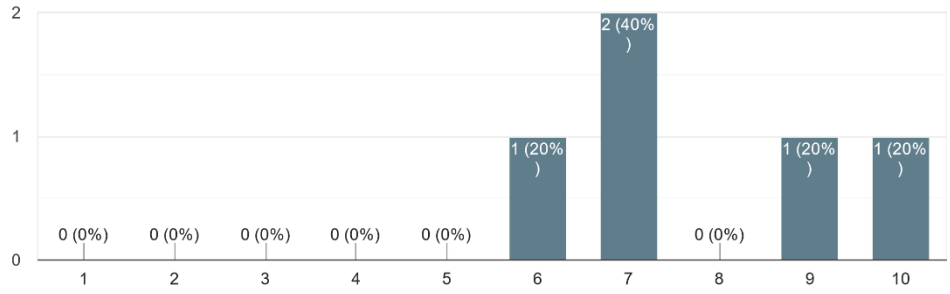
In phase 1, 5 participants were asked to fill in the form above. Out of the five, 2 belonged to the 26-39 age group, while 3 belonged to the 40-55 age group (question 1). When asked if they had used a chat bot before (question 2), 2 stated yes, 2 stated no, and one stated maybe. The rest of the questionnaire's results can be found below:





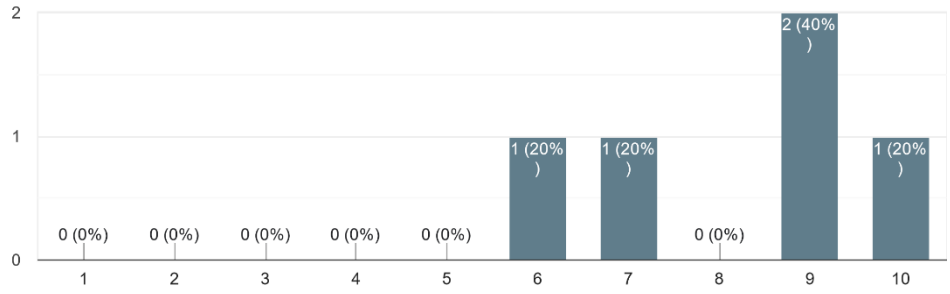
5. How clear were the chat bot's questions?

5 responses



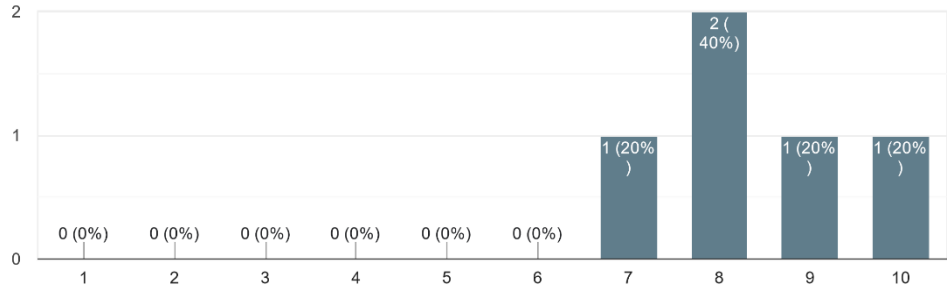
6. Where the chat bot's questions helpful?

5 responses



7. How satisfied were you with the overall experience?

5 responses



8. Would you have any suggestions that would improve Boardbot?

3 responses

It could ask for type of game

Can you add questions about when a game was published?

Board Game Geek also has game type

9. Would you like to comment on something you did or did not like?

4 responses

When I answered minutes it thought i meant players

I like easier games

Examples on questions were very helpful

I like that it guided me with questions

Overall, the users seemed to enjoy the conversation with the bot, they understood the questions posed and they mostly liked the recommendations made to them. As for as the comments in the final section go, users asked for the type of game to be included, as without it, a lot of the game's description would be missing from the questions. This was added in between the first and the second phase of the evaluation, but it created new problems, as the distinction between type and category is clearer to the people considering board games to be their hobby than to casual players.

The problem with the integer number filling both playtime and number of players slots was slightly evident in the comments. What is also commented was the fact that the chat bot took the initiative to guide the user through the process, possibly by the users who had not talked to a chat bot before.

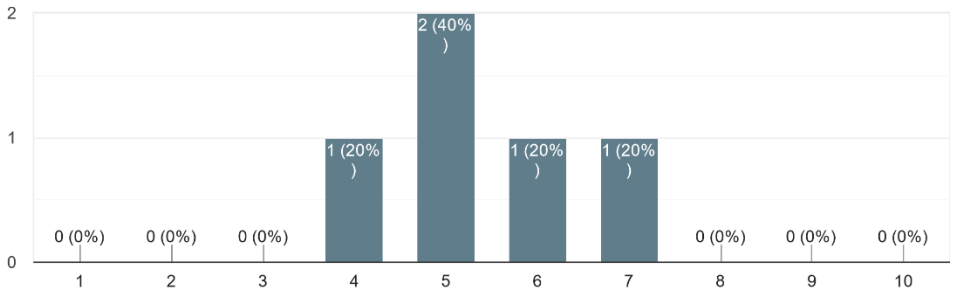
4.2. Evaluation Phase 2

After Phase 1, the game type slot was added, along with some examples of what a type could be, per the example of the category slot, which proved helpful to the users. The questionnaire was again handed out to users, 10 this time, 5 of which were the original testers from the first phase.

Out of 10 participants, 2 belonged to the 18-25 age group, 4 belonged to the 26-39 age group, and 4 belonged to the 40-55 age group (question 1). When asked if they had used a chat bot before (question 2), 7 stated yes and 3 stated no. 5 out of 10 stated that they had used BoardBot before (question 3). The rest of the questionnaire's results can be found below:

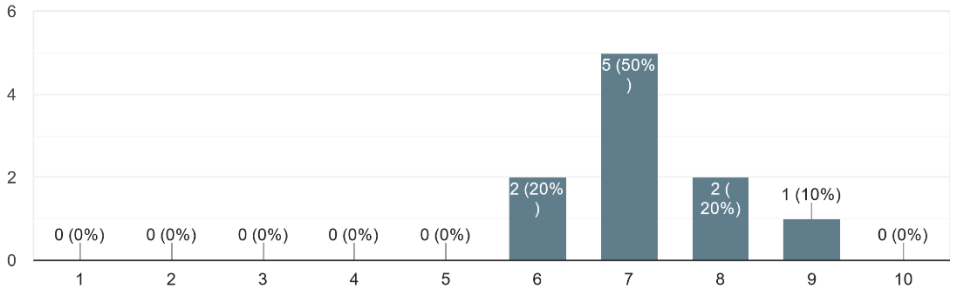
4. How improved do you think your experience was compared to the last time you used BoardBot? (Please answer only if you ticked yes to the previous question)

5 responses



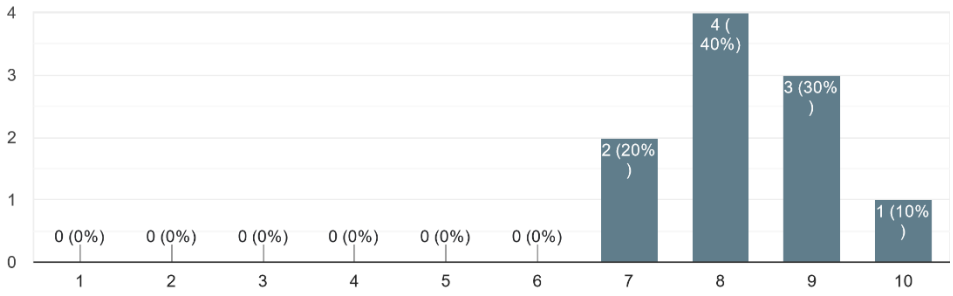
5. How natural did the conversation with the chat bot feel?

10 responses



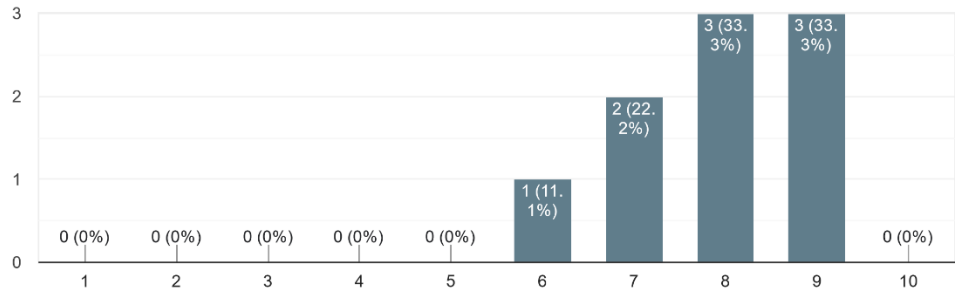
6. How satisfied were you with the chat bot's suggestions?

10 responses



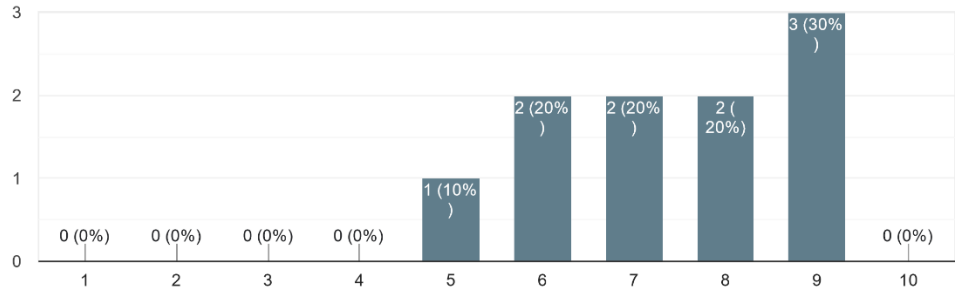
7. How clear were the chat bot's questions?

9 responses



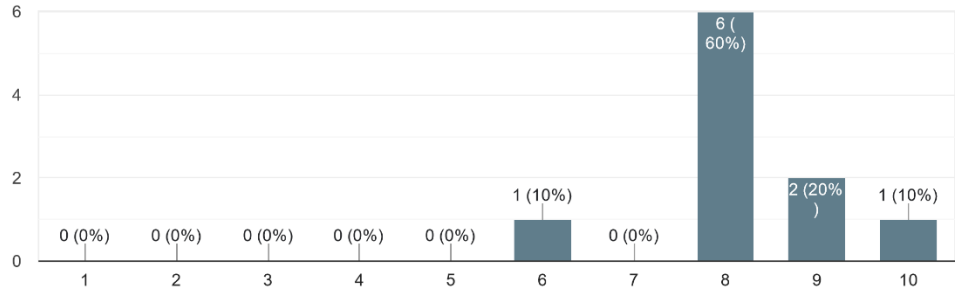
8. Where the chat bot's questions helpful?

10 responses



9. How satisfied were you with the overall experience?

10 responses



10. Do you have any suggestions that would improve BoardBot?

6 responses

Link to buy the game?

Add more questions

Suggest another game if i already had it

i dont need 110 players

No

translate in greek

11. Was there something you did or did not like?

3 responses

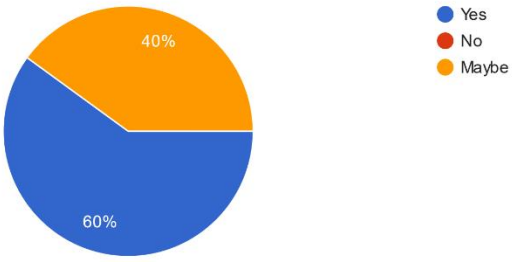
50 players problem not solved

I liked the game it suggested

I liked it very much

12. Would you use BoardBot in your everyday life?

10 responses



It is evident from the forms above that the addition of types slightly deteriorated the understanding of the questions, but it also seemed to keep user satisfaction high with regards to their satisfaction over suggestions. Some user suggestions proved interesting, such as adding a link to buy the game, or to make another suggestion if the user already owns the game. The latter was considered to be added in this installment, lack of time however did not permit it.

5. Conclusions and future work

For the purposes of making a chat bot that is both versatile and user friendly, the Rasa platform proved to be a very powerful tool. A big part of its strength was the clear characterization of the

pragmatic aspects of an actual dialogue: the distinction between user intents, system utterances, even forms highly reflect the foundations of a human-to-human conversation. With these in place, it makes it even more potent to utilize the NLU model that makes it all possible. It is also evident that the Rasa platform has many features that were underutilized or were not utilized in this project, mainly because of time limitations. They do however point out how BoardBot could evolve to become even better.

To improve BoardBot, then, a first suggestion would be to utilize the `rules.yml` file more, in order for the conversation to be less linear and to let the user take more initiative if that is what they wished. Also, adding answers to more potential user intents could move the bot to this direction, intents like asking what kind of categories are available, what is the publishing company of the suggested game, which games are in the top 10, etc. As suggested by a user and was supposed to be included already, the system could make alternate suggestions if the game was already in the user's collection. Finally, part of this project's future work should include connecting the system to a voice assistant like Alexa or Siri, in order to maximize its accessibility.