K-Bot: a K-Pop virtual assistant

Olga Zaghen 224436
University of Trento
olga.zaghen@studenti.unitn.it

Abstract

K-Bot is an assistant that can help any user find K-Pop songs matching their taste in music, and/or retrieve useful and interesting information about some K-Pop groups. This work was carried out as final project for the *Human Machine Dialog* course.

1 Introduction

K-Bot is a virtual assistant that addresses two tasks in the K-Pop domain. The main one is song suggestion and it is carried out by taking into account constraints that the user can optionally provide. The assistant targets not only K-Pop listeners, but also users that are not familiar at all with such genre. No background knowledge is necessary, hence the song selection process can be driven by only expressing preferences on general features, such as cheerfulness and energy. The second and minor task consists in providing information on some specific K-Pop groups, when requested by the user.

Most efforts were put in ensuring the best possible user experience for the whole target audience, rather than focusing on the variety of available songs, groups and facts.

The system was built within the Rasa framework, with the final goal of deploying it as an Amazon Alexa Skill.

2 Conversation design

The structure of conversations that the bot can handle varies according to the two tasks. Full dialogs are generally characterized by a mixed-initiative, but system- and user-initiative can be predominant in the major and minor task, respectively.

2.1 Song suggestion

The conversation flow for song suggestion is quite flexible, and it adapts to the user profile and the number and type of constraints provided when asking for a song. These can involve the artist, cheerfulness level, energy level and danceability level. Over-informative users may directly specify all of them, as in the example of fig. 1 or they may request a totally random song, and in these scenarios the agent will automatically look for a solution. When no preferences are explicitly expressed, the bot asks the user whether they still want to set some constraints or not; if so, a dominantly system-initiative conversation takes place, with questions such as *Do you want the song to be cheerful, or not?* that allow the agent to set all feature values and perform the search.

The system follows a form to achieve this task and, in order to establish a common ground with the user, some kind of acknowledgement is put in place to inform when the slots have been filled in correctly. As soon as all of them are set, the feature values are communicated before starting the song search, as an implicit form of confirmation. Some kinds of unhappy paths are also contemplated, in which the user may interrupt the form temporarily (e.g. asking for the list of available groups) or definitively, if they want to switch tasks.

2.2 Providing information about K-Pop groups

The minor task consists in question answering, providing facts about some available K-Pop groups. Although this may appear to be quite straightforward with respect to song suggestion, some potentially critical scenarios have to be properly addressed. These involve the user asking for some unavailable information, or when multiple (and possibly all) facts are requested at a time, as in *I want to know everything about EXO*, and *What is their company and how many of them are there?*. This task is mostly user-driven, but the agent takes the ground as soon as they stop making requests. Some of the discussed scenarios are shown in fig. 1.

2.3 Switching between tasks

When interacting with K-Bot, one may want to ask for some information and/or request a song suggestion multiple times, potentially alternating between the tasks. For this reason some functionalities were put in place to make the task switch as smooth as possible. For example, as soon as a song is recommended, the bot automatically asks the user whether they want to know more about the artist, facilitating the switch. Furthermore, when the user asks for facts related to a particular K-Pop group and then wants a song to be suggested, their request may be generic, such as *Ok, now I want a song suggestion*, or it may instead contain an implicit reference to the artist: *Please suggest one of **their** songs*. The system is able to interpret such references (*their*, *by them*, etc.) as constraints, hence it won't ask for the artist again, allowing for a more natural conversation (fig. 1).

2.4 Other types of requests

A user that is not familiar with K-Pop or with the bot itself may want to ask questions such as *But what is K-Pop? What information can I ask for? Can you list the groups I can choose from?* and the agent is designed to promptly answer them at any point in the conversation. Some user requests may also be out-of-scope: along with a general default answer, tailored responses were designed to address the specific cases

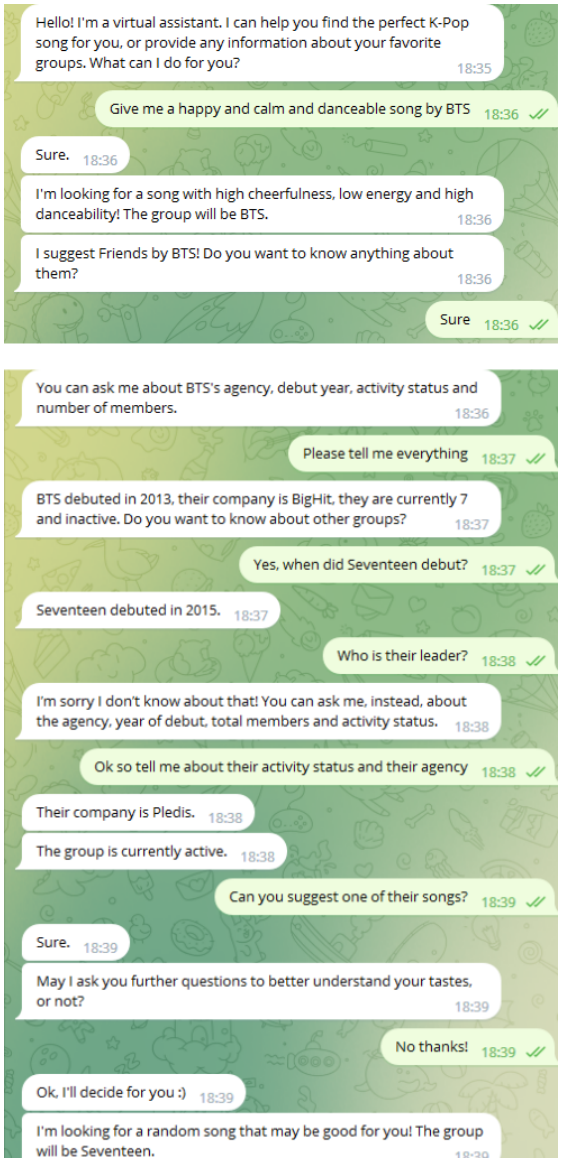

Figure 1: Examples of song suggestion and information requests. In this case, K-Bot is run as a Telegram bot.

in which an unknown group is specified, or an unavailable fact is requested (one example is in fig. 1).

3 Data Description and Analysis

3.1 Domain data

In order to achieve the tasks, the agent makes use of two different databases, both freely available on *Kaggle*[1].

K-Bot can provide information about and suggest songs by 7 different groups, namely BTS, NCT, Seventeen, TXT, EXO, Blackpink and Stray Kids. The user can ask about their debut year, agency, number of members and activity status and these facts could be easily retrieved from the *K-Pop Database*[2]. Having to deal with only a limited amount of artists allowed to further improve the user experience. For example, the system can easily list all of them to help the user when they keep requesting unavailable artists, or when they explicitly ask for the group list.

For song recommendations, the *K-Pop Hits Through The Years*[3] dataset was used instead (a detailed description is on the website). It contains the top K-Pop songs of the last 30 years, with approximately 50 per year. Besides the artist, for each song many features are reported such as danceability, energy, key and loudness, for a total of 11. I chose to consider only 3 of them for the song selection process: energy, danceability and valence (i.e. cheerfulness). These are the ones that, in my opinion, any kind of user mainly takes into account when looking for a song to listen to. On the contrary, most users wouldn't care about specifying properties such as key or mode. Feature values in the dataset are continuous, ranging between 0 and 1, hence I mapped them to three discrete levels (low-medium-high) to make the search feasible and to let the users express their preferences more naturally. Among all songs in the dataset, I selected and exported a set of 84, each one belonging to one of the 7 considered groups.

3.2 Training data

In order to train the model, NLU data was generated for the NLU pipeline and training stories were produced for the CORE components. During the initial stages I designed such data myself by also trying to integrate my domain knowledge, for instance by providing as many group names as possible in the examples to help the system generalize well and recognize unavailable group names, too.

Proceeding with the development of the project, in order to avoid the agent being biased by my own knowledge and perspective, I set up a Telegram bot and every few days I collected the conversations of 6 users interacting with it. Even though the system was not functioning satisfactorily yet, my goal in this phase was just understanding what kind of requests real users would make and how. The integration of such data was essential to properly address corner cases and different user profiles, considering their potential non-familiarity with the system and/or with K-Pop.

Regarding the NLU training data, for each intent an average of 22 examples were provided, with a minimum of 8 for *group_info_activity*, that only consists in asking about the activity status of a group, and a maximum of 105 for *request_song_suggestion_form*, that may be expressed through complex sentences involving all the possible entities. For such statistics multi-intents are not taken into account: only 5 examples were defined for each of them since, in theory, only a few are sufficient for the model to generalize well on the multiplicity of intents, starting from the indivisual ones.

A complete description of intents and entities can be found in appendix A. For each of the four entities at least 35 examples were given. A fair amount of them was required because of the several possible synonyms: for the *group* entity, different names may refer to the same group (e.g. *SKZ* and *Stray* for *Stray Kids*), while in the case of *song_valence*, *song_energy* and *song_danceability* the same categorical value could be expressed in multiple ways (e.g. *happy* and *not sad* both

[1] https://www.kaggle.com
[2] https://www.kaggle.com/datasets/kimjihoo/kpopdb
[3] https://www.kaggle.com/datasets/sberj127/kpop-hits-through-the-years?select=KPopHits2010.csv

Data	NLU pipeline						Policy			
	Intent Classification			Entity Extraction			Correct actions			
	Ac.	F1	Pr.	Ac.	F1	Pr.	Ac.	F1	Pr.	Correct
Training (NLU data + stories)	99.8	99.8	99.8	99.0	96.0	100.0	100.0	100.0	100.0	1216/1216
Test stories	97.2	97.2	97.2	100.0	100.0	100.0	97.7	97.7	98.1	127/130

Table 1: Performance of NLU pipeline and Policy components of the final model on test stories.

Featurizers				Intent Classification						Entity Extraction					
Regex	L.S.	C.V.	C.V. char	Train			Valid			Train			Valid		
				Ac.	F1	Pr.	Ac.	F1	Pr.	Ac.	F1	Pr.	Ac.	F1	Pr.
✗	✗	✗	✗	99.6	99.7	99.8	89.2	88.6	89.8	98.9	95.5	99.3	96.8	82.9	91.9
	✗	✗	✗	99.5	99.6	99.8	87.0	86.5	88.9	98.9	95.4	99.1	96.5	81.9	90.7
✗		✗	✗	99.5	99.6	99.8	86.6	85.6	87.9	99.0	95.7	99.7	96.1	78.2	87.0
✗	✗		✗	99.8	99.9	100.0	82.2	80.7	82.3	98.9	95.2	99.0	96.6	82.5	89.8
✗	✗	✗		97.9	98.0	98.8	84.0	83.3	86.6	98.6	93.9	97.3	96.9	84.1	90.0

Table 2: Ablation study on the featurizers used in the final model.

meaning *high cheerfulness*).

For what concerns training stories, 49 of them were considered, varying in the number of turns from a minimum of 2 to a maximum of 25: the shorter ones were tailored on specific requests, while the longer ones involved multiple switches between the two tasks, and often made use of *checkpoints*.

4 Conversation Model

The default Rasa components were used to define the model, after verifying that all of them worked satisfactorily and played essential or at least useful roles (the performed analysis is described in detail in section 5).

Specifically, in the NLU pipeline WhitespaceTokenizer was used for tokenization purposes, setting the *intent_tokenization_flag* to True to handle multi-intents; RegexFeaturizer, LexicalSyntacticFeaturizer and CountVectorsFeaturizer (both at word-level and char-level) were set as featurizers, DIETClassifier was chosen for intent classification and entity extraction, and EntitySynonymMapper to address entity synonyms. Additionally, ResponseSelector and FallbackClassifier were used to predict responses and classify *nlu_fallback* intents, respectively. For what concerns the policies, the adopted components were: MemoizationPolicy, that remembers training stories at inference time, RulePolicy, that puts in place rules explicitly specified for certain interactions, UnexpecTEDIntentPolicy, that can trigger the *action_unlikely_intent* action, and TEDPolicy, essential for predicting the next actions and recognizing entities.

The hyperparameters for all NLU and CORE components were set as the Rasa default ones, except for TEDPolicy and UnexpecTEDIntentPolicy for which *max_history* was increased to 8, that showed empirically to work better.

To enhance the performance of the system two forms were defined: *song_suggestion_form* to address the song suggestion task, and *group_info_form* to handle information requests. During the conversation, a total of 9 slots act as memory for the agent, storing for example group names and their facts.

In order to find a solution for the song recommendation process, some custom actions were defined. Specifically, the group constraint is always satisfied first when present, otherwise it is ignored in the search. For what concerns the valence, energy and danceability features, instead, it may be the case that not all combinations of their values are available for all artists. When this happens, the algorithm simply chooses one of the songs that fulfill the highest number of constraints. The set of available songs was selected in a way that, for each group, at least one constraint is always satisfied.

5 Evaluation

5.1 Intrinsic Evaluation

The model's performance on the test set was assessed through evaluation metrics such as accuracy, F1-score and precision. Beyond that, further investigation was performed on the NLU pipeline to gain insight on which featurizers were more and less useful, and to compare the behaviour of deep and shallow components for entity extraction and intent classification.

5.1.1 Test data

The data used to test the NLU and CORE components was composed of 6 test stories, obtained through the interaction of 6 different human testers with K-Bot on Telegram. The number of turns were respectively 16, 18, 24, 42, 18, and 28, and only two users were familiar with K-Pop. Such long dialogs allowed for a reasonable amount of actions to be predicted and were hence suitable to test the CORE pipeline.

Nevertheless, the users were mostly under-informative, expressing only a limited set of intents, and they didn't fully exploit the system's capabilities in entity extraction. Due to this lack of NLU test data, in order to perform a meaningful ablation study of the NLU components and a proper comparison with the baseline shallow model, a 5-fold Cross Validation on the entire NLU data was executed for these two analyses.

5.1.2 Evaluation on test stories

The performance of NLU and Policy components on training and test data is shown in table 1. Both components prove to generalize well but, as previously mentioned, the users were mostly under-informative and only 3 entity values were

Multi-intents	NLU Pipeline	Intent Classification						Entity Extraction					
		Train			Val			Train			Val		
		Ac.	F1	Pr.	Ac.	F1	Pr.	Ac.	F1	Pr.	Ac.	F1	Pr.
Yes	Baseline	91.7	89.8	89.1	66.4	62.9	63.4	99.0	96.0	100.0	96.6	84.1	98.3
	Final	99.6	99.7	99.8	89.2	88.6	89.8	98.9	95.5	99.3	96.8	82.9	91.9
No	Baseline	99.8	99.9	100.0	74.9	73.9	77.7	98.8	95.9	100.0	96.0	85.1	98.0
	Final	99.8	99.9	100.0	87.5	86.4	87.8	98.7	95.3	99.3	96.2	84.7	91.6

Table 3: Comparison on the performance of baseline and final NLU pipelines, with and without multi-intents.

provided throughout all test stories; for this reason the entity extraction results shouldn't be considered completely realistic.

5.1.3 Ablation study on featurizers

The goal of this study, whose results are in table 2, was understanding which featurizers from the Rasa default pipeline (listed in section 4) were more or less essential for the considered tasks. RegexF. and LexicalSyntacticF. were designed to enhance entity extraction, while CountVectorsF. for intent classification, and this is reflected in the results. Specifically, when CountVectorsF. is missing the performance in intent classification drops; entity extraction, instead, is not significantly affected and it even seems to improve when such featurizer operates at the level of character n-grams. This may be due to values related to different entities sharing some words (e.g. ***very*** *happy* and ***very*** *danceable*), hence confusing the model when the analysis is performed at n-grams level. On the contrary, LexicalSyntacticF. shows to be the most relevant one for entity extraction; one reason may be it considers features such as the entity position in the sentence, that is quite stable in the examples especially for the *group* entity. In general, all featurizers proved to be useful to some extent.

5.1.4 Baseline vs final model

The baseline pipeline was obtained from the actual one by substituting DIETClassifier with shallow models for intent classification and entity extraction, namely SklearnIntentClassifier and CRFEntityExtractor. Since Sklearn requires dense features, the default featurizers were substituted with LanguageModelFeaturizer, using RoBERTa as reference language model. The comparison was performed in two settings: with and without the presence of multi-intents in the domain file, stories and NLU data. This choice was made because while DIET can easily treat multi-intents as combinations of single intents it already knows and exploit this for training, Sklearn struggles in doing so and treats them as additional independent intents. The results in table 3 reflect this issue: for intent classification, the baseline scores are much lower with respect to the final model, and this trend gets even worse when multi-intents are considered. On the contrary, CRFEntityExtractor shows to work very well, also slightly outperforming DIET in entity extraction in both settings.

5.2 Extrinsic Evaluation

Along with intrinsic evaluation, the feedback provided by human testers must be contemplated, in order to gain insight on their satisfaction level and their general feelings on the interactive experience. For this kind of assessment, 6 users were asked to converse with K-Bot on Telegram and answer a set of questions in the end with a score ranging from 1 (low) to 5 (high). None of them were used to human-machine dialog systems, and only three of them were more or less familiar with K-Pop groups.

The questions and the average answers were the following:

How much did you feel the system understood you?	**3.7**
How was the pace of the interaction?	**4.3**
How was the quality of the speech feedback?	**3.8**
Did you know what to say at each point?	**3.8**
How easy was interacting with the system?	**4.2**
Was the system coherent in the conversation?	**4.2**
How appropriate were the system's responses?	**3.8**
How helpful were the system's responses?	**4.0**

It seems that the testers were overall satisfied. The most appreciated features were the natural pace and the facility of interaction; on the contrary, lower scores were given to the quality and appropriateness of the speech feedback.

K-Bot sometimes struggles in properly distinguishing between different out-of-scope request types, namely unavailable group, unavailable group-fact, or completely intractable request. Reinforcing this skill would be very useful, especially for K-Pop-non-familiar users: for example, the agent should be able to easily detect when a non-K-Pop group is proposed and provide a tailored and informative fallback response.

Another drawback affecting the system is its confusion when over-informative users use long utterances with multiple intents that it only encountered individually in the training phase. For example, when asked *Do you have some group preferences?*, one user answered *No, just suggest a song*: the bot interpreted it as an intent to start the song suggestion procedure, instead of a simple negative response, hence restarted the form from scratch. In order to better handle these situations, a wider range of multi-intents should be defined and addressed in the domain file and training data.

6 Discussion and Conclusion

One of the drawbacks of K-Bot is the low variability of groups and facts it can provide. The initial plan involved dealing with larger and updated datasets for both tasks. During the development, downsizing the data helped very much in refining the system and enhancing the user experience. It would be very interesting to expand this work in such direction in the future, making use of a bigger and updated knowledge base while keeping it flexible from the user's perspective.

For now, although its limitations, the agent can assist expert and non-expert users dealing with the K-Pop domain for simple tasks, providing a satisfactory experience for most user types and profiles.

References

[Bunt et al.2010] Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an ISO standard for dialogue act annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

[Mezza et al.2018] Stefano Mezza, Alessandra Cervone, Evgeny Stepanov, Giuliano Tortoreto, and Giuseppe Riccardi. 2018. ISO-standard domain-independent dialogue act tagging for conversational agents. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3539–3551, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

A Intents and Entities

The list of entities in the domain file is:

- *group*
- *song_valence*
- *song_energy*
- *song_danceability*

When any of them is provided, its corresponding slot is filled in. For what concerns the intents, instead, they are:

- *welcome_greet*
- *request_song_suggestion_form*
- *request_group_info_form*
- *inform_group*
- *welcome_greet*
- *request_song_suggestion_form*
- *request_group_info_form*
- *inform_group*
- *group_info_general*
- *group_info_debut*
- *group_info_members*
- *group_info_company*
- *group_info_activity*
- *group_info_unknown*
- *response_positive*
- *response_negative*
- *response_middle*
- *response_neutral*
- *request_list_groups*
- *general_info_kpop*
- *general_info_bot*
- *inform_song*
- *stop*
- *goodbye*
- *out_of_scope*
- *group_info_debut+group_info_members*
- *group_info_debut+group_info_company*
- *group_info_debut+group_info_activity*
- *group_info_debut+group_info_unknown*
- *group_info_members+group_info_company*
- *group_info_members+group_info_activity*
- *group_info_members+group_info_unknown*
- *group_info_company+group_info_activity*
- *group_info_company+group_info_unknown*
- *group_info_activity+group_info_unknown*
- *group_info_members+group_info_debut+group_info_activity*
- *group_info_members+group_info_debut+group_info_company*
- *group_info_activity+group_info_debut+group_info_company*
- *group_info_activity+group_info_members+group_info_company*

Of course, in the setting where no multi-intents were considered (section 5.1.4) the last 14 were removed.

It can be observed that only some intents coincide with the most general Dialog Acts (as the ones proposed for the ISO-standard (Mezza et al., 2018; Bunt et al., 2010)), while most of them are highly specific. For example, instead of having a single intent for requesting information about groups, many were defined, one for each fact: *group_info_general*, *group_info_debut*, *group_info_members*, *group_info_company*, *group_info_activity* and *group_info_unknown*.

This choice was forced by the need of keeping the system as simple as possible while still performing satisfactorily although the relatively limited amount of training data. Indeed, an intent such as *group_info_members* may be expressed with both *How many members are there?* and *What is the group size?*, and the same goes for *group_info_company* with *What is their agency?* and *I want to know to which company they belong*. Because of such a variety of equivalent expressions, correctly identifying the request using a single intent would require a complex configuration of entities and synonyms, that in turn would require a higher amount of training data with respect to current availability.