Proceedings of the Workshop on Hybrid Intelligence for Natural Language Processing Tasks HI4NLP (co-located at ECAI-2020)
Santiago de Compostela, August 29, 2020, published at http://ceur-ws.org

Domain-Independent, Task-Oriented Chatbot Creation and Conversation Policy Management Framework

Tolga Çekiç[1][2] and **Yusufcan Manav**[1][3] and **Enes Burak Dündar**[4] and **Osman Fatih Kılıç**[5] and **Onur Deniz**[6] and **Seçil Arslan**[7]

Abstract. In this paper, we present a chatbot creation framework that can help people with no technical expertise to design and create chatbots for any domain. This framework enables the creation of highly customizable chatbots that can range from simple question answer systems to chatbots that can handle more complex dialogue flows. In order to be domain independent we created a general Turkish language model using ELMo architecture and intent detection models for chatbots are trained using embeddings generated via this language model. Additionally, in order to make conversations more seamless and cohesive, dialogue act classification is integrated into conversation policy management. The framework also includes an additional tool that allows monitoring of past chatbot conversations and provides analytic tools supported by clustering algorithms.

1 Introduction

Conversational systems, or as they are commonly called chatbots are ubiquitous nowadays as they are used for simple general conversation or more specialized tasks such as customer services. With their popularity much research is focused on chatbots to make them more effective in solving users' problems and ensure conversations with a chatbot are akin to conversation between humans.

While general conversation chatbots usually consider a few utterances from users without coming to a point, for task-oriented chatbots sometimes the system must keep information from a much previous utterance and should try to steer conversation toward a certain point, thus completing its appointed task such as booking a ticket. While the two approaches have much in common, considering the research and the techniques used, they present different challenges [7].

Research into chatbots started decades ago with the advent of ELIZA[13] which was designed to mimic psychotherapy and uses pre-programmed rules to generate answers. It receives user utterances and by using certain words and word-of-speech tags, it prepares a response by following predetermined rules. Other chatbots following similar rule-based design to ELIZA have also been developed over the years with their own advancements such as PARRY[4] and ALICE[12].

In order to create chatbots that can perform more complex conversations, rule-based approach can be limiting since it can be unmanageable to create rules that cover more cases. Thus new methods are devised to create more effective chatbots. IRIS[3], a general conversation chatbot, uses data from many conversations and extracts the most appropriate responses to user utterances. For information retrieval, IRIS uses a vector space model. Gandhe and Traum also proposed a TF-IDF model to retrieve answers for chatbot from a dataset of text dialogue scripts.

With the advancements in deep neural networks, machine learning based chatbots have become increasingly successful. Vinyals and Le introduced sequence to sequence (seq2seq) learning for conversation systems for generating dynamic responses to each user utterance [11]. Although they are very successful in generating human-like sentences as pointed out Sordoni et al. They are actually limited in context sensitive conversations and carrying on information from previous utterances [10].

While machine learning methods can help create human-like conversations for general conversation, task-oriented chatbots usually require more complex tools for conversation management. Since task-oriented chatbots are generally deployed commercially and interact with customers, their responses must be more precise and carefully constructed. Thus, unpredictability of seq2seq model sentences can sometimes be undesirable for such chatbots. Also, some information must be specifically collected from users and must be kept to complete task. For instance a chatbot that sells flight tickets must collect departure and arrival locations as well as date so as to be able to present an offer and sell a ticket. In order to create such chatbots intent-slot model is used [5]. Intents are what a user wants to do with a chatbot and slots are actually entities that must be collected or filled for the chatbot to complete its task related to the intent. In order to create chatbots with intent-slot model, machine learning methods for intent detection and entity extraction are mixed with other techniques such as state tracking and policy management.

Task-oriented chatbots with intent-slot model can be used for many different domains and they would have similar designs. In order to create a scalable system for developing task-oriented chatbots for multiple domains new frameworks were devised. These frameworks such as Amazon Lex, Google Dialogflow, Microsoft Luis help create chatbots with little requirements for programming expertise. In this paper we offer a novel chatbot creation framework equipped with tools to handle common problems that can be encountered by conversation systems. In order to have a powerful intent detection mechanism that can work in multiple domains, our framework uses a general ELMo based language model and uses a deep neural network classifier based that uses ELMo embeddings [8]. Furthermore, an additional hybrid intent detection classifier is built with rule-based intent detection as well as a machine learning so as to find intents even

[1] Equal Contribution
[2] YapıKredi Teknoloji, Turkey, tolga.cekic@ykteknoloji.com.tr
[3] YapıKredi Teknoloji, Turkey, yusufcan.manav@ykteknoloji.com.tr
[4] YapıKredi Teknoloji, Turkey, enesburak.dundar@ykteknoloji.com.tr
[5] YapıKredi Teknoloji, Turkey, osmanfatih.kilic@ykteknoloji.com.tr
[6] YapıKredi Teknoloji, Turkey, onur.deniz@ykteknoloji.com.tr
[7] YapıKredi Teknoloji, Turkey, secil.arslan@ykteknoloji.com.tr

if training data is sparse or unevenly distributed. Our framework also has a recurrent neural network (RNN) based dialogue act classifier that is used in addition to the intent detection and the entity extraction to have a flexible policy management that can handle complex conversations. Additionally, an actively used chatbot may need to be updated to understand and perform new tasks over time. In order to effectively find these new tasks and to collect their related data the machine learning models, we have developed an analytics tool to reinforce capabilities of our chatbot generation framework. With this tool, new intents can be discovered or data sets for existing intents can be expanded.

2 Dialogue Design

Generally chatbot content is gathered and structured by the people who work in public relations or marketing. They mostly have very little to none coding experience and need people with programming expertise to maintain the chatbot content for them. We desired people who supervise the chatbot content also manage the dialogue design. Thus, the content design is decoupled from technical parts that manages the conversation to simplify creating the chatbots. We developed an interface for users to create new topics, and structure the content according to their needs like edit, delete, create flows swiftly.

To provide this user interface and ease the creation of chatbots, we streamlined the process and decided to use basic building blocks. These are:

1. **Intent**
Intents are the tasks users want to accomplish. They are the main building blocks of the chatbots in this framework, and they contain some or all the following components in them. They can be used in solitude or chained to each other to create flows for more complex tasks.
2. **Entity**
They are used if information is needed to be collected from the user to complete a task associated to an intent. Two types of collection method can be used to collect entities; prompt; a question is asked to user and waiting for input or choice; the answer is selected from predetermined set of choices. There are multiple built-in entity types to satisfy the users' needs like, phone number, date etc. as well as ability for chatbot content supervisors to create their own custom entities.
3. **Training Sentences**
Training sentences are used in the training of the intent detection classifier. They are the possible sentences which customers use to state a specific intent.
4. **Response Actions**
This component determines the responses given by chatbot for a specific intent. After these responses given, either chatbot can revert to its default state waiting for user utterances to find an intent or another follow-up intent can be set and chatbot prompts the user according to this new intent. Before determining the responses of chatbot some other actions may need to be taken first, these actions are determined by functions component.
5. **Functions**
Functions are actions taken by chatbot to complete the required task of an intent. These actions generally uses collected entities as inputs and they can have connection to external systems such as when making money a transfer.

The organized and compact chatbot structure has shorten the time and eliminated the coding knowledge needed to create chatbots, while providing the capability to complete tasks independent of domains.Also as can be see from Figure 1, domain model also has configurations satisfy conversation flow needs for different domains.

Chatbot models are created with defining one or more intents according to the topics that have been determined by the content supervisors. While creating each intent, possible user utterances that can specify the intent are added as training sentences. If any entity is needed to be collected for an intent the corresponding type of the entities are inserted to the intent, and their collection type is selected. After that, any integration needed are added as functions. Eventually the response actions added to the intents. There can be single or multiple response actions and they may have different conditions which provide the dynamic responses with regarding to users input to the entities.

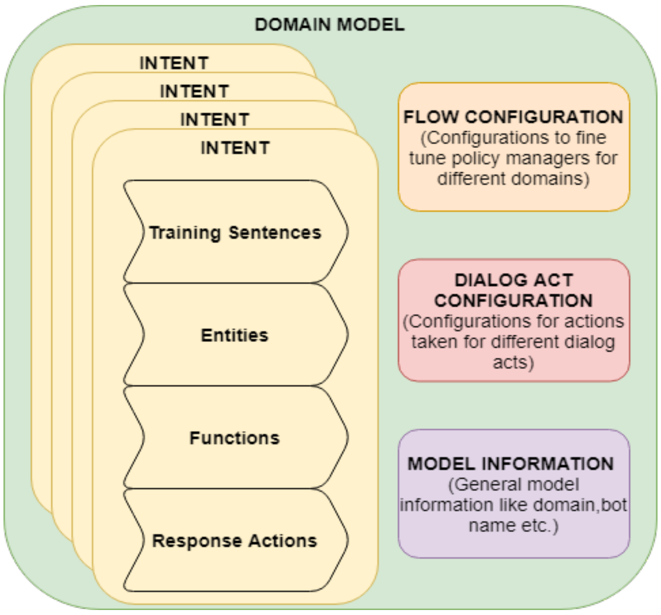
DOMAIN MODEL
INTENT
INTENT
INTENT
INTENT
Training Sentences
Entities
Functions
Response Actions
FLOW CONFIGURATION
(Configurations to fine tune policy managers for different domains)
DIALOG ACT CONFIGURATION
(Configurations for actions taken for different dialog acts)
MODEL INFORMATION
(General model information like domain,bot name etc.)

Figure 1: Domain Model Structure

Each of the chatbot models that are deployed, are uniquely identified in the conversation manager so that users can go back to an older model if they want to withdraw their changes and continue on top of that older model. This uniquely identified models also provide to serve multiple models in a domain and channel agnostic way. Users from different domains are directed to their respective chatbot models, and given an answer from that model.

3 Intent Detection

Task-oriented chatbots need to understand users to help them with their problems. Chatbots created with this framework can contain multiple intents. Smaller scale chatbots that are tailored to one specific task could be created with this framework, but in many domains more complex chatbots with multiple intents are desired. Hence, having a powerful and flexible intent detection classifier is a must in task-oriented chatbots. We developed a two stage hybrid structure for this task.

First stage is a machine learning based intent classifier that uses ELMo contextual embeddings[8]. For Turkish, we have trained an ELMo language model from scratch using Turkish Wikipedia dump and Bogazici Web Corpus as data sets [9]. Our combined training data contains more than 500 million tokens. Character based nature of the ELMo is very helpful for obtaining better word representations for Turkish since it is an agglutinative language.

The ML-based intent classifier is trained with the sample sentences on the chatbot model. Contextual embeddings of these sentences are extracted using ELMo and then those embeddings are fed to a multi-layer perceptron with softmax output layer. For chatbot usage three most probable intents of a message are retrieved using this model.

The rule-based intent classifier is stationed as the second part of the hybrid intent detection module. This part is used for the intents with less than minimum amount of training sentences or for specific intents that can be identified by very distinct keywords or phrases. For these intents, basic phrases are defined, and these patterns are searched in the user utterances. The matched patterns ranked according to their token counts, with the assumption longer phrases are more specific in describing an intent. Some intents could be reliably determined at this stage. However, this method does not consider semantic or contextual information. It makes only pattern based matching and can miss words with similar meanings that are not present in the list of pre-determined phrases.

In the general flow of the intent detection those two methods are used in tandem. The possible intents from the first stage is checked if any of them returned with enough confidence then this intent is selected, but if any of them does not have enough confidence, then the second stage is used and checked if any of the phrase patterns matched in the user utterance. If a single intent is matched then this intent is selected. If zero or more than one intents matched, then up to a number(which is a configurable property) of possible intents from the both stages are presented to the user and ask if they would choose any of the choices.

4 Entity Finder

Task oriented chatbots sometimes may need to extract information from user utterances to complete their tasks. The entities needed to be collected have many different types and also users can specify the given type of entities in a different ways, for example users can specify the date as “next Monday” or “06.01.2020” and the entity finder should determine both are entities that denotes a date.

First step of the entity finder is spell correction for utterances. For spell correction in Turkish Language, we utilized Zemberek, a natural language processing library [2]. If the word is not in the dictionary, it is then converted to the word with the nearest Levenshtein distance to the original one [6]. Then, the numbers in letter are converted to the digits to simplify the extraction of entities with numeric parts.

There are entity types with different challenges in extracting. For the basic types like numerical, time etc. we use regular expressions that cover the possible notations which users can write. To extract more complex types like dates we created a module that detect part of speech tags (POS) using the aforementioned Zemberek library and then extract entities using those POS tags and regular expressions.

There can be also custom types of entities which can have a finite number of values that are predetermined by chatbot content supervisors. Those types of entities have choices and also different possible words or phrases added as synonyms. Then those choices and their synonyms searched in the user utterances, in a fuzzy manner with Levenshtein distance. If a similar or exact pattern is found then this entity value is extracted.

Each entity if they are required to be collected in their related intent, has a prompt to ask the user specifically to get a value from the user. However, users may have provided values for entity without being prompted and chatbot need not ask for that entity again. So that after finding intent, entity finder module starts to extract entities related to that intent, even for the entities that it did not prompt explicitly to provide users a more realistic conversation.

5 Conversation Management

Managing the conversation meticulously in the task oriented chatbots is important for helping users. Conversation management is about tracking chatbot’s state as well as adapting to changing purposes of the users to create more natural conversations. Users may change their question while a chatbot is trying to collect entities for an intent, or a user may give negative feedback about an answer provided, and chatbots should handle those user utterances like human to human conversation rather than being stuck in the same state of conversation.

Chatbot state is determined by found intent and chatbot’s tasks in this intent like entity collecting, or responding. Chatbot states are:

1. **Idle:** Default state of the chatbot. Chatbots in this state wait for user utterances to detect an intent.
2. **Slot Filling:** If an entity needed in an intent, chatbot actively tries to collect entity values by asking questions and if possible giving choices to the user.
3. **Confirmation:** Some intents require user confirmation before taking an action. In this state chatbot asks confirmation to users for proceeding with the task and waits for their response.
4. **Return Action:** In this state chatbot returns an answer to the user or takes action using functions inserted from the UI.
5. **Next Intent:** If current intent will be followed by another intent in return action state, chatbot transitions to this state.
6. **User Refine:** If chatbot is unsure about the intent of the user and it has some possible intents then a question is asked the user to clarify if they meant any of the possible intents found

Those different states alter the actions taken by the chatbots to complete tasks, but that is a linear structure and may not work for some cases. Sometimes users can change their questions, or give a negative feedback to a chatbot reply. To accommodate such user behaviors intent detection alone would not be enough. Besides the intents of an utterance, dialogue acts of the user utterances should also be extracted. Dialogue act is the function of an utterance in a conversation context, like question, statement or command. We tailored a distinct dialogue act class set for the chatbot system by analyzing the user behavior in the utterances collected from real-life conversations. These are: *Statement, Wh-Question, Yes-No Question, Answer Accepted, Answer Rejected, Not Understanding Feedback, Command, Greeting*.

The dialogue act classifier uses the same ELMo based general Turkish language model used in intent detection for word representations. These word representations are fed to a bi-directional LSTM with attention layer. At the end of the pipeline there is a multi layer perceptron to decide the dialogue act class. The data set we used for training the dialogue act classifier is created by manually tagging the utterances received by our chatbots.

Figure 2 shows the architecture of the conversation management and analytics. The chatbot state, the dialog act of the user and the intent found is fed to a policy manager to decide the next action of the chatbot in a dialogue. For example if a user rejected an answer, giving that answer to the next user utterance again would be frustrating for the user. Also, the chatbot state is readjusted according to dialogue

Domain	Channels	Intent Count	Content
Banking	WhatsApp Facebook Messenger Yapi Kredi Web Page	234	Answers frequently asked questions in banking also makes calculations and gives information to users. (Topics are coherent around the channels only the responses are customised to channels' visualisation capacity.)
Help Desk	Help Desk Page Pilot	346	Helps bank employees with their questions and technical problems
Tourism	POC	24	Help users to find hotel or transportation, make and edit reservations
HR	POC	36	Helps employees to get answer about personal benefits like leaves

Table 1: Chatbots From Different Domains Created Using this Framework

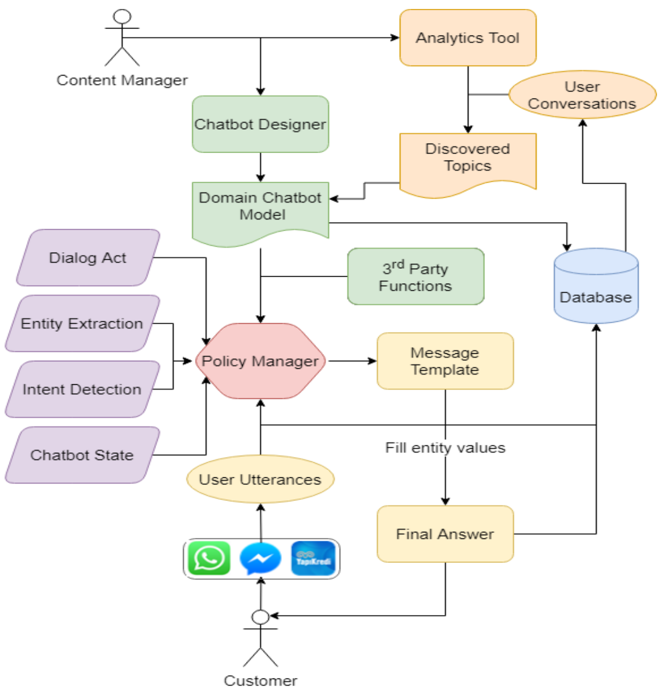

Figure 2: Conversation Management Architecture

acts. For instance, if a user asks a question while chatbot is in the slot filling state, the state is changed to idle and that user utterance sent to intent detection module to find the new intent of the user.

6 Analytics

After a chatbot is created, chatbot content must be maintained continously with updated or new intents in order to respond to the changing demands of the users. Content supervisors for chatbots should analyse previous dialogues in existing chatbots to determine when this should be done. Such an analysis would take a sheer amount of time and workforce if done manually. In order to address these issues we developed an analysis tool for chatbot conversations that employs unsupervised clustering algorithms and search capabilities.

Each utterance received by the chatbots are indexed with their metadata in Lucene; a search library written in Java which has capabilities like, fuzzy, phrase and wild card searches [1]. Using indexed information supervisors can search utterances thoroughly. They can look for utterances with no intent to see what sort of questions chatbots cannot answer. Supervisors can also search for a specific phrase, keyword to see user utterances around that topic This allows supervisors to gain granular insights from the data from the general topics.

Despite being helpful, examining each utterance individually is not enough for a complete analysis and may lead to wrong conclusions. Clustering methods are useful in inspecting vast amount of utterances, because they help humans to look to the topic in a broader perspective, by grouping the similar utterances together.

We create an n-gram based clustering method for this task. Utterances to be clustered are spell corrected then lemmatized to lessen the diversity of the n-grams due to the agglutinative Turkish Language. Later, stop words are eliminated, and then the n-grams are created. In the next step by using co-occurence of these n-grams, the sentences are grouped together. This step iterates a number of times (which can be set by supervisors) with increasing token co-occurence threshold for merging clusters. This threshold is not constant because at first iteration each sentence is basically their own cluster and they have limited number of n-grams and each co-occurence is more important than the later steps with bigger clusters. Higher threshold in the beginning iterations prevents sentences from creating clusters while lower thresholds in the later stage gravitate towards one big cluster that covers all utterances. At the last stage created clusters are visualized. Those utterances can be imported as training sentences to intents in the model.

The clustering can be used with the search in a way that supervisors can select a set of the utterances with the search. They can then execute the clustering algorithm over this set to have more specific analysis. Furthermore this clustering algorithm can be triggered automatically for each domain in fixed time intervals. This step can be performed over the utterances with no intent for the detection of the emerging topics.

7 Conclusion

In this paper, we presented our chatbot creation framework that can generate chatbots for any domain. Our framework is supported with state-of-the-art NLP techniques to better understand user messages and to be flexible and adaptable in generating replies to mimic human conversation behavior better. Also, by making every step of the chatbot generation configurable, chatbots with widely different behaviors can be generated and used.

Our framework has been in active use since December 2018 and chatbots for different domains have been created. First chatbot is created as Yapı Kredi Bank's customer service chatbot to serve customers on WhatsApp. Since its inception, this chatbot had 765,000 conversations and replied to 2.1 million messages. With the help of our framework the content managers also updated their models daily for better bot coverage rate, increasing intent count from 55 to 234. This chatbot model also extended to other channels with just tweaking the responses of the bot to channels multimedia capabilities. Second one is chatbot for bank's help desk to help employees with their technical and bussiness problems, which is now in pilot phase serving 100 branches of the bank, with 346 intents. Other use cases are in the POC phase for tourism and HR domains.

8 Acknowledgements

The authors would like to thank Atılberk Çelebi for his contributions to the work presented in this paper.

REFERENCES

[1] Apache Lucene. https://lucene.apache.org/. Accessed: 2020-03-10.

[2] Zemberek-NLP. https://github.com/ahmetaa/zemberek-nlp. Accessed: 2020-02-20.

[3] Rafael E. Banchs and Haizhou Li, 'IRIS: a chat-oriented dialogue system based on the vector space model', in *Proceedings of the ACL 2012 System Demonstrations*, pp. 37–42, Jeju Island, Korea, (July 2012). Association for Computational Linguistics.

[4] Kenneth Mark Colby, *Artificial Paranoia: A Computer Simulation of Paranoid Processes*, Elsevier Science Inc., USA, 1975.

[5] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall PTR, USA, 1st edn., 2000.

[6] Vladimir I Levenshtein, 'Binary codes capable of correcting deletions, insertions and reversals', *Soviet Physics Doklady*, **10**, 707, (February 1966).

[7] Maali Mnasri, 'Recent advances in conversational NLP : Towards the standardization of chatbot building', *CoRR*, **abs/1903.09025**, (2019).

[8] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, 'Deep contextualized word representations', *CoRR*, **abs/1802.05365**, (2018).

[9] Haşim Sak, Tunga Güngör, and Murat Saraçlar, 'Turkish language resources: Morphological parser, morphological disambiguator and web corpus', in *Advances in Natural Language Processing*, eds., Bengt Nordström and Aarne Ranta, pp. 417–427, Berlin, Heidelberg, (2008). Springer Berlin Heidelberg.

[10] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan, 'A neural network approach to context-sensitive generation of conversational responses', in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 196–205, Denver, Colorado, (May–June 2015). Association for Computational Linguistics.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, 'Sequence to sequence learning with neural networks', in *Advances in Neural Information Processing Systems 27*, eds., Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, 3104–3112, Curran Associates, Inc., (2014).

[12] Richard Wallace, *The anatomy of A.L.I.C.E*, 181–210, 01 2009.

[13] Joseph Weizenbaum, 'Eliza a computer program for the study of natural language communication between man and machine', *Commun. ACM*, **9**(1), 36–45, (January 1966).