A Chatbot by Combining Finite State Machine, Information Retrieval, and Bot-Initiative Strategy

Sanghyun Yi
College of Liberal Studies
Seoul National University
Seoul, Korea
sangyi92@snu.ac.kr

Kyomin Jung
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea
kjung@snu.ac.kr

Abstract

In this work, we designed a conversational system by combining a finite state method, a retrieval model and a machine-initiative dialogue strategy. By using the machine-initiative strategy, most of the user queries are handled by the finite state machine which models predefined dialogues. If the user's utterance is out of the range of modeled dialogues, the retrieval model processes the input. On comparative experiments, our retrieval model showed a better user satisfaction than the neural-network-based model. The experiments also showed that our context handling method and keyword emphasizing are key components of a retrieval model for the superior performance. In addition, dialogue strategies to retain initiative to machine was better to make a sustaining conversational system than to be strictly constrained to input sentences.

1 Introduction

In this work, we developed a computationally efficient conversational system by combining a finite state machine, an information retrieval model and a machine-initiative dialogue. These days, neural-network-based natural language understanding showed promising results[1]. The end-to-end framework works well in various domains without feature engineering. We used classical methods to design a chatbot, however, for several reasons.

First, even though users of the conversational system can talk about anything they want, we found out from customer interactions that most of the queries can be grouped into several similar types. Therefore, using the finite state machine as in [2–6]would be an efficient way to tackle the cases. Second, information retrieval works well[7, 8] and classical methods using TF-IDF or SVM make comparable performances to deep learning models in natural language understanding[9–11]. Therefore, under limited hardware resources, applying deep learning to all aspects of conversational system is a non-cost-effective overkill. We adopted thus an IR-method that uses TF-IDF and SVM. Lastly, it is natural that humans switch topics during dialogue for whatever reason (in case no further progress is necessary or interesting, for example). That is, utterances do not always have to be closely related. In chatbot system, this behavior can be adopted as a strategy for the machine in order to avoid being exploited and giving irrelevant responses. Therefore, we made a machine-initiative dialogue strategy: explore-new-topics by machine before exploit-current-topics by humans. For the machine-initiative dialogue, we made use of Washington Post data.

Comparative experiments using human evaluation proved the validity of our model. Despite its computationally light-weightedness, our system made better user scores than a neural-net-based conversational model. In addition, our machine-initiative dialogue strategy was the key to improve the user satisfaction by 0.56 points.

1st Proceedings of Alexa Prize (Alexa Prize 2017).

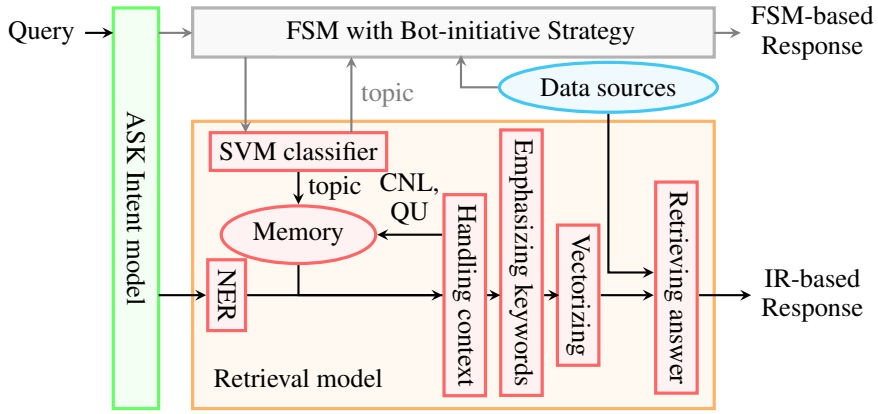

Figure 1: The overview of the system. The black and gray arrows represent the processes when the query is handled by the retrieval model and by the finite state method respectively. Details about 'Current NE List(CNL)' and 'Queries from User(QU)' are covered in Section 3.2.2. Also, see Section 3.2.3 and descriptions about intents in Section 3.1.1 for the information about the output of the SVM classifier, 'topic'.

2 Used data and knowledge sources

To build a conversational model, we used data from various sources. First, to handle everyday conversations, we used Evi[1] and dialogue data scrapped from Twitter[2]. Also to answer the user queries that request informations, our system used external knowledge sources including Wikipedia, Wolfram alpha, Urban Dictionary, BBC Good Food[3], Rotten Tomatoes, Official Charts[4], DBpedia Spotlight[12], The NewYork Times and Washington Post.

2.1 Popular topics data come from Washington Post data

To handle popular topics such as sports, movies, musics and other classes, we made the most of Washington Post data. Washington Post data are largely divided into 'Article' and 'Discussion'. In order to handle conversation about popular topics using finite state method, we used 'Article' data. And for retrieval model, we used both 'Article' and 'Discussion' data. Data named 'Article' are normal article data from Washington Post with 'Section', 'Title', 'Blurb', 'Body' and 'Comments' attributes. Data named 'Discussion' are similar with 'Article' but 'Body' of 'Discussion' are interviews from Washington Post. Therefore, they have structures of Q&A data. When a user requests to talk about a topic, for example, politics, then the finite state machine in our system introduces one of articles in 'Politics' section to the user and expresses an opinion by returning one of the comments of the article. And when a user's query is handled by the retrieval model too, the system responds using the Washington Post data when necessary. The details are given in the following sections.

Since the raw data had too many sub-types under the 'Section' attribute, we grouped them into larger classes. For instance, sections related to local affairs like 'Washington', 'Virginia' and 'Local' are grouped as 'WashingtonDC'. The classes are as follows: 'WashingtonDC', 'Sports', 'World', 'Politics-Society', 'Business', 'Transportation', 'Entertainment-Movie-TV-Book-Magazine-Media', 'Lifestyle-Hobby-Specialinterests', 'Opinions', 'Education', 'Science-Technology' and 'ETC'. 'Transportation' is a class of sections named such as 'Development-Transportation' and 'Traffic & Commuting' which are about public transportation issues in Washington DC. 'ETC' includes sections like 'Obituaries' and 'Weather' which are hard to be grouped in other classes.

[1] https://www.evi.com
[2] https://github.com/Marsan-Ma/chat_corpus
[3] https://www.bbcgoodfood.com
[4] http://www.officialcharts.com/charts/singles-chart/

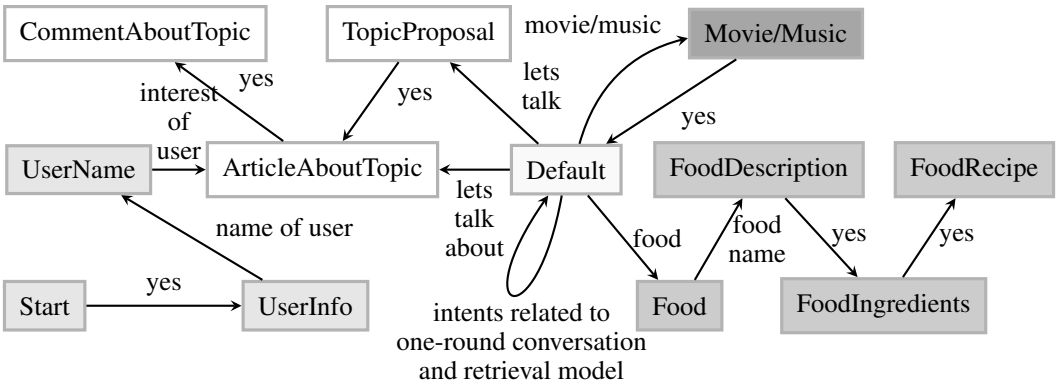

Figure 2: Expected dialogue structure in the finite state machine. States colored with a same color correspond to each dialogues. For instance, white-colored states are related to 'Current affair' dialogue. Transitions are done by the custom intents with the names of arrows which are made with ASK intent model. Intent named 'no' changes a state to default regardless of the current state, so edges from it are omitted. Regardless of the current state, intents other than 'yes' intent can go to the states indicated by the transition edges. For example, when the state is in 'UserInfo' and the user query is classified as 'movie' intent, then the system will give a response about movies and go to 'Movie' state.

3 System design and architecture

Our conversational system is a mixed-initiative system that manages dialogue using finite state method and a retrieval model. Mixed-initiative means that dialogues can be controlled by a system as well as a user[4]. The system can take the initiative in dialogue to get the required information but the user can also take the initiative to ask questions at any time they want. Dialogue management based on finite state machine means that a user's query will transit the current state of the system and the system will respond based on its state and the transition. The state stores the context of conversation and the transition is done by 'Intent' where a user's query is classified by ASK(Alexa Skills Kit)[13] intent model. Finally, if the user query was classified as an utterance that can not be handled by the finite state machine, then the query is sent to a retrieval model which we designed using TF-IDF[14] and SVM classifier[15].

3.1 Dialogues by finite state method

In order to model dialogues, we made 12 states and defined 31 transitions using ASK intent model(See Figure 2). When our system is initialized, it enters 'Start' state. A response is selected among the candidate sentences based on the transition triggered by a user query and by the explicit state which stores context information.

The dialogues managed by finite state method has two types: multi-round and one-round dialogue.

3.1.1 Multi-round dialogues

We designed five multi-round dialogues: 'Greeting', 'Current affair', 'Food', 'Music' and 'Movie'(See Figure 3 for examples).

'Greeting' dialogue is the conversation that takes place when a user starts the system. The system greets the user and asks some questions about the user. The states named 'Start', 'Yes' and 'UserName' are related to the 'Greeting' dialogue.

'Current affair' dialogue is the conversation about popular topics. When the 'Greeting' dialogue is finished or by the request of a user, the dialogue about specific topic starts. The newly grouped classes described in section 2.1 handled. The system will introduce a relevant article from Washington Post to the user and say its opinion about the article by returning one of the comments on the article.

'Food', 'Music' and 'Movie' dialogues are triggered when the user asks about the topics. 'Food' dialogue provides a general information about the food that user asked about and provides information

S1: Hi, this is an Alexa Prize socialbot. Nice to meet you! What do you want to talk about? I can talk about anything you want! If you want to talk about science, then ask me like “let’s talk about science". Or If you want to search something, then just ask me about it. I’ll let you know. But before we chat, can I have some questions about you?

U2: Sure.

S3: Thanks. Can I have your name? Please answer me like ‘My name is blah blah’.

U4: My name is John.

S5: Thanks John. The last question. What is your favorite topic among these? Sports, World, Politics, Entertainment, Hobby, Education and Science. Please answer only the word from the list I told you. I’ll repeat the list. Sports, World, Politics, Entertainment, Hobby, Education, Science.

U6: Science.

S7: Oh you like Science-Technology. Then, Do you know this news article? I read this a few days ago. The title is `<Title>` And it’s about `<Blurb>`. Did you know this?

U8: Yes. By the way let’s talk about music.

S9: What is your favorite music? I like edm. For machines, edm is like a meditation music. By the way, do you know top songs these days?

U10: Yes.

S11: Yeah I love Despacito (remix) by Luis Fonsi/Daddy Yankee/Bieber. . It’s really cool though. By the way, what is your favorite music?

U12: I love classic.

S13: I mean it’s exactly the response i was expecting.

Figure 3: An example of interaction with the system. U denotes a user’s query and S denotes the system’s utterance. Slots named `<Title>` and `<Blurb>` are filled with ‘Title’ and ‘Blurb’ attributes of an ‘Article’ of Washington Post data. From S1 to S5 is ‘Greeting’ dialogue and U8 to S11 is ‘Music’ dialogue. U6 triggerd ‘interest of user’ intent. U12 was handled by the retrieval model and the system gave the S13 as a response.

about ingredients and recipes of the food as the conversation progresses. Knowledges about food are scrapped from BBC Good Food. ‘Music’ and ‘Movie’ dialogues consists of giving personal preference on music or movies at the start and providing informations about current popular music or movies by scrapping Rotten Tomatoes or Official Charts respectively.

Now we describe our custom intents for making state transitions in multi-rounds conversations.

Name-of-user intent When a user says his or her name, ASK interaction model classifies the user query to this intent. For example, "My name is John" triggers this intent and changes the state to ‘UserName’. If the user’s name was not stored in memory, this intent stores the name in memory.

Interest-of-user intent This intent is triggered during ‘Greeting’ dialogue. During the dialogue, our system asks the user what topics they are interested in. For example, see Figure 3. U6 triggers this intent. The system will give a response like S7 and change the state to ‘ArticleAboutTopic’. There is a corresponding class of Washington Post data for each elements in the topic list in S5 and the slots in S7 are filled with a randomly selected article from the corresponding class. If the user requests about topic that is not predefined, we use a SVM classifier built with article data to select an appropriate class. Details about the SVM classifier are covered in Section 3.2.3. The class information where an article is retrieved is stored in memory as ‘topic’ of current conversation for a future use by the retrieval model.

Let’s-talk-about intent User’s queries such as “Let’s talk about science" or “I want to talk about sports" are classified to this intent. The system will change the state to ‘ArticleAboutTopic’ and give a response like S7 of Figure 3. We predefined keywords for each classes of Washington Post data and if a user query matched with one of them, an article form the corresponding class is randomly selected and the system makes a response like S7. If the user-requested topic is not predefined, it is

processed as same as ‘interest of user’ intent. The class information where an article is retrieved is stored in memory as ‘topic’ of current conversation for a future use by the retrieval model.

Let’s-talk intent This intent is triggered when the input utterance is like “Let’s talk" or “Can we talk about something else". Then the system will suggest to talk about a specific topic and change its state to ‘TopicProposal’. The corresponding class to the topic is stored in memory as ‘topic’ of current conversation for a future use by the retrieval model.

Food intent When a user says like “Food" or “Do you know how to cook", the query is classified to this intent. Then the system will return “Which food do want to know about or want to know the recipe?", change the state to ‘Food’ and save ‘Lifestyle-Hobby-Specialinterests’ as ‘topic’.

Food-name intent User’s utterances such as “Spaghetti" or “What is the recipe of hamburger" trigger this intent. The system will give a short description about the food and ask whether the user wants to know about the recipe. And the state will be changed to “FoodDescription". Data about foods are from BBC Good Food.

Music/Movie intent These intents are triggered when a user asks about music or movies(U8 of Figure 3). The system will tell its thought about music/movies and ask the user whether he or she knows the most popular music/movies these days(S9 of Figure 3). State will be changed to ‘Music" or ‘Movies’, and ‘Entertainment-Movie-TV-Book-Magazine-Media’ is saved as ‘topic’.

Yes/No intent When a user’s query is classified to these intents, the response of the systems depends on the current state. In ‘Start’ state, ‘yes’ intent make the system return an utterance like S3 of Figure 3 but in ‘Music’ state, it produces a response such as S11 in Figure 3. ‘No’ intent changes the state to ‘Default’ but ‘yes’ intent depends on the situation(See Figure 2).

3.1.2 One-round dialogues

We also defined intents for one-round conversations. These intents return an appropriate utterance and change the state to ‘Default’. First we made an intent named ‘Frustration’ for responding to user complaints. When the user was frustrated by an inappropriate output, our system will suggest new topics to the user and prevent from exploiting the current topics by the user. Moreover, there are 3 intents to answer when a user requests about current hot topics or sports news. Those intents use The New York Times as a knowledge source. We also made two intents to handle users’ inquiries about something. We handle the requests using Wikipedia, Urban Dictionary or Wolfram alpha. To handle user’s questions about the system’s personal information such as age, name, sex, address, maker and taste we made 6 different intents for each of them. Also we designed intents to respond to ethical issues or user’s requests such as ‘Play music’ or ‘What is my name’. When the user asks about his or her name, the system responds based on the name stored in memory. If there is none, the system will ask about the user’s name. We made an intent using Evi which handles users’ requests to do something. We also defined an intent for repeating the last response of the system. There is an intent for meaningless inputs such as “Hmm". Commonplace responses are returned when inputs trigger the intent. Finally, we made an intent which makes use of Evi to handle greetings from users.

3.2 Retrieval model

We designed a retrieval model to handle casual dialogues and popular-topic-related user utterances that can not be handled by the finite state dialogue management. We devised retrieval model that automatically select output sentences from candidate response dataset since we can not expect all of possible user queries. The used data includes Twitter dialogue and Washington Post data.

The retrieval model consists of four steps. First, it detects keywords or named-entities(NE) from the input sentence. Second, it handles context based on the detected NE’s. Third, with the keywords information and context, the topic of the input is classified into one of the grouped classes of Washington Post data or ‘Casual’ class which uses Twitter corpus and represents everyday conversations. Finally, from the Twitter or Washington Post data, an appropriate answer is retrieved and sent to the user.

3.2.1 Vectorizing data

To classify the input sentence to one of the classes and to retrieve an appropriate response from dataset, we vectorized all the documents in the data using TF-IDF(term frequency-inverse document frequency). We used TF-IDF because it is a computationally efficient algorithm to match documents relevant to queries[14] and enables state-of-art accuracies in sentence classification[10, 11]. The value is calculated by Equation 1

$$
\begin{aligned}
TF\text{-}IDF(t,d,D) &= TF(t,d) \times IDF(t,D) \\
TF(t,d) &= \begin{cases} 0, & \text{if } tf(t,d) = 0 \\ 1 + log(tf(t,d)), & \text{otherwise} \end{cases} \\
IDF(t,D) &= log\frac{|D|}{1 + |\{d \in D : t \in d\}|}
\end{aligned}
\tag{1}
$$

where t is the considered term, d is the document where the term is, D is the set of documents and $tf(t,d)$ is the number of times that t occurs in d. We considered the 40,000 most frequent n-grams(up to bigrams) from the data. Therefore, each input documents are converted to vectors with 40,000 dimension which each elements represent one of the 40,000 most frequent n-grams. For example, each questions in Twitter corpus, each articles' body, each comments in the article, each questions in discussions in Washington Post data are converted to vectors by TF-IDF. Also, every queries from a user during the interaction are vectorized by TF-IDF.

We made four different TF-IDF vectorizers, i.e., we used 4 different data sets for D in Equation 1. One of the D is Twitter corpus and the other is bodies of Washington Post articles. Also we used comments of Washington Post articles and question part of Washington Post discussions for the D. We made four vectorizers in order to make better vector representations at each steps of the retrieval process than when using only one vectorizer with the whole dataset.

3.2.2 Context handling with NE and emphasizing keywords

The retrieval model handles context by comparing NE's in the current query and NE's of the past queries so far. We used an open source project that annotate entities in query using DBpedia developed by Daiber et al. [12]. For the context handling, the retrieval model stores a list of keywords of the current conversation. We call it 'Current NE List(CNL)'. Also, the model stores list of queries from a user during the conversation. We call it 'Queries from User(QU)'. Since we can not guarantee the correctness of responses from the retrieval model, we only keep the inputs from the user. Our steps to handle context and emphasizing keywords are as follows.

First, make a NE list of the input from a user by using [12]. Then, intersect the NE list with CNL. If the size of the intersection is 0 delete the element of QU except current 2 queries because it means that the current input has higher possibility to be irrelevant to the past conversation. Else, keep the QU. Next, append the current input to the QU and update CNL to the new NE list. When we append the current input to the QU, we emphasized keywords by adding NE's at the end of the original input. Since the input will be vectorized by TF-IDF, the order of the word in the input is not important. By this method, the topic of the conversation is emphasized by increasing the term $TF(t,d)$ in Equation 1. Therefore, it helps to find the class to which the sentence belongs. Finally, concatenate all the inputs in QU into one string. This string contains context information by having user's past utterances, and keywords of the context and the current query. Based on this string, our system retrieve an appropriate answer.

3.2.3 Input sentence classification

Our retrieval model classifies the input to an appropriate class from which the model retrieves a response. The input is the string which was processed by the method we described in the above section.

At First, with 'topic' in memory, our model determines whether the query belongs to 'Casual' class or not. If 'topic' was empty, it indicates that the conversation was in progress without a specific topic(See descriptions about intents in Section 3.1.1). Therefore our model classifies the query to 'Casual' and search Twitter for a response. Else if 'topic' was filled with a name of a specific class name, our model searches for a response on Washington Post data having corresponding 'Section'.

Whether to use 'Article' or 'Discussion' of the data were randomly chosen. Second, we used linear multiclass SVM classifiers. We built two SVM classifiers to classify query into one of the classes of Washington Post data: one with the 'Article' and the other with 'Discussion' of Washington Post data. Each classifiers are used when the model searches the 'Article' or 'Discussion' respectively. SVM classifiers were trained with vectorized 'Body' of 'Article' and question parts of 'Discussion' described in Section 3.2.1. Therefore the SVM classifiers get a vectorized query as an input and give an output of a class name of Washington Post data which is the most relevant to the query.

Technically, SVM classifiers are not used in retrieval model but in finite state model. This is because 'topic' in memory already contains a name of the relevant class to the conversation which is obtained from a keyword dictionary or a SVM classifier(See descriptions about intents in Section 3.1.1). But in the comparative experiment which we will cover in following sections, we used the classifiers in the retrieval model.

3.2.4 Retrieving answers

In order to retrieve an answer, we used cosine similarity between two vectors:

$$cos\_sim(\mathbf{q}, \mathbf{r}) = \frac{\mathbf{q}^\intercal \mathbf{r}}{||\mathbf{q}||||\mathbf{r}||} \tag{2}$$

This is similar to the methods used by Ji et al. [8] where $\mathbf{q}$ and $\mathbf{r}$ are TF-IDF vectors of query and the considered document for matching respectively.

When the input string was classified to 'Casual', our system calculates cosine similarities between the input and the question part of the data and retrieve the candidate answers which are pairs of questions with the top 3 highest scores. Then the model randomly extracts one of the candidate sentences and returns it to the user.

If the model determined to use Washington Post data, the system go through 2 steps. First, it searches top 3 documents base on cosine similarity between the query and the 'Body' of the data. This will retrieve the top 3 most relevant 'Article' or 'Discussion'. Next, if the system searched on 'Article', it calculates cosine similarities between the query and 'Comments' on the article. Based on the score, our model returns one of the top 3 comments. This will find a comment that is similar to the query. An answer is not necessarily similar to the query. However, a query and a response usually have common words, hence this works [8]. If the system searched on 'Discussion', it searches top 3 questions on the document of 'Discussion' we found using the same method. Then the system randomly returns one of the three answers which are paired with the questions. Since it is not guaranteed that the top most one is the appropriate response and in order to increase the diversity of the conversation, we randomly returns one among top 3 candidates.

3.3 Strategies for successful conversations

For a better user experience, we added dialogue strategies on the system. After adopting the dialogue strategies, user evaluations were increased about 0.56 points (Checked on 5th of Aug) which was scored in 1 to 5 scale during the semifinal phase of Amazon Alexa Prize.

First, we used machine-initiative dialogues. Since we can not guarantee which sentences will be retrieved from our model, it is safer to keep the conversation on the dialogue that we already defined. Therefore, we designed multi-rounds dialogues to be yes/no questions as much as possible. This method make the finite state machine for dialogue management be simple and can keep the machine-initiative conversation longer. Also, by suggesting topics that the corresponding dialogues were already defined when the system mislead a user to frustration, the machine can take initiative again and give a better experience to the user. In fact, it is impossible to keep all the conversation to be machine-initiative in general purpose dialogue system. However, dialogues such as 'greeting' which we applied to our system can be helpful for better user retention time.

Second, we made responses that do not closely related to a corresponding query. Current approaches on developing conversational systems are based on maximizing likelihood of outputs under inputs[8, 9, 16–22]. However the real conversation is not done that way. Interactions between humans can retain without strong correlation between utterances. For example, when a member of people talk about sports, all the members in the conversation can talk about different things; their own experiences on sports, articles on sports section they read, their thought about sports or etc. Also, the topic of the

conversation can be shifted to totally different one from time to time. Based on this observation, we designed (1) 'Current affair' dialogue where machine talks about and article that is weakly related to a user query; (2) 'explore-new-topics' by "let's talk" intent or "frustration" intent where machine switches to known topics before being 'exploited in the current topics' by the human partner. This strategy was effective and derived higher user evaluations.

4 Comparative experiments of retrieval models

4.1 Models for comparison

To verify the performance of our retrieval model and effectiveness of each steps, we designed 7 models to be compared.

Retrieval model(RM) This model is the same as we described on Section 3.2. Since the evaluated system did not include finite state machine part, we used SVM classifiers to determine the class of a user query.

Retrieval model with MLP(RMMLP) This model is the same as RM. But in RMMLP, we used multi-layer perceptron(MLP) as a classifiesr instead of the SVM classifiers. The MLP's have 2 hidden layers with each size of 1024 and 256 from lower to higher layer respectively. This model was devised to verify the effectiveness of the SVM classifier compared to the MLP classifier.

Retrieval model without emphasizing keywords(RM-K) This model is RM without the keyword emphasizing part described on Section 3.2.2. We devised this model to verify the role of emphasizing keyword in the model. Since there is no keyword detection process on this model, its context handling method was modified to keep last 2 inputs on QU regardless of CNL.

Retrieval model without context handling(RM-C) This model is RM without the context handling covered on Section 3.2.2. This model was made to verify the effectiveness of the context handling method by keeping QU empty.

Retrieval model without emphasizing keywords and context handling(RM-K-C) This is RM without both of emphasizing keywords and context handling described on Section 3.2.2.

Sequence to sequence(S2S) This is a deep learning model using sequence to sequence LSTM with attention. We use the objective function proposed by Li et al. [16] to avoid commonplace responses. The model was trained on the Twitter corpus we used on the retrieval model and the corpus consists of 380K of question and answer pairs. The model has four LSTM layers for encoder and decoder with 128 nuerons on each hidden layers. Used vocabulary size is 100,000.

Retrieval model only with casual class(RMC) This model is the same as we described on Section 3.2 except for the used data sources. RMC uses only the Twitter corpus which we used to train S2S. This model was devised to compare the user satisfaction between S2S and our retrieval algorithm.

4.2 Results

The evaluation is done by human judgements since automatic metrics do not reflect actual performances of conversational systems [23, 24]. We made a Facebook page where participants can use the conversational systems through text messaging. Each users interacted with one of a system at a time and evaluated it after the end of the conversation. People scored the models in the score of 1(very bad) to 5(very good) for four questions; how coherent was the conversation(Q1), how engaging was the bot(Q2), whether it avoided inappropriate responses(Q3) and overall experience(Q4). The order of the system they talked to was randomized to prevent any bias. The total number of participants was 17.

The results show that using SVM was a good choice for sentence classification. RM with SVM shows better performances than RM using MLP in all categories. We couldn't check the classification accuracies but from user satisfaction, RM using SVM was superior to RMMLP. This result also shows that the computationally light algorithm can outperform the heavy algorithm.

Table 1: Human evaluation of models

Model	Number of participants	Avg number of dialog turns	Question 1	Question2	Question 3	Question 4
RM	10	10.50	2.40*	2.50*	2.70*	2.40*
RMMLP	10	7.90	1.60	1.70	1.70	2.00
RM-K	11	8.64	1.73	1.82	2.45	2.00
RM-C	12	8.50	1.67	2.17	2.33	1.92
RM-K-C	12	9.00	1.67	2.00	2.00	1.67
S2S	11	7.27	1.73	1.91	1.73	2.00
RMC	13	10.92*	1.92	2.08	2.23	2.00

Also, we could verify that our keyword emphasizing and context handling methods are effective. Without both of the components, user satisfaction on overall experience decreased 0.73 points which p value was 0.11 with two-sided t-test. In particular, without the context handling part, the user's score decreased by 0.73 in Question 1 which is about the coherency of conversations. In RM-K, since sentences are unconditionally stored in QU, it also showed a decrease in Q1 score. Taken together, these results demonstrate that our context management method using QU and CNL is valid.

Finally, our retrieval model has a better performance then the deep learning model S2S(See Table 1). Though the difference is not statistically significant(p values of two-sided t-test are larger than 0.05), our retrieval model made higher scores than the deep learning based model on question 1, 2 and 3. We can find that a simple and computationally efficient model using classical methods can outperform the neural-network-based approach. It is a limitation that the hidden layer size of the S2S model was smaller than in other papers[16, 17]. Neverthelses, the performance gains of big models relative to a information retrieval model was small[9, 17]. Therefore, a similar result will be obtained even when a larger S2S model is used.

5 Conclusion

In this work, we designed a conversational system that models dialogue by combining a finite state machine, an information retrieval method, and human-inspired dialogue strategies. For natural language understanding and sentence classification, we used ASK intent model, TF-IDF vector and linear SVM classifiers. We used a variety of data sources to handle various user requests. The retrieval model we designed showed a better performance than neural-network-based model. The proposed context handling method and emphasizing keywords method of the retrieval model showed significant role in achieving better human evaluation. Dialogue strategies that retain machine-initiative dialogue as much as possible and using article data were the keys to achieve a satisfactory user experience.

Acknowledgments

We would like to thank Chihyung Jeon, Heesun Shin, Hanbyul Jeong and Sunguen Kim of Graduate School of Science and Technology Policy in Korea Advanced Institute of Science and Technology for a kind advice about conversation strategy. Also we want to thank Jaemin Cho for sharing his experience on chatbot development and doing paperworks for the Alexa Prize.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Gregory D Abowd, Hung-Ming Wang, and Andrew F Monk. A formal technique for automated dialogue development. In *Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques*, pages 219–226. ACM, 1995.

[3] David Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai. A form-based dialogue manager for spoken language applications. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 701–704. IEEE, 1996.

[4] Michael F McTear. Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. *development*, 5(7), 1998.

[5] Stephanie Seneff and Joseph Polifroni. Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 11–16. Association for Computational Linguistics, 2000.

[6] Antoine Raux and Maxine Eskenazi. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 629–637. Association for Computational Linguistics, 2009.

[7] Sina Jafarpour, Chris Burges, and Alan Ritter. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking*, 10, 2010.

[8] Zongcheng Ji, Zhengdong Lu, and Hang Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*, 2014.

[9] 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. *arXiv preprint arXiv:1506.06714*, 2015.

[10] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[11] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[12] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM, 2013.

[13] Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, and Markus Dreyer. Just ask: Building an architecture for extensible self-service spoken language understanding. *arXiv preprint arXiv:1711.00549*, 2017.

[14] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[15] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[16] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

[17] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784, 2016.

[18] Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 18–27. Association for Computational Linguistics, 2009.

[19] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

[20] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[21] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[22] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[23] Amanda Stent, Matthew Marge, and Mohit Singhai. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer, 2005.

[24] Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*, 2016.