The Latest Progress in Human-Computer Dialogue System

Yanbo Feng
Central China Normal University Wuhan, China
fyb@mails.ccnu.edu.cn

Abstract. The Human-Computer dialogue system allows a computer to make conversations with humans through natural languages, and some designs can accomplish tasks given by humans. The development of the Human-Computer dialogue system can significantly expand the range of users for computers. The profound ELIZA Program designed scripts containing keywords and their corresponding rule-based sentence transformation to respond to the user. As the computation power increases, Human-Computer dialogue systems are handling more varied and complicated scenarios by introducing deeper and more complex artificial neural networks. The designer of a task-oriented system can choose between the pipeline method and the end-to-end method. The pipeline method is basically a pipeline that lets the user input go through three parts handling Natural Language Understanding, Dialogue Management, and Natural Language Generation, to generate an appropriate response back to the user. The end-to-end method, however, uses joint models to allow the parts to interact with others and get more efficient in handling the information. With the development of ChatGPT, more general language models and more varied methodologies are becoming more prevalent in dialogue systems.

Keywords: Human-Computer dialogue system, pipeline method, end-to-end method, non-task-oriented dialogue system.

1. Introduction

Ever since the electronic computer was invented, the gap between its high computing speed and the relatively low efficiency in interaction with its users was always a severe bottleneck in the application of computers. Expertise in the structure of computers was required to write programs in binaries in the earliest days. Then programming languages and command-line interfaces (CLI) were invented to simplify the use of computers. The command-line interface on a computer screen consists of lines of text. The user can input commands designed using mnemonics to launch programs, and the computer will respond to the user with the text output synthesized by text templates pre-defined in the program. Then the graphics user interface (GUI) was popularized for normal users who only have a little knowledge of the principles of computers. To expand the application ranges of computers, there have been many works exploring the possibility of enabling computers to interact with users in natural languages.

The Human-Computer dialogue system receives input from a user in the form of natural language in text or speech, attempts to understand what the user means, and then produces a suitable output in natural language and returns it to the user. Some Human-Computer dialogue systems are designed to help the user to accomplish some pre-defined tasks like booking a ticket for a train or creating an alarm clock through the user's voice. Other systems are just designed to talk with users without a fixed goal. The systems designed with specific tasks were named task-oriented dialogue systems, while those without tasks were named non-task-oriented dialogue systems.

The most straightforward idea for designing a dialogue system to talk with humans is to let programmers iterate each circumstance possibly reached by the user and write corresponding code to generate a perfect response. To reduce the requirement for work amount, some general rules can be applied because of the patterns in human language. The ELIZA program [1] was the first design that pretended itself as a psychotherapist simply by a set of sentence transformation rules instead of making a real comprehension of the user input. This program includes two parts. One part is a fixed program, which detects keywords in the input to select a decomposition template from the set of rules in the script and then uses matched words and the reassembly rule specified by the template to

generate a response. The other part is the ELIZA script, a set of rules written by the designer of this program based on patterns in natural language. The separation of the program core as code and the ELIZA script as data enables the ELIZA program to be edited and improved easily. Though it was clear that the computer running it did not understand the semantics in the user's input at all but made mere string processing to create an illusion of dialogue, the program was still a pioneer hinting at the possibility of designing a dialogue system to talk in natural languages with humans.

After the invention of machine learning technology, language models are capable of mining knowledge automatically from data consisting of a giant number of examples given by the AI trainer. Later scholars increased the depth and complexity of neural networks and developed the deep learning approach, achieving higher accuracy and wider ranges of use. The functionality and performance of Human-Computer dialogue systems got improved massively during this period. ChatGPT, a chatbot developed by OpenAI, was announced recently, surprising the public a lot with its ability to comprehend normal conversation and solve simple tasks about natural languages.

There have been some reviews of Human-Computer dialogue systems. Zhao et al. [2] described the pipeline method and the end-to-end method in task-oriented dialogue systems but did not discuss the advances that happened in systems that were not designed for pre-defined tasks. Wang & Yuan [3] also focused on task-oriented systems and described the development of joint models concretely. Navigli [4] proposed a review of natural language understanding, which was used as a stage in the pipeline method. Semaan [5] reviewed natural language generation but didn't introduce other parts of a dialogue system either. Chen, Ma, and Wang [6] mentioned both task-oriented and non-task-oriented systems but only briefly introduced the stages in the pipeline method.

This paper describes the background and progress of the Human-Computer dialogue system. The methods are split into two sections. Section II introduces the methods used in task-oriented systems. Section III presents the methods used in non-task-oriented systems. Then Section IV contains an analysis of the direction of future development in this area.

2. Task-oriented dialogue systems

2.1. Pipeline Method

The pipeline method splits the process of dialogue into three steps and handles each step individually, i.e., the system takes the input from the user, tries to extract the semantic information from the input, uses a policy to manage the status of the dialogue, and finally generates an output back to the user. These three steps are named respectively as Natural Language Understanding (NLU), Dialogue Management (DM), and Natural Language Generation (NLG) [1]. Fig. 1 shows the design of the architecture of the pipeline method. The output from one stage is used as input for the next stage. When the dialogue is vocal, however, the system adds two additional parts, Automatic Speech Recognition (ASR) and Text-To-Speech (TTS), to translate between voice and text. The following texts are descriptions of the core three steps.

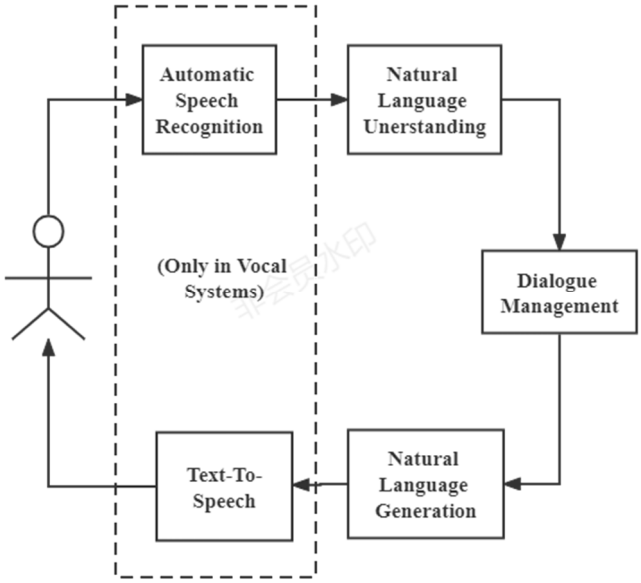

Fig. 1 The modern architecture of the pipeline method (Photo credit: Original)

2.1.1. Natural Language Understanding (NLU)

Natural Language Understanding means extracting information from the user's words. The information is later used in the next step. There are often three subgoals in the step of NLU: identifying the domain of the user's discourse, understanding the intent of the user's words, and filling the semantic slots for the output to the next step [2]. The steps are shown in Fig. 2. The identification of the domain and the intent of the user is a typical classification task, while the semantic slot filling is typically viewed as a sequence labeling task. Multiple attempts with different machine-learning models, including KNN [4,6], Support Vector Machine (SVM)[7], and Deep Neural Networks (DNN)[8], etc. were made to enhance the performance of these two identification tasks located at the beginning of the entire pipeline. The accuracy has been improved to a relatively high level of more than 90% in some corpora[3]. Similarly, the accuracy of the semantic slot-filling task has also reached above 90% with models like Recurrent Neural Networks (RNN) [9-10]. Other efforts in slot filling like Conditional Random Fields (CRF) [11-12], LSTM [13-14], and Gate Recurrent Unit (GRU) [15] are also improving the accuracy continuously.

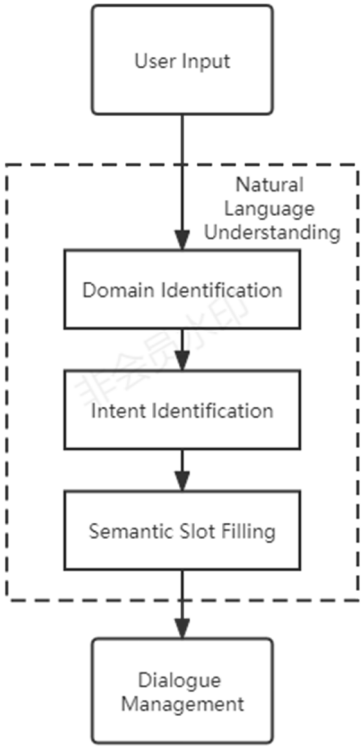

Fig. 2 The steps inside the Natural Language Understanding (Photo credit: Original)

2.1.2. Dialogue Management (DM)

Dialogue Management has an awareness of the dialogue context across many rounds of interactions and makes decisions about the reaction that the system needs to perform. It takes the input from the Natural Language Understanding and hints at its next stage, Natural Language Generation, to generate answers. As shown in Fig. 3, this stage is usually split into two parts [16]. One is the dialogue state tracking part that accumulates the understanding of preceding conversations in the dialogue, allowing the dialogue system to keep the context in the entire dialogue. The other part is the dialogue policy that tells the system the next move in a conversation. There are successful results in training the module of dialogue management both in supervised learning and in unsupervised learning. The most primeval supervised learning method is the Finite State Machine (FSM), which uses a state-transfer model defined manually by experts and automatically tunes the parameters in the model. To avoid constructing models by handcrafting, some models, such as POMDP, have been proposed based on the method of reinforcement learning. As deep reinforcement learning develops, more models like the DQN model [17] based on deep reinforcement learning were proposed, improving the performance significantly.

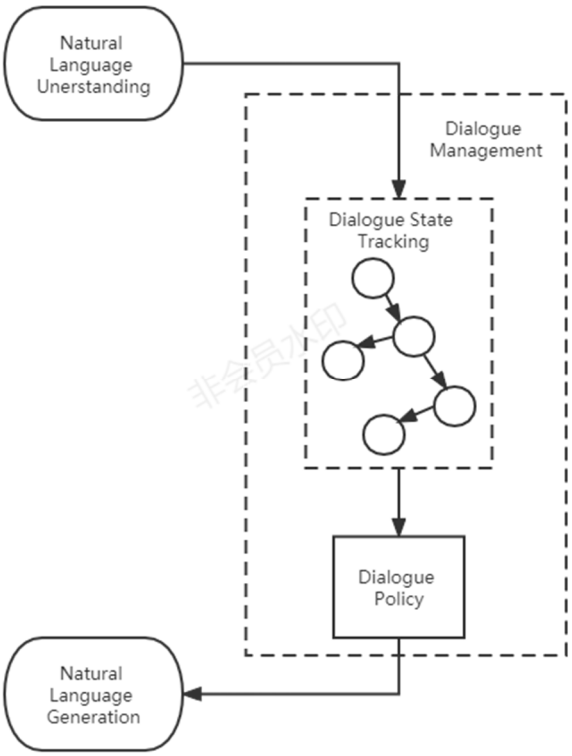

Fig. 3 The steps inside the Dialogue Management (Photo credit: Original)

2.1.3. Natural Language Generation (NLG)

Given the action picked by the dialogue policy, the dialogue system then needs to translate the action into natural languages, which is the reason why Natural Language Generation becomes the final stage of the pipeline. This step takes the input from the stage of Dialogue Management and outputs it to the user in natural language. The generation can be made either by templates or by machine learning. Given the requirement for stability and accuracy in task-oriented systems, the method of template-based generation [18] is still used. The problems with using such fixed templates are maintainability and portability. A model depending on templates written by hand is hard to maintain and needs to rework in every domain [2]. To overcome this problem, later systems use trainable models to generate sentences dynamically through steps including content planning, sentence planning, and surface realization. The steps are shown in Fig. 4. Firstly, the content planning stage generates a set of appropriate information according to the dialogue policy given by the previous part of the dialogue pipeline. Secondly, the sentence planning stage chooses a structure, usually represented as a tree, with the required linguistic resources to achieve the content plan. Finally, the surface realization stage translates the response from the tree representation to natural language, which humans can understand without expertise in computers. Though this approach can generate flexible and complex sentences with the support of a knowledge database, the quality of generated discourses in specific domains or contexts is still likely to be lower than the template-based method. The corpus-based and the phrase-based method were then proposed to improve the text quality [2]. Later, as the linguistic data accumulated and the deep learning approach rose, more advanced models based on the transformer architecture got used, removing the need for handcrafting templates of expected responses.

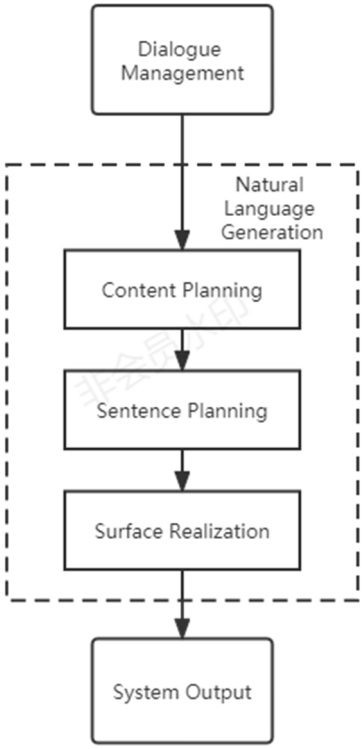

Fig. 4 The steps inside the generative method in NLG (Photo credit: Original)

2.2. End-to-end method

Unlike the pipeline method, which separates the conversation into three steps and makes the data fall through the pipeline in one direction, the end-to-end method, in contrast, treats the process in joint models, which realize analogous functionalities with one or several tightly coupled models. By integrating the separated models together, the end-to-end method allows the interaction across the original model boundaries for better performance, makes the errors easier to get controlled, and avoids the problem of retraining a part in a long pipeline [3].

Though the idea of using one single model for the whole process in conversation seems simple, mixing just several models in the pipeline method can also achieve part of the goals. This section first talks about the end-to-end models combining the tasks in each stage of a pipeline. Then the models mixing across the pipeline stage borders are introduced. The methods of single models are put at last.

In the stage of Natural Language Understanding, a variety of supervised methods were applied, given their better performance achieved with labeled data, compared to the unsupervised alternatives [3]. Methods like CNN, RNN, Conditional Random Field (CRF), and even a combination of several methods was used in NLU [3]. Given the supervised methods cost labeled data when training, introducing pre-trained models here to reduce the need for vast handcrafted labels may be a direction for future development.

Dialogue Management, including the dialogue state tracking part and the dialogue policy part, was also developed with joint models. Zhao & Eskenazi [19] proposed a joint model through Deep Reinforcement Learning (DRL), implementing the two parts together. There were also other models combining Dialogue Management with other stages in the pipeline.

The efforts in joint models for the Natural Language Generation were also non-negligible. Many of them were based on RNN and LSTM.

When the borders between models separated by the pipeline method are blurred by the joint models, more integrated models get invented. The model proposed by Cuaahuitl[20] could directly take the text translated by ASR as input and was able to output the decision of dialogue policy to drive the

NLG stage. Other models are even more complete as they can directly generate the text in natural language as output.

3. Non-task-oriented dialogue systems

3.1. Retrieval-based method

The retrieval-based method sounds like the template-based text generation introduced in task-oriented systems. Unlike the high demand for stability and accuracy in task-oriented systems, users of non-task-oriented systems often expect the computer to make a more flexible and vivid response. Thus, the retrieval-base method, which means that the computer selects among a limited set of responses handcrafted by its designers, exposes more limitations compared to the task-oriented equivalent. In the simplest implementation, the output of the system was only determined by the current user input without any relevance to the preceding context [6]. More complicated systems appeared later, dealing with longer contexts consisting of multiple rounds of conversations.

3.2. Generative method

With the research progress in machine learning, the generative method is improving its performance continuously. Seq2Seq, the most typical model in the generative method, is widely applied in multiple NLP-related areas, including machine translation, text generation, voice recognition, text recognition, and chatbots. The framework of the Seq2Seq model is illustrated in Fig. 5. The Seq2Seq model iterates the input sequence from the start to the end. Each time the model reads one token, translates the token to an intermediate vector representation through an encoder, then translates the representation back to another token through a decoder, and finally its output is also happening one token at a time [21]. There have been works making use of such encoder-decoder models based on RNN and LSTM on non-task-oriented models.

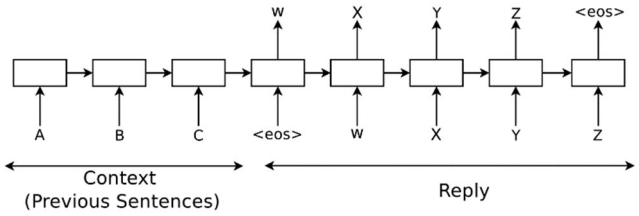

Fig. 5 The framework of the Seq2Seq model [21]

In 2022, a chatbot named ChatGPT attracted much attention on the Internet [22]. This chatbot developed by OpenAI was able to write long, coherent, and elegant paragraphs in multiple languages. When required by the user, it was also able to play interactive games through the textbox. Someone even constructed a virtual machine by requiring the ChatGPT to act as a Linux terminal waiting for users' input. The appearance of ChatGPT was a great surprise for the public compared to the simpler implementations that were applied in production before. ChatGPT was trained by multiple steps consisting of both supervised learning and reinforcement learning [23]. Fig. 6, Fig. 7, and Fig. 8 show the three steps used to train ChatGPT.

Step 1 was fine-tuning GPT3.5, the profound general pre-trained transformer model, with the real conversational data played by humans on both sides, the user and the chatbot.

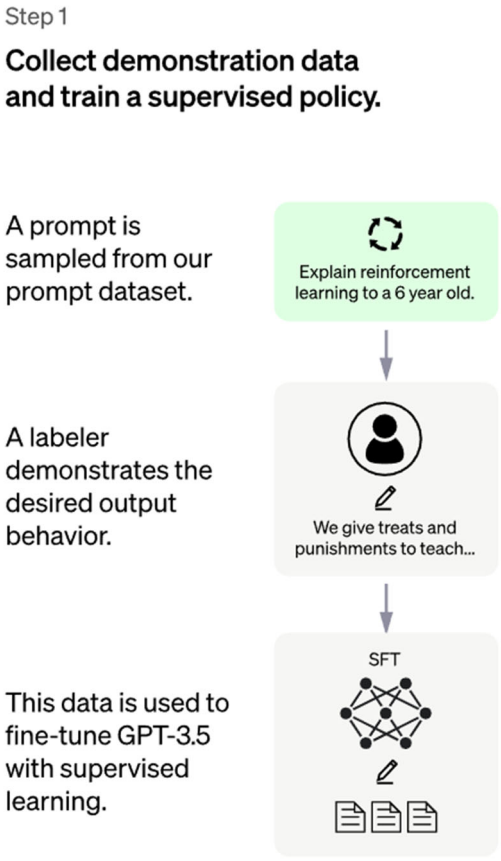

Fig. 6 Step 1 in training ChatGPT [23]

Step 2 was also supervised. The researchers in OpenAI used different models to get samples of replies at different levels of quality. The models used here included the GPT3.5 model tuned in the first step and InstructGPT, a predecessor of ChatGPT, which used similar methods to train. The conversational samples from these models were merged and labeled by humans according to the qualities of the replies. With these conversational data labeled by qualities, the researchers got able to train a reward model (RM) in judging the quality of a conversational reply.

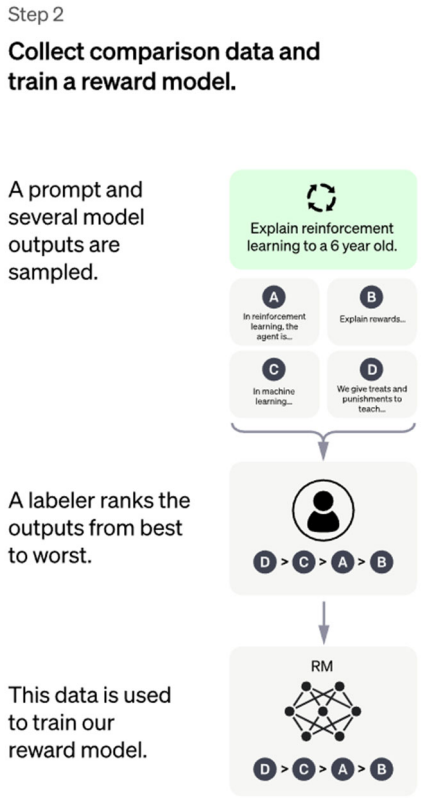

Fig. 7 Step 2 in training ChatGPT [23]

Step 3 made use of Proximal Policy Optimization (PPO), the reinforcement learning algorithm developed by OpenAI, to generate an optimized policy by reinforcement learning against the reward model trained in the second step.

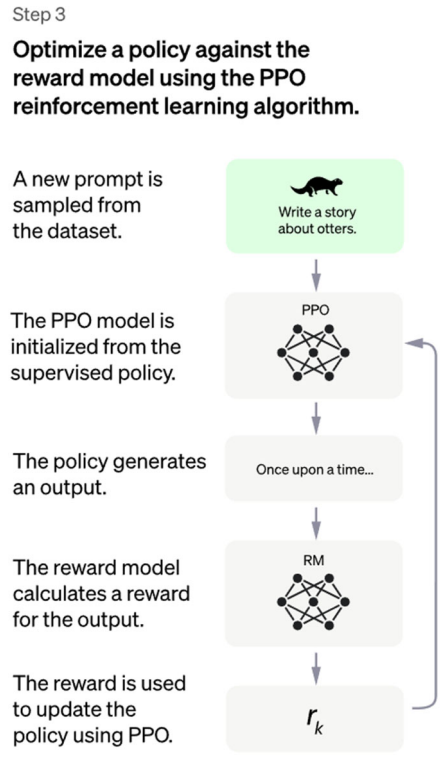

Fig. 8 Step 3 in training ChatGPT [23]

4. Future work

There have been many suggestions about the direction of future development in the existing literature. Zhao [2] indicated three possible directions: low-resource bootstrapping, domain adaptability, and the introduction of domain knowledge. Wang & Yuan's review [3] pointed out that joint models would be the next step in the development of Human-Computer dialogue systems. Chen, Ma, and Wang's review [6] viewed the personalization of dialogue systems, the application of the Knowledge Graph, and the method of evaluation as the future directions.

With the appearance of ChatGPT, the giant potential in pre-trained language models is exposed to researchers in this area. Introducing a pre-trained model can not only solve the problems of reducing the requirement of the initial dataset but also brings adaptability across multiple domains given its oceanic dataset used in pre-training. The ChatGPT also showed high flexibility in language styles of different cultural customs and an ability to make naive logical reasoning in conversations. The challenges proposed by the reviews listed above can all be relieved by making use of such a pre-trained language model in future Human-Computer dialogue systems.

However, the pre-trained model is not the solution to everything. There are still several problems to be solved before it can expand the range of applications:

1) Model size. The GPT3 model contains hundreds of billions of parameters and costs many computing resources to train or even to forward. With the excellent performance achieved by ChatGPT, the prospect of reducing the demand for computing resources after constructing a dialogue system using GPT is attractive.

2) The representation of domain knowledge for task-oriented dialogue systems. Though GPT has oceanic linguistic knowledge, task-oriented systems are still required to handle domain knowledge in tasks. Exploring a general framework for the representation of domain knowledge becomes significant.

3) Complex logic. Though ChatGPT can solve simple logic and arithmetic problems now, the ability to handle complex logic is still important to be widely used. The GPT model might be integrated with a logic engine to enhance its ability in this area.

5. Conclusion

This paper described the progress of the dialogue systems between humans and computers. The dialogue systems could be designed with or without pre-defined tasks. In dialogue systems with pre-defined tasks to accomplish, the computation was split into three subgoals: NLU, DM, and NLG. The pipeline method sequentially reached all the subgoals in stages. The subgoals of the three stages in a modern design of the dialogue pipeline were interpreted in the section on the pipeline method. The end-to-end method solved the problem with joint models merging the stages together. In the section on non-task-oriented systems, the retrieval-based method and the generative method were explained. The latest advances like ChatGPT in this approach were also included. In the last, several directions for future work were given.

References

[1] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," Communications of the ACM, vol. 9, no. 1, pp. 36–45, 1966.

[2] Y. Y. Zhao, Z. Y. Wang, P. Wang, T. Yang, R. Zhang, and K. Yi, "A survey on task-oriented dialogue systems," Chinese Journal of Computers, vol. 43, no. 10, pp. 1862–1896, 2020.

[3] X. Wang and C. Yuan, "Recent advances on human-computer dialogue," CAAI Transactions on Intelligence Technology, vol. 1, no. 4, pp. 303–312, 2016.

[4] R. Navigli, "Natural language understanding: Instructions for (present and future) use." in IJCAI, vol. 18, 2018, pp. 5697–5702.

[5] P. Semaan, "Natural language generation: an overview," J Comput Sci Res, vol. 1, no. 3, pp.50–57, 2012.

[6] J. P. Chen, J. H. Ma, and Y. J. Wang, “A survey of human-machine dialogue systems based on multiple rounds of interaction,” Journal of Nanjing University of Information Science Technology (Natural Science Edition), vol. 11, no. 3, pp. 256–268, 2019.

[7] P. Haffner, G. Tur, and J. H. Wright, “Optimizing svms for complex call classification,” in 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03)., vol. 1. IEEE, 2003, pp. I–I.

[8] R. Sarikaya, G. E. Hinton, and A. Deoras, “Application of deep belief networks for natural language understanding,” IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 4, pp. 778–784, 2014.

[9] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding.” in Interspeech, 2013, pp. 2524–2528.

[10] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu et al., “Using recurrent neural networks for slot filling in spoken language understanding,”IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 3, pp. 530–539, 2014.

[11] J. Lafferty, A. McCallum, and F. C. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” 2001.

[12] F. Peng and A. McCallum, “Information extraction from research papers using conditional random fields,” Information processing & management, vol. 42, no. 4, pp. 963–979, 2006.

[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” Neural computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[15] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” arXiv preprint arXiv:1409.1259, 2014.

[16] Y. J. Zhao, Y. L. Li, and M. Lin, “A review of the research on dialogue management of task-oriented systems,” in Journal of Physics: Conference Series, vol. 1267, no. 1. IOP Publishing, 2019, p. 012025.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” nature, vol. 518, no. 7540, pp. 529–533, 2015.

[18] L. Baptist and S. Seneff, “Genesis-ii: A versatile system for language generation in conversationalsystem applications,” in Sixth international conference on spoken language processing, 2000.

[19] T. Zhao and M. Eskenazi, “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning,” arXiv preprint arXiv:1606.02560, 2016.

[20] H. Cuayáhuitl, “Simpleds: A simple deep reinforcement learning dialogue system,” Dialogues with Social Robots: Enablements, Analyses, and Evaluation, pp. 109–118, 2017.

[21] O. Vinyals and Q. Le, “A neural conversational model,” arXiv preprint arXiv:1506.05869, 2015.

[22] F. Sun, “Chatgpt, the start of a new era,” 2022.

[23] https://openai.com/blog/chatgpt/, 2022.