Journal of Organizational and End User Computing
Volume 36 • Issue 1

Intelligent Customer Service System Optimization Based on Artificial Intelligence

Zhong Wu, Wuhan Business University, China
Qiping She, Hubei University of Economics, China*
Chuan Zhou, Wuhan Institute of Shipbuilding Technology, China

ABSTRACT

To elevate the intelligence of customer service dialogue systems, this article proposes an intelligent customer service system comprising chat dialogue subsystems, task-oriented multi-turn dialogue subsystems, single-turn dialogue subsystems, and an integration model. Firstly, to enhance diversity of responses and improve user experience, particularly in casual chat scenarios, this article presents a Seq2Seq-based approach for multi-answer responses, allowing for more expressive emotional expression in responses. Secondly, to address situations where customers cannot articulate their needs in a single sentence during multi-turn dialogues, this article designs a task-oriented multi-turn dialogue module. It employs intent recognition and slot filling to maintain contextual information throughout the conversation, aiding customers in problem resolution. Lastly, to overcome the current limitation of intelligent customer service models providing relatively one-dimensional answers in specific domains.

KEYWORDS

Chat Dialogue Subsystems, Intelligent Customer Service, Seq2Seq, Single-Turn Dialogue Subsystems, Task-Oriented Multi-Turn Dialogue Subsystems

1. INTRODUCTION

In recent years, with the rapid evolution of artificial intelligence and the continuous progress in natural language processing (NLP) technology, intelligent customer service dialogue systems have emerged as a focal point in the NLP field (Nie et al., 2021). Originating from expert knowledge bases, these systems are designed to assist users in answering questions and enhance response efficiency (Ji & Zhang, 2023). Considered a valuable product for human-machine interaction, intelligent customer service dialogue systems have garnered substantial attention from both industry and academia. An intelligent customer service dialogue system is a computer program designed to parse and comprehend user queries through modules such as natural language understanding (NLU), natural language generation, and dialogue management. Importantly, this entire process involves minimal human intervention (Wang et al., 2023a or b). In comparison to traditional human-operated customer service,

DOI: 10.4018/JOEUC.336923 *Corresponding Author

intelligent customer service dialogue systems exhibit robust capabilities, enabling them to access high-quality, accurate, comprehensive, in-depth, and real-time information from extensive knowledge databases effectively. This capability bridges the significant gap between users' precise and diverse information needs and the vast scale of internet data (Cai et al., 2016). Additionally, they enhance user interaction, ultimately improving the overall user experience. Intelligent customer service holds significant application potential, making research on enhancing the intelligence of customer service systems a focal point in current studies (Xiao & Kumar, 2021).

The earliest research on intelligent dialogue systems can be traced back to 1950 when Alan Turing, known as the "father of artificial intelligence," posed the question of whether machines could communicate in natural language like humans. This question later evolved into the famous Turing Test, considered the ultimate goal of artificial intelligence (Chen et al., 2017). The first chatbot system, developed by Joseph Weizenbaum at the Massachusetts Institute of Technology in 1966, was ELIZA (Al-Rfou et al., 2016), designed to emulate a psychotherapist for clinical therapy. This dialogue system primarily relied on keyword matching and manually crafted response rules. Subsequently, in 1988, a chat dialogue system named UNIX Consultant was developed by researchers, including Robert Wilensky from the University of California, Berkeley. It aimed to assist users in learning UNIX system administration. In 1995, Richard S. Wallace, inspired by ELIZA, created the renowned ALICE system, known for its heuristic template matching dialogue strategy. It is considered one of the best-performing systems of its kind. The ALICE system rapidly constructs a robust and relatively flexible retrieval-based dialogue system using the artificial intelligence markup language (AIML). The methods proposed in the aforementioned literature are all based on matching mechanisms to achieve intelligent dialogue. Even in cases where question-answer pairs have not been preconfigured, this type of dialogue system still requires a significant amount of manual effort (Lowe et al., 2015). Currently, the challenge in achieving highly intelligent dialogue systems lies in the fact that customer conversations typically involve context and subtle language nuances, making it difficult for existing intelligent customer service systems to effectively simulate human thought processes during conversations. Furthermore, dialogue content may consist of incomplete sentences, topic shifts, and disorganized discourse, further complicating the ability of current intelligent customer service systems to fully meet user requirements (Bai et al., 2022). Therefore, further research and optimization of intelligent customer service systems are considered imperative.

To enhance the intelligence of customer service dialogue systems, NLP plays a pivotal role. NLP is instrumental in elevating the intelligence of customer service dialogue systems by enabling them to understand the intricacies of human language, including context, semantics, and nuances. Through sophisticated algorithms, NLP empowers these systems to interpret ambiguous expressions, handle variability in language, and discern the true intent and sentiment behind customer queries. This capability not only facilitates personalized interactions based on user preferences but also ensures efficient information retrieval and continuous learning, allowing the systems to adapt and improve over time. Ultimately, NLP plays a pivotal role in making customer service dialogue systems more responsive, accurate, and attuned to the diverse and dynamic nature of human communication. In order to enable customer service dialogue systems to engage with customers across various topics without being constrained by specific domains and to approach interactions resembling those with real people, research has been conducted in the domain of deep learning in NLP (Fountain et al., 2019). Methods have been optimized by Serban et al. (2016), employing recurrent neural networks, to avoid the manual feature extraction steps. Vinyals was the first to introduce the Sequence-to-Sequence (Seq2Seq) model, an end-to-end encoder-decoder model, into open-domain dialogue systems (Vinyals & Le, 2015). By learning distributed representations of input sequences, Seq2Seq models capture semantic information, contributing to their proficiency in discerning user intent and generating contextually appropriate replies. Additionally, their training paradigm on pairs of input-output sequences aligns seamlessly with the nature of conversational data, making Seq2Seq models a versatile and widely applied solution in the realm of NLP and chat-based applications. As a result,

Seq2Seq has gradually become one of the most popular models for open-domain dialogue systems. However, there is currently no existing literature addressing how to integrate such NLP models with the requirements of customer service dialogue systems.

This article aims to optimize the three types of conversational systems mentioned above by employing a hybrid fusion mechanism based on rules and models. It harnesses the strengths of these systems to craft an intelligent customer service dialogue system that is accurate, efficient, and comprehensive, incorporating both deep learning and software architecture design principles. The primary contributions of this article include:

(1) Addressing the limitation of current intelligent customer service systems, which can only provide single answers; the paper proposes a response method for chat scenarios based on Seq2Seq that offers multiple answers. This approach enhances response diversity, improves the user experience, makes responses more engaging, and brings them closer to human emotional expressions.
(2) To tackle situations where users cannot fully express their requests in a single sentence, the paper introduces a task-oriented multi-turn dialogue module. This module employs intent recognition and slot filling to capture contextual information in the conversation, thereby assisting users in problem-solving.
(3) Given the relatively narrow scope of current intelligent customer service models, the paper suggests integrating subsystems from multiple scenarios using a hybrid fusion mechanism based on rules and models. It combines domain-specific answers and presents corresponding responses. Finally, it designs a multi-scenario intelligent customer service system architecture, creating a complete, feature-rich, efficient, and rational customer service dialogue system.

2. BACKGROUND

2.1. Word Embedding Model

In advanced NLP tasks, the transformation of natural language into machine-understandable language involves tokenizing the text and representing the tokenized results as word vectors, a process known as word vectorization. Within statistical learning models, the utilization of word embeddings is a pivotal technology in NLP, benefiting significantly from recent advancements in deep learning techniques. Currently, two main methods exist for representing words in natural language: one-hot encoding and distributed representation.

One-hot encoding, also known as one-hot representation (Tian et al., 2024), is a common encoding method. The initial step in one-hot encoding involves extracting the vocabulary from the corpus, typically performed after tokenizing the text and removing stop words. This method lists all words in the text without considering word order, assigning each word a unique ID. The one-hot representation represents each word with a vector of the same length as the vocabulary, where the element at the index corresponding to the vocabulary ID is set to 1, and all other elements are set to 0. While one-hot representation is discrete and results in a sparse matrix of word vectors, its drawback lies in significant storage space consumption, and it lacks the ability to capture relationships between word vectors. This limitation means that frequently co-occurring words or synonyms cannot be effectively represented in the one-hot encoding.

Furthermore, research contributions from Tomas Mikolov, Kai Chen, and others have demonstrated the effective use of neural networks to train word vectors with semantic representation capabilities on large-scale corpora (Mikolov et al., 2013). They have also open-sourced the Word2Vec project. Word vectors trained through distributed representations find applications in various practical NLU tasks, such as document vectors, semantic prediction, semantic shift analysis, and sentence similarity comparison. Word2Vec maps words into dense vectors, enabling the measurement of relationships between words by calculating the distance between their vectors.

Word2Vec employs two training strategies: Continuous Bag of Words (C-BOW) and Skip-gram. C-BOW predicts the probability of the current word appearing based on the context, with the leftmost input layer representing the current word's context words using one-hot word vectors. The input includes not only the words before the current word but also those after it. The output layer on the right outputs one-hot word vectors of the same dimension. In contrast, the Skip-gram model predicts the context given the current word, essentially reversing the C-BOW pattern. The network's input contains only one word, and the output is the predicted context for that word.

2.2. BERT Model

The pre-trained language model, Bidirectional Encoder Representation from Transformers (BERT), introduced by Google in 2018, has exhibited outstanding performance across 11 different datasets in various NLP tasks (Devlin et al., 2018). It is regarded as the initiation of a new era in NLP. BERT, serving as the foundational language model, finds application in a diverse array of specific tasks, including sentiment analysis, named entity recognition, question-answering systems, text classification, and more. Beyond being a language model, BERT, owing to its potent semantic representation capabilities, stands as a universal semantic representation model.

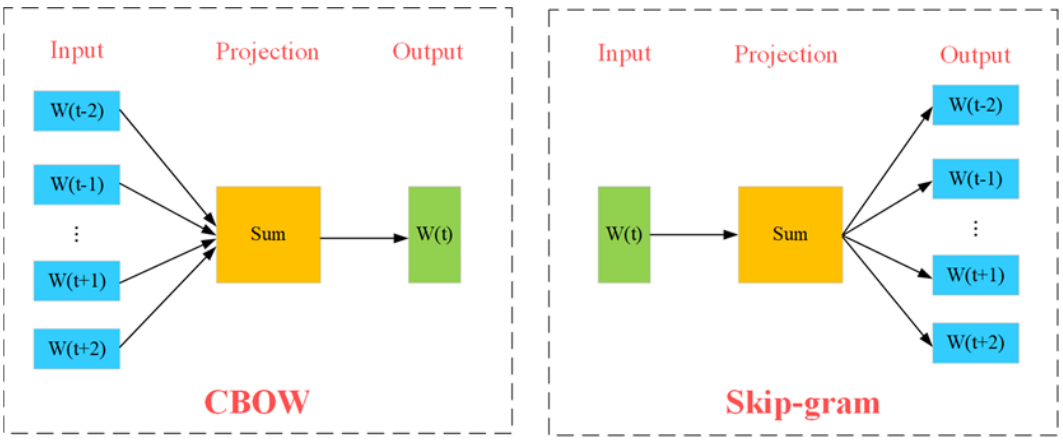
BERT employs the Masked Language Model (MLM) approach, randomly masking certain words to predict the masked portions based on contextual information before and after the masked words (Yan et al., 2023). In contrast to traditional left-to-right single-direction language model fine-tuning methods, MLM integrates context and emphasizes word-level semantic relationships. Furthermore, BERT introduces the "Next Sentence Prediction" task to forecast the content of the next sentence, establishing meaningful contextual relationships at the sentence level. BERT's model architecture utilizes a multi-layer bidirectional Transformer network structure, ensuring comprehensive connections to contextual information across all network layers.

2.3. Recurrent Neural Network Model

2.3.1. LSTM Model

The Long Short-Term Memory (LSTM) model, a type of recurrent neural network (RNN) introduced by Hochreiter and others in 1997 (Hochreiter & Schmidhuber, 1997), distinguishes itself by incorporating gate mechanisms within its recurrent units when compared to traditional RNN structures. These gates dynamically control the flow of information within the recurrent loop, with their weights adjusted

Figure 1. The CBOW and skip-gram model



based on the context. This architectural enhancement effectively tackles the "long-term dependency problem" by preserving crucial information from the sequence's outset.

From Figure 2, it's evident that externally, LSTM appears similar to RNN, with the pivotal changes occurring internally, particularly when calculating ht and Ct. The fundamental concept of LSTM is to capture and retain the state of each recurrent unit. The Ct, traversing through each recurrent unit, signifies the information that requires addition or removal after passing through each neural unit. This ensures that vital information seamlessly traverses the entire recurrent unit, allowing for the normal propagation of gradients, especially in cases of extended sequence lengths. This addresses challenges related to gradient explosion or vanishing.

2.3.2. GRU Model

The Gate Recurrent Unit (GRU) stands as another variant of recurrent neural networks, introduced by Cho et al. in 2014 (Chung et al., 2014). Serving as a streamlined version of LSTM, GRU reduces the three gates in LSTM to two, resulting in reduced computational complexity and enhanced training speed. The model structure of GRU is depicted in Figure 3.

Both structurally and in design principles, GRU and LSTM share close ties, as they both aim to address the challenge of handling long-range dependencies that traditional RNNs often struggle with. While maintaining the effectiveness of LSTM, GRU simplifies its structure. In contrast to LSTM, GRU employs only two gates: the update gate and the reset gate.

The update gate governs the extent to which the previous recurrent unit's state information is integrated into the current recurrent unit's state. A higher value of the "update gate" implies that more of the previous recurrent unit's state information is included in the current state. Conversely, the reset gate regulates the degree to which the previous recurrent unit's state information is disregarded. A smaller value of the reset gate signifies that more of the previous unit's information is neglected.

Both GRU and LSTM leverage gating mechanisms to retain past information and facilitate its effective propagation over long dependencies. Beyond these similarities, GRU holds an advantage in terms of propagation speed, featuring one fewer gating function compared to LSTM. This results in faster training and a reduction in parameters within the GRU model, simplifying the training process and enhancing efficiency. Historical experience with training data indicates that, in scenarios involving moderately sized datasets, GRU strikes a good balance between training effectiveness and speed. However, when dealing with larger datasets, LSTM tends to yield superior training results.

Figure 2. LSTM model

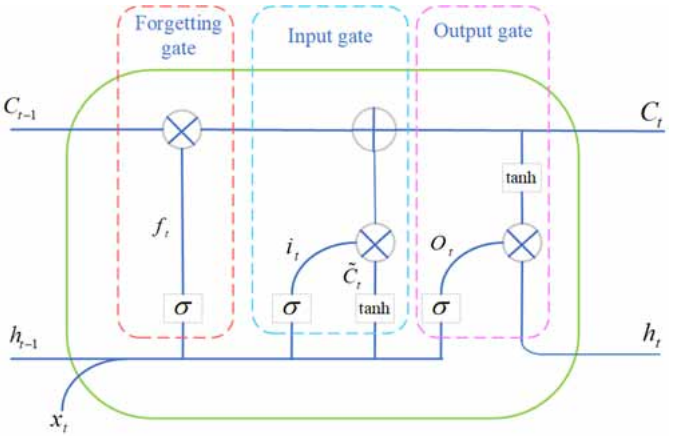
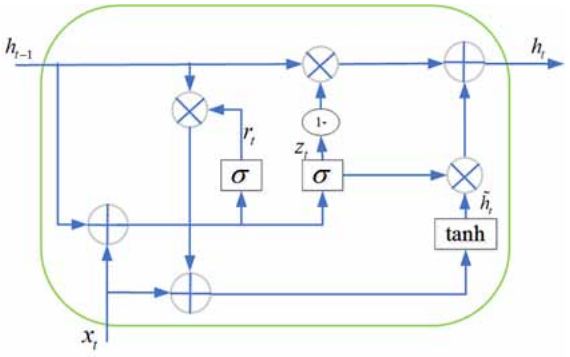


Figure 3. GRU model



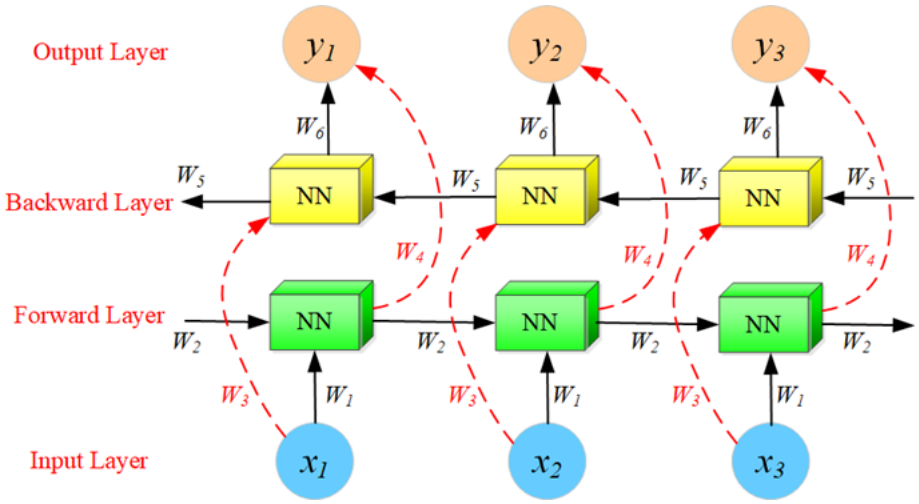
2.3.3. Bidirectional Recurrent Neural Network

While RNNs, including LSTM and GRU, have proven effective in capturing long-range dependencies and features within sequences, they traditionally operate in a unidirectional manner. This implies that they predict the current state solely based on the preceding state, and their predictions are contingent on past states. To mitigate the uncertainty associated with predicting future outcomes using unidirectional models, Schuster and others introduced bidirectional recurrent neural networks (BiRNN) (Schuster & Paliwal, 1997).

This innovative structure combines information from both past and future contexts to better predict the current state. By utilizing bidirectional recurrent neural networks, the model can incorporate information from preceding and subsequent contexts, enhancing its ability to generate more accurate current state predictions. The model's architecture is illustrated in Figure 4.

From the structure of the BiRNN, it's evident that the output results are jointly determined by both the forward and backward RNNs, encompassing information from both the preceding and succeeding elements of the input sequence. The diagram illustrates six shared weight values, which are reused at each step of the process. These six shared weights consist of W_1 and W_3 for the input to the forward

Figure 4. BiRNN model



and backward hidden layers, W_2 and W_5 for the hidden layer-to-hidden layer connections, and W_4 and W_6 for the hidden layer-to-output layer connections (Wang et al., 2023a or b).

In addition, h_t represents the results from the forward scan, and h_t' represents the results from the backward scan. It's important to note that the forward and backward recurrent networks operate independently, with no direct interaction between their hidden layers. After both the forward and backward networks compute outputs for all time steps, the final output for each time step is calculated based on the outputs from both directions. The specific expressions are as follows:

$$h_t = f\left(w_1 x_t + w_2 h_{t-1}\right) \tag{1}$$

$$h_t^{'} = f\left(w_3 x_t + w_5 h_{t-1}^{'}\right) \tag{2}$$

$$o_t = g\left(w_4 h_t + w_6 h_t^{'}\right) \tag{3}$$

3. METHOD

3.1. System Architecture Design

To lead better processes and services, the first step is to learn a process model (Cheng et al., 2020). As shown in Figure 5, the intelligent customer service system proposed in this article mainly consists of four components: the chat dialogue (CD) subsystem, the task-oriented multi-turn dialogue (TOMTD) subsystem, the single-turn dialogue (STD) subsystem, and a fusion model. They are primarily used to address the following requirements:

(1) CD subsystem: In intelligent customer service dialogue systems, CD systems in casual conversation scenarios, while unable to address business issues, play a crucial role in offering friendly feedback. Summarizing the requirements in casual conversation scenarios, they primarily involve expanding the system's applicability, providing richer dialogue needs and enhancing response diversity through algorithmic mechanisms to align more closely with people's everyday communication styles. Because Seq2Seq inherently possesses the capability to handle variable-length input and output sequences, it is exceptionally well-suited for dynamic and diverse conversational contexts. Moreover, its flexibility in adapting to both single-turn and multi-turn dialogues allows for effective contextual understanding, enabling the model to generate coherent responses that maintain relevance to the ongoing conversation. Therefore, we developed the CD based on the Seq2Seq models, enabling them to autonomously structure language and provide diverse responses.
(2) TOMTD subsystem: During actual human customer service interactions, users' questions can sometimes be inadequately described through a simple query. In such cases, it becomes crucial for the system to gather the complete user-provided information through multi-turn interactions. TOMTD requirements are particularly essential in addressing complex user queries, such as in e-commerce shopping scenarios, where products have numerous attributes. The system must recommend products to users based on essential attributes, and users may need multiple dialogues to convey the attributes clearly. The system also needs to internally identify the user's intent, employ named entity recognition algorithms to detect entity information, and populate slots. It then checks the completeness of slot information through a dialogue management module and proceeds to execute specific tasks before returning the results to the user.
(3) STD subsystem: The STD subsystem addresses the need for addressing common user queries. These questions are typically confined to specific business domains, and users can fully express their questions in a single interaction. For example, in the e-commerce domain, questions like "When will it be shipped?" or "Is it still in stock?" fall into this category. The STD subsystem is

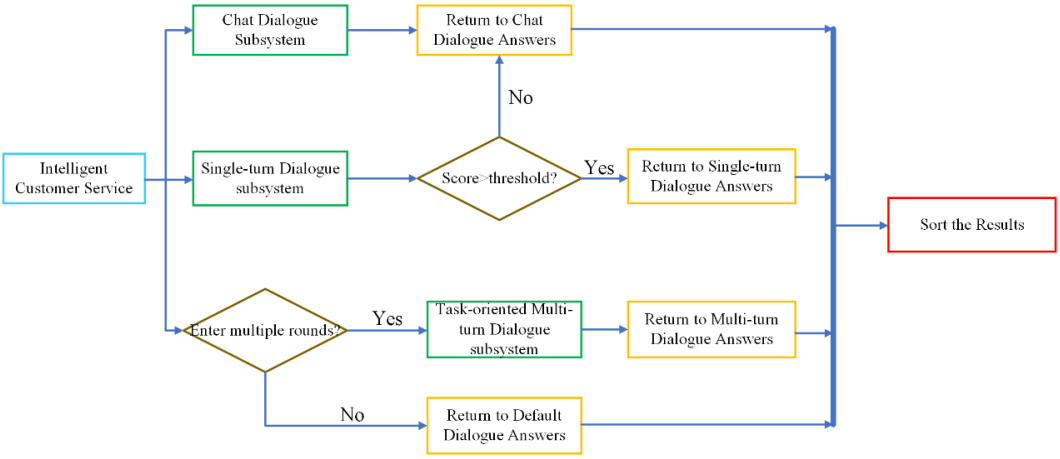
also widely used in the industry, with the common practice of storing business-related questions in a knowledge base. The most commonly used technical solution is a retrieval-based question-answering system, which matches user queries with those in the knowledge base using techniques such as inverted indexing and search engines, ultimately outputting the highest-matching answers.

(4) Fusion model: The fusion model is a method that amalgamates subsystems, employing rules and models to select optimal responses from CD, TOMTD, and STD subsystems. The TOMTD subsystem is tasked with text classification, entity extraction, and conversation management for user queries, focusing on TOMTD corpora for text classification. The STD subsystem outputs information regarding the matching degree between user questions and the knowledge base. The proposed solution involves designing a classification model based on an analysis of multi-turn and STD subsystems. This model classifies text from TOMTD and the other two dialogue subsystems, utilizing mixed corpora from TOMTD, STD, and CD subsystems for training. The classification model plays a crucial role in subsystem integration, distinguishing the TOMTD subsystem from the other two. The decision-making process involves differentiating between the STD and chat dialogue subsystems based on the matching degree of the STD subsystem. If the matching degree exceeds a predefined threshold, the system outputs the STD subsystem's answer; otherwise, it displays the CD subsystem's answer. The overall system employs a sorting process, guided by rule-based output prioritization, with an emphasis on presenting answers from the TOMTD subsystem first. If the TOMTD subsystem does not yield an output, the system checks the matching degree of the STD subsystem. If it surpasses the threshold, the answer from the STD subsystem is displayed; otherwise, the CD subsystem's answer takes precedence.

3.2. CD Subsystem Design

Intelligent customer service dialogue systems typically belong to specific vertical domains, and the responses in these domains are predefined. However, while ensuring the system's integrity, we also consider the need for user-friendly and diverse responses. Therefore, this dialogue system incorporates the CD subsystem in the system's subsystem design, especially for casual conversation scenarios. In casual conversation scenarios, the CD subsystem operates in an open domain, allowing it to respond to user inquiries on a wide range of topics, fostering friendly interactions. Deep learning generative dialogue models inherently excel in handling open-domain questions. In the design of the CD subsystem, we consider the use of Seq2Seq models, optimize model performance by comparing

Figure 5. The flowchart of the fusion model



different attention calculation methods, and employ a heuristic Beam Search algorithm to enhance response diversity.

3.2.1. Seq2Seq Model

The Seq2Seq model is an end-to-end model originally proposed by Sutskever et al. in 2014, and it is a general encoder-decoder model. Initially designed for machine translation tasks, it has later found applications in chat dialogue systems, document summarization, and other tasks. The Encoder-Decoder model is a research model in the field of text processing. The encoder takes an input sequence and encodes it into a fixed-length text vector using RNN. The decoder, in turn, decodes this text vector using RNNs to generate an output sequence. The input sequence and output sequence can have different lengths, and there is no explicit one-to-one mapping between them.

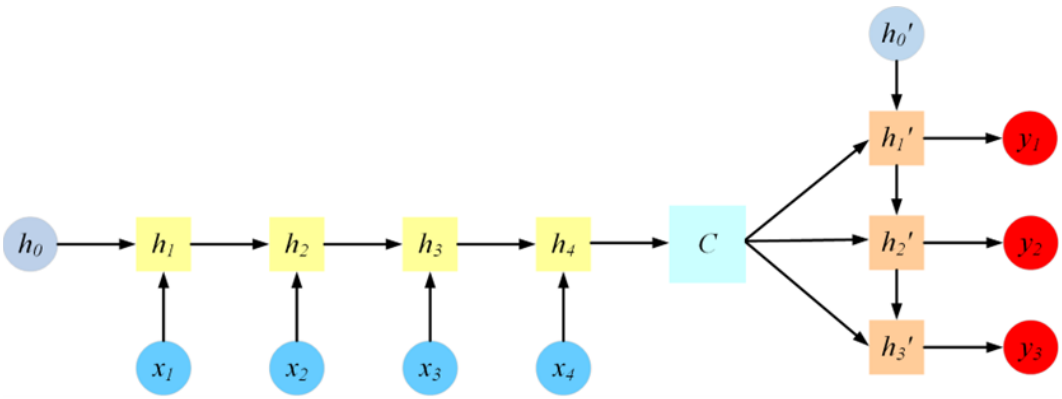
The implementation of the Encoder-Decoder framework using RNN structures is currently a mainstream approach. The entire framework operates in two phases. The first phase is the encoding phase, where the input sentence is tokenized, and then word vectors are generated using word embedding tools. These word vectors are then processed through an RNN model to obtain a probabilistic word vector. After several iterations, a vector C is formed that contains the information of the sentence. This concludes the encoding phase, and the final output is this vector C The second phase is the decoding phase, where the vector C is used in a deep learning network to reconstruct word vectors, and subsequently, the corresponding words are found using these word vectors. Through several iterations, the outputs are concatenated to generate the final answer. The detailed computational process is explained further in Figure 6.

The encoder's job is to take an input sequence $X = \left[x_1, x_2, \cdots, x_n\right]$ of variable length and transform it into a fixed-length Context Vector *C* through a recurrent network. In this context, the RNN can be either an LSTM or a GRU. h_0 represents the initial hidden state input of the encoder, which is typically initialized randomly. At an intermediate time *t*, based on the previous time step's hidden state and the current time step's input representation, the hidden state can be expressed as follows:

$$h_t = f\left(h_{t-1}, x_t\right) \tag{4}$$

where $f\left(\cdot\right)$ can be any non-linear function, such as the commonly used sigmoid function. To take into account the context information, the encoder's propagation direction can be either forward or backward.

Figure 6. The encoder-decoder model



The decoder's role is to take the fixed-length vector *c*, along with the previous output sequence $Y = \left[y_1, y_2, \cdots, y_{t-1}\right]$, and the current hidden layer state h_t and use a function transformation to predict the current output y_t at time *t*. The hidden layer state h_t and the current output y_t are given by:

$$h_t = f\left(h_t, y_{t-1}, C\right) \tag{5}$$

$$P\left(y_t | y_{t-1}, y_{t-2}, \cdots, y_1, C\right) = g\left(h_t, C\right) \tag{6}$$

where $f\left(\cdot\right)$ and $g\left(\cdot\right)$ are both nonlinear activation function.

Finally, by using the method of Maximum Likelihood Estimation (MLE), the goal is to maximize the joint probability of the output sequence. This process is used to obtain the loss of the output sequence, which can be represented as:

$$P\left(y_1, y_2, \cdots, y_t\right) = \prod_{t=1}^{n} P\left(y_t | c, y_1, y_2, y_3, \cdots, y_{t-1}\right) \tag{7}$$

$$loss = -\frac{1}{n} \log P\left(y_1, \cdots, y_t | x_1, \cdots, x_m\right) = -\frac{1}{n} \sum_{t=1}^{n} \log P\left(y_t | y_1, \cdots, y_{t-1}, c\right) \tag{8}$$

In the end, it is the output of the decoder that needs to be transformed into a probability distribution, with the softmax function commonly employed for this nonlinear transformation. Once the probability distribution for each word at a specific time step is obtained, it is through a certain algorithm that the desired output can be achieved, with the simplest approach being the selection of the word with the highest probability as the output.

Through the above process, what can be observed is that the model's core task is to calculate the probability of the next word appearing in the vocabulary based on the current input, with the word having the highest probability selected as the output from the vocabulary. It's essential to note that during training and prediction, some distinctions exist between our encoder and decoder. The encoder's primary focus is on the last hidden layer state of the recurrent neural network that is, the generated Context Vector C without the utilization of the encoder's outputs h_t at each time step. However, for the decoder, at each time step, the original target sequence serves as input, and the prediction of the current time step's output is based on the previous time step's output. Ultimately, learning takes place using the cross-entropy loss function as the loss function.

3.2.2. Attention Mechanisms

In the process of optimizing the Encoder-Decoder model, although enhancements can be achieved by employing RNN model variants like LSTM or GRU to handle longer sequences, the model framework itself exhibits a significant constraint. This constraint is characterized by the exclusive linkage between the Encoder and Decoder, facilitated through a fixed-length Context Vector *C*. In essence, all hidden layer outputs from the Encoder process are compressed into this fixed-length vector (Lai et al., 2014). This compression gives rise to two notable drawbacks:

(1) Inadequate representation: The complete information of the input sequence isn't adequately conveyed by the compressed Context Vector *C*.
(2) Information overwriting: As the sequence length grows, the initial input's information is progressively overwritten by subsequent data. This effect becomes more pronounced with longer input sequences.

These inherent limitations yield a fundamental drawback, diminishing decoding accuracy, especially during the initial stages of the Decoder, due to the inadequacy of the carried-over information.

In response to the mentioned drawbacks, the Encoder-Decoder model with an Attention mechanism offers a solution (Bahdanau et al., 2015; Luong et al., 2015). This model can learn the weights associated with each input element within the sequence and use these weights for weighted combination. As a result, the Attention mechanism serves as an intermediary between the Encoder and Decoder, furnishing the Decoder with information derived from each hidden state of the Encoder. This effectively alleviates the concern of representing the entire Encoder's information with a fixed Context Vector *C*.

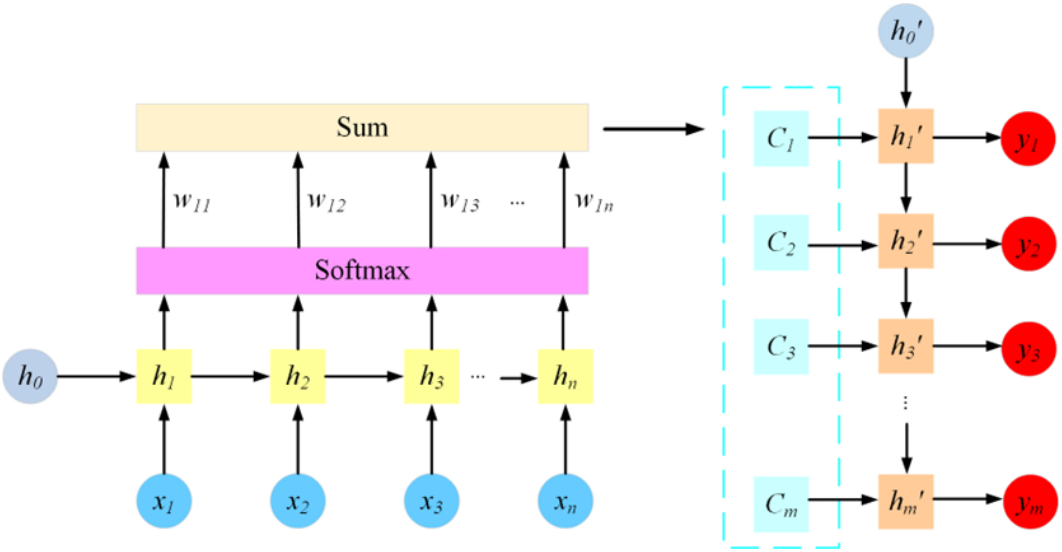
As shown in Figure 7, for the Attention mechanism, the Decoder's input is no longer the direct use of the Context Vector *C* generated by the Encoder. Instead, it is obtained by weighted summation of individual elements based on their respective importance. This can be expressed as:

$$C_i = \sum_{i=1}^{n} w_{ij} * h_j \tag{9}$$

where *i* represents the *i*-th time step, *j* represents the *j*-th element in the sequence, *n* is the length of the sequence, h_j represents the result of encoding the *i*-th input through the hidden layer, and w_{ij} denotes the weight. It signifies the importance of the hidden layer vector of the input sequence at time step *j* in the decoder at time step *i* and reflects the significance of element h_j for Context Vector *C*. These weights are normalized using the softmax function and can be expressed as follows:

$$w_{ij} = \frac{\exp\left(a_{ij}\right)}{\sum_k exp\left(a_{ij}\right)} \tag{10}$$

Figure 7. Seq2Seq attention model



where a_{ij} reflects the degree of match between the element to be encoded and other elements. When the match is higher, the value of w_{ij} is also larger, indicating a greater influence on that particular element.

3.2.3. Beam Search Algorithm

The process of generating dialog in a chatbot is essentially the decoding process of the Seq2Seq model. Typically, during decoding, we aim to choose the sentence with the highest probability as the output, which means selecting the word with the highest probability in each prediction in the Seq2Seq model. This follows a greedy search approach. In addition to the greedy algorithm, there is also a beam search algorithm. Beam Search is a strategy set during the testing phase to improve accuracy. It restricts the number of words with the highest likelihood retained in the candidate set by setting the Beam Size parameter, thus increasing the overall output probability of the sentence. At time *t*, the Beam Size sequence of optimal solutions at the previous moment and the corresponding likelihood probability are:

$$\left[s_{t-1}^{(1)}, s_{t-2}^{(2)}, \cdots, s_{t-1}^{(BeamSize)}\right], \left[p_{t-1}^{(1)}, p_{t-2}^{(2)}, \cdots, p_{t-1}^{(BeamSize)}\right]$$

Besides, s_{t-1}^{i} and p_{t-1}^{i} are expressed as:

$$s_{t-1}^{(i)} = \left[y_1^{(i)}, y_2^{(i)}, \cdots, y_{t-1}^{(i)}\right] \tag{11}$$

$$p_{t-1}^{(i)} = p\left(y_{t-1}^{(i)}, y_{t-2}^{(i)}, \cdots, y_1^{(i)} | x\right) = \prod_{j=1}^{t-1} p\left(y_j^{(i)} | x, y_1^{(i)}, y_2^{(i)}, \cdots, y_{j-1}^{(i)}\right) \tag{12}$$

where $y_j^{(i)}$ denotes the output the j-th words in $s_{t-1}^{(i)}$, and $p\left(y_{t-2}^{(i)}, y_{t-2}^{(i)}, \cdots, y_1^{(i)} | x\right)$ denotes the probability of $y_{t-1}^{(i)}, y_{t-2}^{(i)}, \cdots, y_1^{(i)}$ given *x*.

For the previous time step, we can compute all possible $s_t^{(i)}$ values for $s_{t+1}^{(i)}$. To prevent numerical underflow, when calculating $p_t^{(i)}$, it's common to take the logarithm. Then, we retain the large Beam Size probabilities, which are the most significant $s_t^{(i)}$ values, as the optimal solution for time step *t*.

Finally, for all ending time steps t_n, the ultimate evaluation metric is determined based on the ratio of their logarithmic likelihood probability values to the sequence length. It can be given by:

$$score\left(s_{t_n}^{(i)}\right) = \frac{1}{t_n^a} \log\left(p_{t_n}^{(i)}\right) \tag{13}$$

where *a* denotes the smoothing coefficient, which serves to strike a balance between fully normalizing results with respect to sequence length and not normalizing based on length at all. Ultimately, the sequence with the highest score is chosen as the final output.

3.3. TOMTD Subsystem Design

TOMTD refers to the process in which a conversational system, after identifying the user's intent, engages in multiple rounds of dialogue to gather the necessary information to fulfill the user's specific requirements. Its primary objective is to collect the requisite data based on the user's intent and assist

the user in completing a task or operation. In contrast to single-turn dialogue systems, task-oriented multi-turn dialogue systems serve as a supplement for users who cannot express their needs entirely in a single turn. These systems typically rely on contextual information and manage the dialogue state. The inherent variability and diversity in how people use language make understanding user needs a challenging task. For instance, users may have a clear intent to book a flight or make a restaurant reservation, but due to the complexity of their requirements, they may need to ask multiple questions or change their intent during the conversation. Consequently, TOMTD is of paramount importance for conversational systems.

3.3.1. NLU Module

NLU, as a crucial component of NLP, plays a significant role in processing user input sentences or speech recognition results. Its primary function is to process and extract user dialogue intent and conveyed information. It mainly consists of three components: data cleansing, intent recognition, and named entity recognition.

User queries undergo data cleansing as a first step, involving the removal of special characters and stopwords. Intent recognition, on the other hand, aims to identify the user's intent. In practice, intent recognition is essentially a classification problem. Models analyze and classify sentences to determine the most likely user task intent. Named entity recognition, often referred to as slot filling, is essentially a sequence labeling task. It involves parsing and identifying entity information within the user's query, providing crucial details for executing specific tasks later on.

(1) Intent recognition

Intent recognition is fundamentally a classification task. In the early stages, intent recognition was designed based on rule-based or template matching methods. The drawback of such methods is the requirement for significant manual effort and their relatively low accuracy. However, they offer the advantage of high accuracy and controllability as long as a template is matched. Later on, statistical methods were introduced to address intent recognition, defining it as a classification problem and applying traditional machine learning methods (Haffner et al., 2003; Kim et al., 2016; Liu & Lane, 2016).

Intent recognition based on template matching is implemented using AIML. Its integration into a unified intent determination model primarily leverages its extensibility and high operability. In cases where the model cannot perform recognition effectively in specific situations, template matching serves as a complementary method to achieve flexible recognition.

The workflow of deep learning-based intent recognition models involves data preprocessing, including tasks such as removing stop words and special symbols. Subsequently, text segmentation and word vector generation take place, which can be accomplished using tools like Jieba, Word2vec, or BERT. The objective is to transform textual data into numerical representations suitable for computation. Model design follows, with two common types of networks employed for intent recognition tasks: CNN and RNN. Each of these networks has its own advantages and drawbacks. CNN focuses on local text features but lacks global feature attention, potentially leading to the issue of multiple classifications for a single document. On the other hand, LSTM employs a sequential processing approach and excels in short-text classification.

Based on above analysis, the proposed scheme adopts a unified intent determination model that combines deep learning-based intent determination as the primary approach with template-based determination as a secondary method. When the deep learning model exhibits poor performance in classifying certain sentences, template matching serves as a complementary solution to enhance model accuracy and operability.

(2) Named Entity Recognition

Named Entity Recognition (NER) refers to the identification of specific entities in text, such as names of individuals, locations, organizations, proper nouns, etc., and is typically considered a part of the tokenization process (Bikel et al., 1999). The current dominant model in NER is BERT+LSTM+CRF, which comprises three components. The function of BERT model is similar to Word2Vec. The combination of LSTM+CRF primarily aims to introduce deep learning into traditional machine learning methods to enhance the accuracy and recall of NER models.

The reason LSTM+CRF has become the primary algorithmic structure for NER is due to their complementary advantages. LSTM models are capable of capturing long-range contextual information and possess the ability to fit non-linear patterns through neural networks. However, they do not consider the dependencies between various output states, which are essential in sequence labeling tasks. On the other hand, CRF models are designed with interdependencies among output states in mind, placing more emphasis on the overall probability distribution of the sentence. Nevertheless, they may not account for long-range contextual information. The BiLSTM+CRF model structure makes full use of the strengths of both models by paying attention to contextual information and ensuring that the output sequence adheres to basic dependency constraints. The model structure of BiLSTM+CRF is depicted in Figure 8.

Firstly, in the preprocessing stage, special characters in the text need to be removed, and the text format should be standardized. Then, the text is tokenized into Chinese words. Common tokenization methods can make use of open-source tools such as Jieba, SnowNLP, PkuSeg, and others. Finally, sequence labeling is applied, with common methods like BIO encoding. In this encoding, 'B' represents the beginning of an entity, 'I' represents the middle of an entity, and 'O' signifies non-entity parts.

Next, the words are vectorized, meaning they are represented in vector form. The annotated sentences are transformed into a sequence $X = \left(x_1, x_2, \ldots, x_n\right)$, where x represents the vectorized form of words in One-Hot encoding with dimensions equal to the size of the dictionary. Then, after passing through the Word Embedding layer, a BERT pre-trained model can be utilized. This converts the One-Hot vectors into corresponding Word Embedding word vectors, denoted as $W = \left(w_1, w_2, \ldots, w_n\right)$.

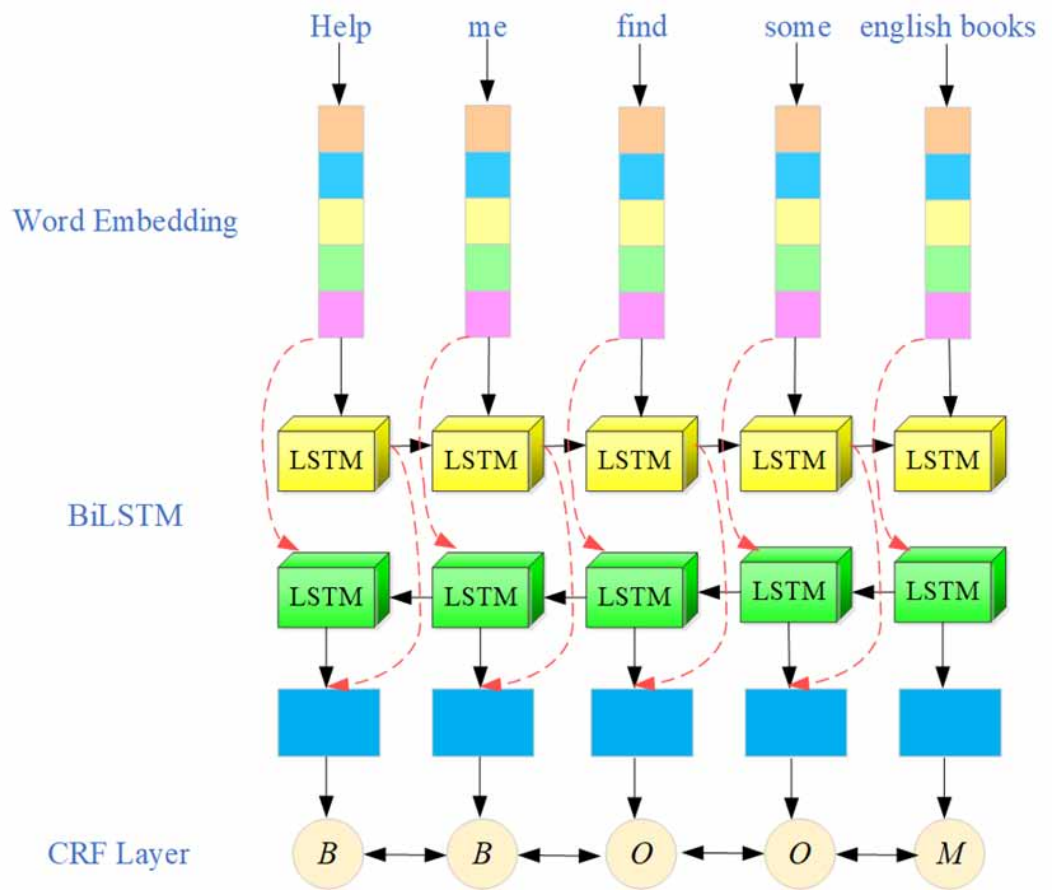
Besides, after obtaining the word vectors *W* through Word Embedding, the process proceeds into the Bidirectional Long Short-Term Memory (BiLSTM) neural network. This yields the forward LSTM's Hidden Layer sequence $\vec{h} = \left[\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots, \overrightarrow{h_n}\right]$ and the backward LSTM's Hidden Layer sequence $\overleftarrow{h} = \left[\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots, \overleftarrow{h_n}\right]$. These sequences are then concatenated based on their positions to form $h = \left(\overrightarrow{h_t}, \overleftarrow{h_t}\right)$. Assuming the Hidden Layer is an m-dimensional vector, we have $h \in R^{n \times m}$. In addition, we include dropout mechanisms in this process to prevent overfitting.

Subsequently, in order to map the m-dimensional vectors to a k-dimensional set of labels, resulting in automatically extracted sentence features $P = \left(p, p_1, \ldots, p_n\right), P \in R^{n \times k}$, where each dimension can be considered as a score for each word with respect to different labels, a linear transformation is applied to the vectors.

Finally, the parameters of the CRF layer consist of a (k+2)×(k+2) matrix *O*, where the additional dimensions are k+2 because the calculation needs to encompass the entire sentence. Here, $O_{i,j}$ denotes the transition probabilities from the *i*-th label position to the *j*-th label position, enabling the prediction of the current label position based on previous label positions.

Assuming the obtained label sequence is $y = \left(y_1, y_2, \cdots, y_n\right)$, the scores for the corresponding label sequences can be divided into two parts. The first part can be viewed as the scores of the labels corresponding to the automatically extracted features of the sentence, determined by BiLSTM. The second part can be seen as the scores for transitioning from one label to another, determined by CRF. Adding these two parts together results in the final score as follows:

Figure 8. BERT + BiLSTM + CRF NER method



$$score\left(x,y\right)=\sum_{i=1}^{n}P_{i,y_i}+\sum_{i=1}^{n+1}O_{y_{i-1},y_i} \tag{14}$$

Finally, normalization is performed through softmax to obtain the probability formula as shown in (12), which is then transformed into a log-likelihood function. The training process involves maximizing the likelihood function, as shown in (15):

$$P\left(y|x\right)=\frac{exp\left(score\left(x,y\right)\right)}{\sum_{y}exp\left(score\left(x,Y\right)\right)} \tag{15}$$

$$\log P\left(y^x|x\right)=score\left(x,y^x\right)-\log\left(\sum exp\left(score\left(x,Y\right)\right)\right) \tag{16}$$

During model prediction, dynamic programming is used to solve the problem, and the Viterbi decoding algorithm is employed. The formula is given by:

$$y^*=argmax_y score\left(x,Y\right) \tag{17}$$

3.3.2. TOMTD Management Module

The TOMTD management module is the core of the entire task-oriented multi-turn dialogue system. It is responsible for controlling the dialogue process. This module relies on the semantic representations provided by the NLU module to maintain and track the dialogue state and determine the next steps based on specific strategies. The TOMTD management module consists of three sub-modules:

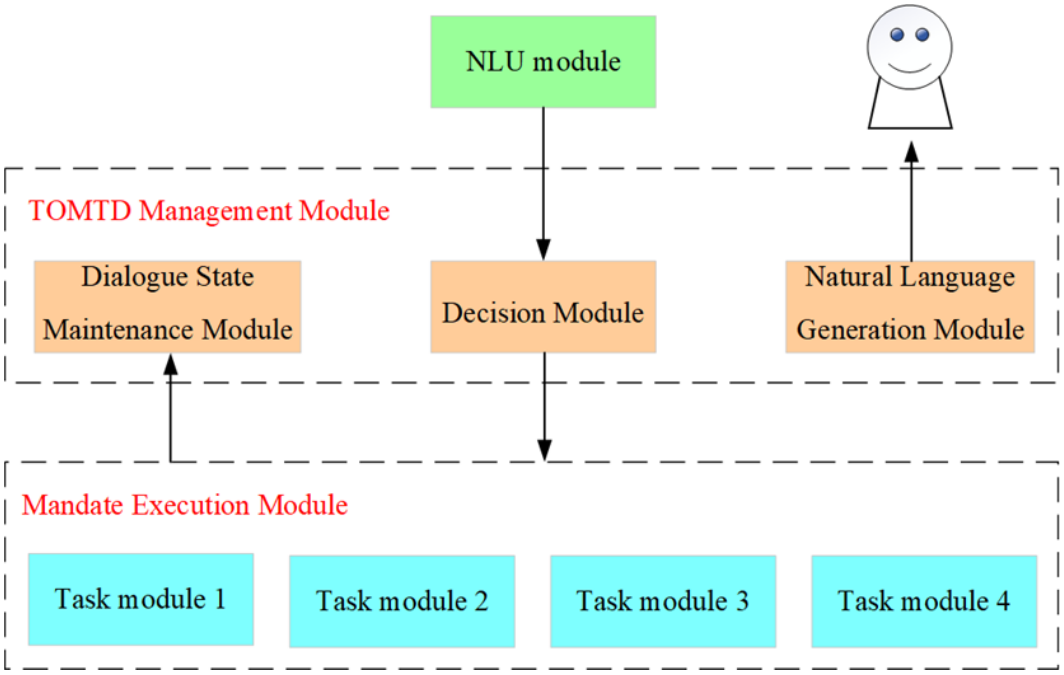
(1) Dialogue State Maintenance Module: This module is responsible for maintaining the dialogue state by organizing the entity and intent information extracted by the NLU.
(2) Decision Module: It decides the next system actions based on the current dialogue state and system behavior.
(3) Natural Language Generation Module: This module generates natural language responses based on the output from the decision module to respond to the user.

As shown in Figure 9, the TOMTD management module receives the intent and slot information extracted by the NLU module and stores this information in a database. It checks whether all the necessary slot information required to execute the user's intent has been filled. If there are unfilled slots, the decision module formulates the next actions, and the natural language generation module generates response language. When all slots are filled, indicating that the task can be executed, the decision module sends the specific task to the task execution module and then generates a natural language response to the user, completing the interaction of the entire subsystem.

3.3.3. Task Execution Module

The main function of the Task Execution Module is to perform specific backend tasks based on the task execution commands generated by the Task-Oriented Multi-Turn Dialogue Management Module. The inputs to the Task Execution Module include the task type, user intent, and the required slot

Figure 9. TOMTD management module flowchart



information. After executing the designated task, it returns the task execution results to the Task-Oriented Multi-Turn Dialogue Management Module, extracts key information from the results, and organizes a natural language response to the user.

The task execution phase can be categorized into two types. First, there are offline operations, which do not require internet connectivity and involve calling internal system functions, such as setting alarms, reminders, or querying the current time. The second type consists of online operations, which require obtaining relevant information from external resources via the internet. Typical online tasks involve making requests to external APIs, such as querying the weather for the day. In such cases, the Task Execution Module, based on the intent and slot information provided by the TOMTD Management Module (e.g., address and date parameters), constructs a standardized URL for requesting data from external APIs. Subsequently, the module receives and processes the requested results and returns them to the dialogue management module.

3.4. STD Subsystem Design

STD subsystem differ from TOMTD systems as they emphasize a one-question-one-answer format, providing precise answers based on the user's questions. They are primarily used in scenarios where common questions need to be answered, and their workflow closely resembles an information retrieval process. The fundamental distinction between the two lies in whether the system needs to maintain user goal states and whether it requires a decision-making process to complete tasks. STD subsystem typically acquire answers in two ways. Firstly, they can extract answers from unstructured natural language document collections, such as using web search engines to retrieve answers from search results. This approach is suitable for questions that require real-time answers, such as news-related queries. The second approach involves retrieving answers from structured knowledge bases, which is ideal for factual questions, especially within specific vertical domains. Given that customer service inquiries typically involve business-related issues (Liu et al., 2021), this paper adopts a knowledge-base-based design for STD subsystem.

When a user's question enters the STD subsystem, the first step is to perform preprocessing, which involves removing special symbols and stop words from the query. These words are meaningless for semantic understanding and their removal enhances subsequent retrieval efficiency. Next, a Restful API is used to access Elasticsearch. Elasticsearch's role is to utilize word frequency algorithms such as TF-IDF and BM25 to select question-answer pairs with the highest similarity. It then retrieves the corresponding answers from the knowledge base based on the IDs of these pairs. Finally, the results are concatenated and returned to the user as the answer.

3.5. Database Design

3.5.1. Knowledge Base Design

The knowledge base of this system is primarily used to store question-answer pairs for the single-turn dialogue subsystem. In subsequent system updates and iterations, a dedicated knowledge base management platform can be designed to add, delete, or modify the knowledge content. The advantage of this approach is that it allows real-time changes to the feedback content of online services through a user-friendly management tool, aligning with the rapid development needs of the business.

3.5.2. Dialogue Table Design

During the construction of the intelligent customer service dialogue system, it is necessary for the system to record the conversation content between the user and the system. This practice offers several advantages. Firstly, it allows for real-time monitoring of user dialogue content, especially in critical situations. For instance, in the context of ride-sharing customer service, in the event of emergencies, historical conversation records can be reviewed to understand the dialogue content of passengers or drivers. This can provide valuable and critical information to ensure the safety of passengers and

drivers, playing a pivotal role in such scenarios. Secondly, recording conversation content can be used for system optimization. By comparing the dialogue content of users and the system, potential issues can be identified, enabling targeted system improvements to enhance the system's ability to intelligently resolve issues. Finally, when a user opts to switch to human customer service, the conversation data between the user and the human agent can be saved, providing authentic and effective conversation data for future model training.

3.5.3. Storage of Training Data

In order to facilitate rapid model training and the addition of new incremental data to the training dataset, it is necessary to save the training dataset. In the subsequent model training process, real human customer service dialogue data can be added to the existing training dataset in real-time. Then, through a script program, we can periodically train the model and record parameters such as model accuracy. If the accuracy of the newly trained model is higher than that of the previous model, the system can automatically replace the old model. The frequency of running this script program can be adjusted periodically based on the size of the new data to ensure that models with higher accuracy are deployed promptly.

4. EXPERIMENT ANALYSIS

4.1. Experimental Environment

The experimental environment for this chapter utilizes the Python programming language and employs the open-source deep learning framework PyTorch to build the Seq2Seq model. Training deep learning models involves extensive floating-point calculations, so GPU acceleration is commonly employed to speed up the training process. In this experiment, the Juchip cloud platform was chosen, as it offers a rich supply of GPU resources that can be selected as per the model's requirements. The machine used in this experiment is equipped with an RTX2080Ti GPU, an i7-8700K CPU, 16GB of RAM, and the software environment includes Python 3.6, PyTorch 1.0.1, CUDA 9, and cuDNN 7.

4.2. Dataset

After the screening of multiple Chinese conversation corpora, it was ultimately decided to use the Qingyun dataset for model training in this experiment. The Qingyun dataset is derived from a chatbot communication group, comprising dialogues in common casual conversation scenarios that span various life domains. It consists of 105,914 text dialogues, presented in a question-and-answer format with tab key separators.

The quality of a dataset forms the foundation for training a good model, and the dataset's quality directly impacts the effectiveness of model training. Therefore, filtering and refinement were conducted on the Qingyun dataset, primarily involving the removal of lengthy texts and sensitive conversations. As a result of this screening, the dataset was reduced to 90,356 dialogues.

4.3. Experiment Results of CD Subsystem

4.3.1. The Impact of Recurrent Unit Types on Model Performance

In order to investigate the impact of different recurrent neural network structures (RNN, LSTM, and GRU) on the model's performance, the comparative experiments in this study employed a test-to-training data ratio of 1:9. The Adam optimizer was selected, and both the recurrent neural network encoder and decoder were configured with five layers. Additionally, the size of embedding and hidden layer are both chosen as 256. The dropout rate is 0.1. In addition, the parameters of Adam optimizer are set as $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.99$.

The evaluation of the experimental model is primarily based on the BLEU (Reiter, 2018) algorithm and the Similarity algorithm as the model's assessment criteria. BLEU is a metric used for evaluating the quality of machine-generated text, especially in the context of machine translation. It was designed to measure the similarity between a candidate machine-generated translation and one or more reference translations. The batch size is set to 1024, and the number of epochs is set to 500 rounds. The experimental results of the model are shown in Figure 10. In addition, the Similarity algorithm is expressed as:

$$similarity\left(s_i, s_j\right) = \frac{v_{s_i} v_{s_j}}{v_{s_i} \quad v_{s_j}} \tag{18}$$

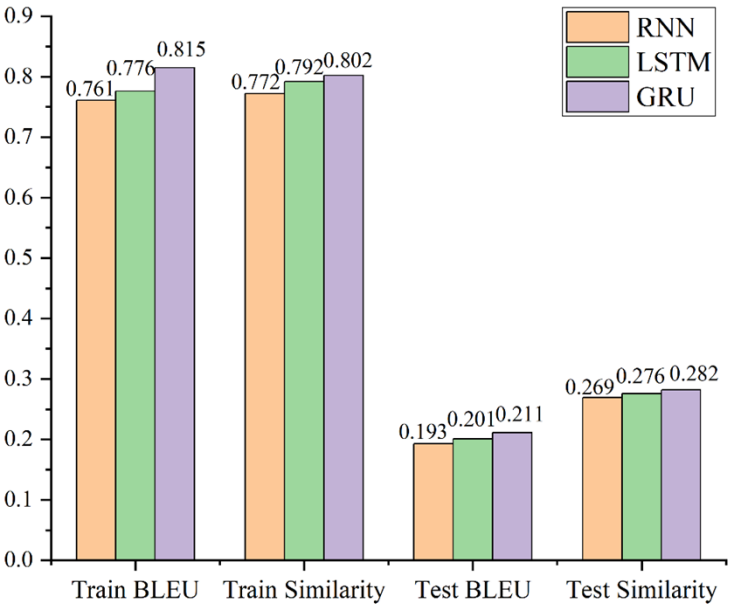
where v_{s_i} and v_{s_j} denote the sentence vector of s_i and s_j, respectively.

Drawing insights from the experimental outcomes, the GRU recurrent unit emerges as the most effective performer. Simultaneously, a notable disparity in scores between the test set and the training set becomes apparent. The diminished BLEU and Similarity scores on the test set, in comparison to the training set, can be primarily ascribed to the inherent randomness and diversity in the dialogue dataset. It's crucial to acknowledge that while BLEU and Similarity metrics offer valuable insights, they only provide a partial reflection of the model's performance. Therefore, a subjective evaluation by human assessors remains imperative for a comprehensive understanding of the model's capabilities.

4.3.2. The Impact of Recurrent Unit Layers on Model Performance

The Seg2Seq model's structure consists of an encoder and a decoder, with their association facilitated through the context vector *C*. In the model's architecture, we chose GRU as the recurrent neural unit and investigated the impact of the number of recurrent unit layers on model performance. For this experiment, we assumed that the encoder and decoder have the same number of layers. The number

Figure 10. Experimental results of different recurrent unit



of recurrent neural network layers was set to three, four, and five as variables. The test-to-training data ratio was set at 1:9, and the optimizer selected was the Adam algorithm. The batch size was set to 1024, and the number of epochs was set to 100 rounds. The model's evaluation was based on the BLEU and Similarity algorithms. The experimental results are illustrated in Figure 11.

The experimental results clearly indicate that with the increase in the number of layers in the GRU network, all evaluation metrics demonstrate an improvement in accuracy. This suggests that augmenting the number of layers in the recurrent neural network has the potential to enhance the model's performance to a certain extent.

4.3.3. The Impact of Different Attention Calculation Methods on Model Performance

In the preceding section, we observed significant enhancements in the relationship between semantics and vocabulary through the analysis of attention mechanisms. In this section, we aim to experimentally validate the impact of various attention mechanisms on model performance. Throughout the experimental process, attention mechanisms will be incorporated into the encoder-decoder model. Comparative experiments will entail the introduction of different attention algorithms into the model, investigating their influence on overall performance. For this experiment, GRU will persist as the recurrent neural unit, maintaining a depth of five layers for both the recurrent unit encoder and decoder. The Adam optimizer will be employed to facilitate model convergence. Four distinct attention calculation methods, including Bahdanau's attention, Luong's dot, general, and concat, will be tested on the training set. The batch size for training data will be set at 1024, and the number of epochs will be fixed at 100 rounds. The results are visually presented in Figure 12.

Various attention methods exhibit a common foundational structure and algorithmic principles but diverge in their computational approaches. In the context of this dataset, the Bahdanau Attention algorithm has demonstrated superior performance within the model. The attention mechanism deviates from the previous pattern of connecting the encoder and decoder through a fixed-length context vector, thereby augmenting the alignment between words in the encoder and decoder. This alteration results in more seamless and nuanced responses.

Figure 11. Results of deep test of GRU

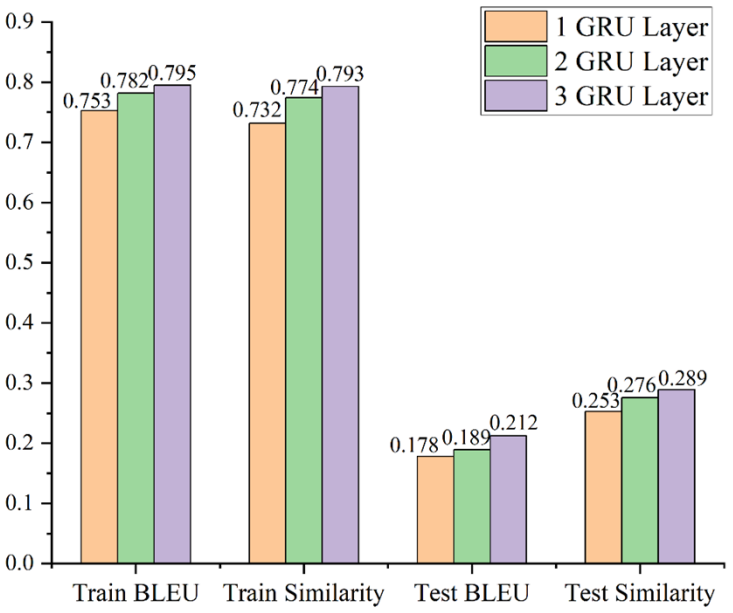
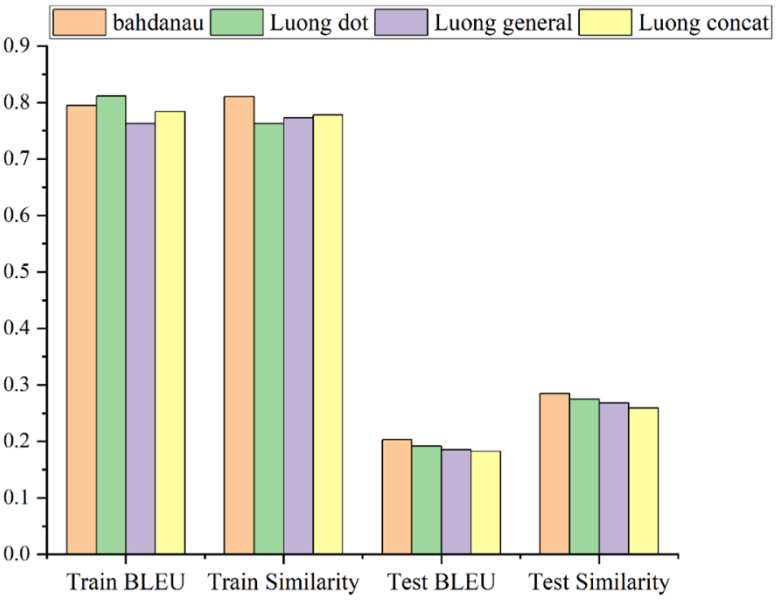


Figure 12. Comparison of experimental results with different attention methods



4.4. Experiment Results of TOMTD Subsystem

4.4.1. Intent Recognition Experiments

Intent recognition stands as a pivotal element within this dialogue system. In this experiment, we devised intent recognition modules employing neural networks with varying model structures, encompassing BiLSTM, CNN, and CNN+BiLSTM. Serving as a classification model, the intent recognition model extracts features through the network model and transforms these features into probabilities using a softmax layer.

The training process kicks off with data preprocessing, involving operations such as punctuation removal, stop-word elimination, and tokenization. Following this, word vectors are generated using word embedding tools or pre-trained BERT models. With word vectors in place, the network model comes into play for feature extraction, subsequently undergoing normalization through Softmax to yield classification categories.

The predictive model construction comprises two components. The first part involves the neural network-based intent recognition model, as depicted in the diagram. Acknowledging that the accuracy of deep learning models may not be flawless and they might struggle with accurately recognizing certain commonly used queries, we introduce a rule-based AIML template matching method to complement the deep learning model. This method precisely recognizes and outputs the correct intent for critical queries that may pose challenges for the model. Through the integration of multiple modules, our goal is to bolster intent recognition accuracy.

Given the distinctive characteristics of CNN and RNN networks, it is imperative to conduct comparative experiments on various network structures. In this study, we conducted testing using manually annotated data collected from the web and supplemented with self-authored data. The dataset encompasses a total of 1,763 instances distributed across four categories: bill inquiries, weather queries, product recommendations, and case inquiries.

Three distinct model structures, namely Bi-LSTM, CNN, and CNN+LSTM, were chosen. For model evaluation, 20% of the data was randomly selected to form the test set. Evaluation metrics

encompassed accuracy, recall, and F1 score. The parameters used for training were batch size=64 and epoch=10. The specific performance results of each model are illustrated in Figure 13.

An examination of the test parameters reveals that, when contrasted with both BiLSTM and CNN models, the CNN+LSTM model structure demonstrates a marginally superior performance on the test set, achieving an impressive accuracy rate of 99%. Following closely is the CNN model in terms of performance.

4.4.2. NER Recognition Experiments

The primary task of the named entity recognition module is to identify entity information within user queries, providing this information to the dialogue management module for slot filling. Named entity recognition is a natural language sequence labeling problem, involving two main tasks: determining the position of entities within the sentence and classifying these entities. This experiment focuses on exploring the impact of different model structures on model accuracy through comparative analysis.

Currently, one of the widely used models for named entity recognition is RNN+CRF. This paper adopts this model structure, which can be enhanced by making minor improvements, such as incorporating the BERT pre-trained model into the word embedding process or substituting RNN with LSTM or GRU recurrent neural networks. In the following sections, different model structures will be compared through comparative experiments.

The model structures included in the comparative experiments are BiLSTM+CRF, BiGRU+CRF, BERT+BiLSTM+CRF, and BERT+BiGRU+CRF. Common parameters for the model comparison experiments include batch size=64, epoch=10, and a test-to-training data ratio of 8:2. The experimental results are presented in Figure 14.

Based on the aforementioned comparative experiments, it can be concluded that, on this dataset, the BiGRU+CRF model structure exhibits higher accuracy compared to the BiLSTM+CRF model structure. The accuracy of models incorporating BERT is higher than that of models without BERT. It's worth mentioning that models with BERT converge faster during training than those without BERT. After the first epoch of training, the model with BERT achieved a 0.23 increase in accuracy

Figure 13. Intention recognition experiment results

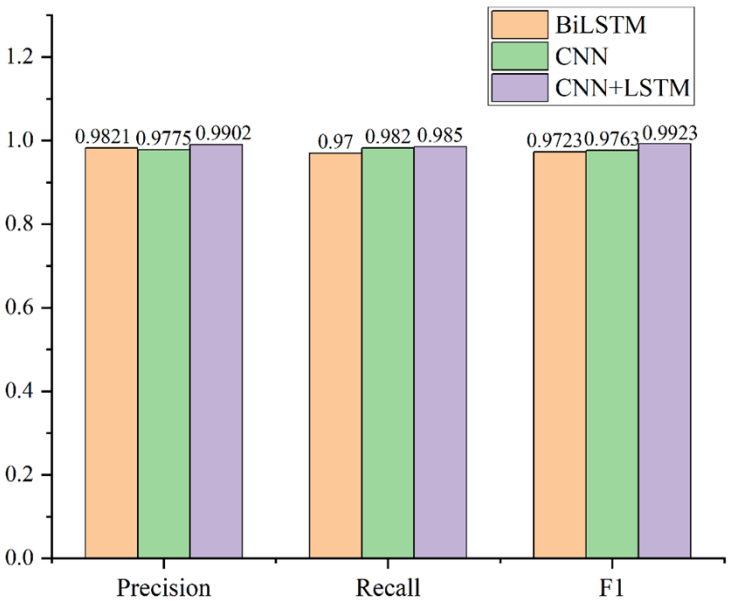
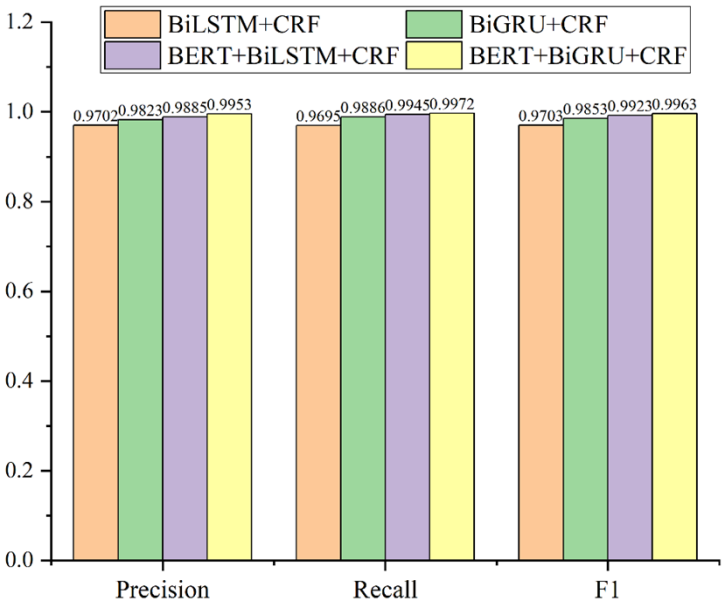


Figure 14. NER recognition experiment results



compared to the model without BERT. This underscores the improved generalization performance when integrating BERT pre-trained models. Therefore, it is recommended to use BERT pre-trained models in small datasets to enhance model generalization performance.

4.5. Performance Tests

The response time of the system reflects its performance. In scenarios with high concurrent requests, the response speed of interfaces determines the system's ability to provide messages to users. Performance testing primarily focuses on testing the response time of interfaces and the pressure they can handle. In the testing process, 100 random questions were selected as user input, covering all three subsystems of the system to ensure comprehensive testing. The Jeter testing tool was employed for the testing, and the testing environment used was the Mac OS operating system with 8GB of memory and a 5-7360U processor. The performance test cases and results are presented in Table 1.

From the average response times of various modules in the system, it can be observed that the component impacting the system's response time is primarily the multi-turn dialogue sub-system. This aligns with expectations, as the multi-turn dialogue sub-system involves more complex models and

Table 1. The examples of system performance tests

Number	Name	Explanation	Average response time
1	Get Answer	The process from the user sending a message to receiving a message	743 ms
2	Chatbot Answer	The process from the CD subsystem receiving a message to producing an output message	135 ms
3	Taskbot Answer	The process from the TOMTD subsystem receiving a message to producing an output message	534 ms
4	Execute Task	Time of execution of the tasks	235 ms
5	Esbot Answer	The STD subsystem receives messages and outputs them	253 ms

functional components compared to the other two sub-systems. Response time is not only dependent on the algorithm but is also influenced by the computational power of the experimental equipment. Although these response times may not match those of most current customer service systems, it's noteworthy that all response times remain within the 1 s. Hence, these response times are within an acceptable range, and it can be concluded that the system's performance meets the expected standards.

4.6. Satisfaction Tests

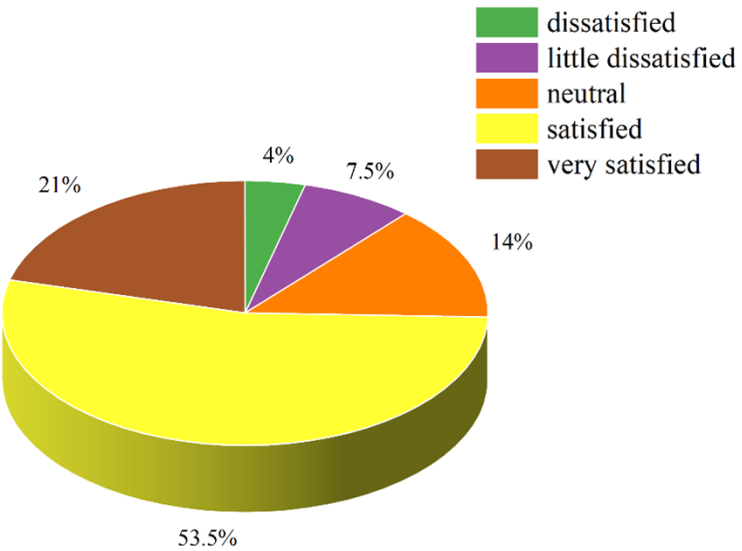
Furthermore, to validate whether the intelligent customer service proposed in this paper can achieve a satisfactory level, we randomly selected 200 testersto engage in conversations with the intelligent customer service. The testers were randomly selected from four grades in the school in a 2:2:3:3 ratio. In addition, the ratio of male to female students was 1:1. Testers were free to initiate conversations with any content they chose. Following the conclusion of the dialogues, testers were asked to evaluate their satisfaction with the conversation. Satisfaction levels were categorized into five levels: "dissatisfied," "little dissatisfied," "neutral," "satisfied," and "very satisfied." The test results are illustrated in Figure 15.

The results in Figure 15 show that the percentage of people who are satisfied overall is 74.5% and the percentage of people who are dissatisfied is 11.5%. Therefore, the intelligent customer service proposed in this paper meets the requirements in terms of overall performance.

5. CONCLUSION

In response to the evolving requirements of intelligent customer service dialogue systems, this paper proposes an intelligent customer service dialogue system consisting of three subsystems: CD, STD, and TOMTD. For the CD subsystem, we explore three dimensions to enhance Seq2Seq model performance, namely RNN units, attention calculation methods, and the depth of RNNs. In the TOMTD subsystem, the paper introduces intent determination and named entity recognition modules, elevating overall model accuracy through a combination of rule-based and model-based approaches. Additionally, a STD system based on the Elasticsearch database is devised. Through comparative experiments aimed at optimizing model performance, a fusion mechanism that integrates the three

Figure 15. Satisfaction test results



subsystems using both rule-based and model-based approaches is employed, resulting in a fully functional, straightforward, and efficient intelligent customer service dialogue system. However, it's worth noting that the method in this paper has some limitations, such as the incompleteness of the dataset and the model's limitations in handling more intricate conversations. Future research directions will focus on further enhancing the natural language recognition and processing capabilities of the intelligent customer service dialogue system.

COMPETING INTERESTS

The authors of this publication declare there are no competing interests.

FUNDING

This research was supported by the Natural Science Foundation of Hubei Province (2022CFB663) and the Hubei Province Excellent Youth Social Science Talent Training Plan, and the Wuhan University of Business Doctoral Fund projects (2021KB002).

REFERENCES

Bai, S., Yin, Y., Wu, Y., Zhang, J. Z., Yu, Y., & Jasimuddin, S. M. (2022). Consumer engagement on social networking sites: The antecedents and mediating mechanism. [JOEUC]. *Journal of Organizational and End User Computing*, *34*(1), 1–19. doi:10.4018/JOEUC.307567

Bikel, D. M., Schwartz, R., & Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning*, *34*(1), 211–231. doi:10.1023/A:1007558221122

Cai, T., Giannopoulos, A. A., Yu, S., Kelil, T., Ripley, B., Kumamaru, K. K., Rybicki, F. J., & Mitsouras, D. (2016). Natural language processing technologies in radiology research and clinical applications. *Radiographics*, *36*(1), 176–191. doi:10.1148/rg.2016150080 PMID:26761536

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explorations*, *19*(2), 25–35. doi:10.1145/3166054.3166058

Cheng, L., van Dongen, B. F., & van der Aalst, W. M. (2019). Scalable discovery of hybrid process models in a cloud computing environment. *IEEE Transactions on Services Computing*, *13*(2), 368–380. doi:10.1109/TSC.2019.2906203

Fountaine, T., McCarthy, B., & Saleh, T. (2019). Building the AI-powered organization. *Harvard Business Review*, *97*(4), 62–73.

Haffner, P., Tur, G., & Wright, J. H. (2003, April). Optimizing SVMs for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003* (Vol. 1, pp. I-I). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735 PMID:9377276

Ji, B., & Zhang, Y. (2023). Few-shot relation extraction model based on attention mechanism induction network. *Journal of Jilin University Science Edition, 61*, 845-852.

Kim, J. K., Tur, G., Celikyilmaz, A., Cao, B., & Wang, Y. Y. (2016, December). Intent detection using semantically enriched word embeddings. In 2016 IEEE Spoken Language Technology Workshop (SLT) (pp. 414-419). IEEE. doi:10.1109/SLT.2016.7846297

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). doi:10.1609/aaai.v29i1.9513

Liu, C., Zeng, Q., Cheng, L., Duan, H., & Cheng, J. (2021). Measuring similarity for data-aware business processes. *IEEE Transactions on Automation Science and Engineering*, *19*(2), 1070–1082. doi:10.1109/TASE.2021.3049772

Nie, J., Wang, Q., & Xiong, J. (2021). Research on intelligent service of customer service system. *Cognitive Computation and Systems*, *3*(3), 197–205. doi:10.1049/ccs2.12012

Reiter, E. (2018). A structured review of the validity of BLEU. *Computational Linguistics*, *44*(3), 393–401. doi:10.1162/coli_a_00322

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. doi:10.1109/78.650093

Serban, I., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. (2016, March). Building end-to-end dialogue systems using generative hierarchical neural network models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1). doi:10.1609/aaai.v30i1.9883

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.

Tian, S., Li, L., Li, W., Ran, H., Ning, X., & Tiwari, P. (2024). A survey on few-shot class-incremental learning. *Neural Networks*, *169*, 307–324. doi:10.1016/j.neunet.2023.10.039 PMID:37922714

Wang, C., Ning, X., Li, W., Bai, X., & Gao, X. (2023a). 3D person re-identification based on global semantic guidance and local feature aggregation. *IEEE Transactions on Circuits and Systems for Video Technology*, 1. doi:10.1109/TCSVT.2023.3328712

Wang, L., Huang, N., Hong, Y., Liu, L., Guo, X., & Chen, G. (2023b). Voice-based AI in call center customer service: A natural field experiment. *Production and Operations Management*, *32*(4), 1002–1018. doi:10.1111/poms.13953

Xiao, L., & Kumar, V. (2021). Robotics for customer service: A useful complement or an ultimate substitute? *Journal of Service Research*, *24*(1), 9–29. doi:10.1177/1094670519878881

Yan, C., Yang, Y., & Liu, Y. (2023). Two stage ensemble algorithm based on clustering quality. *Journal of Jilin University: Science Edition*, *61*, 899–908. doi:10.13413/j.cnki.jdxblxb.2022246