Enhancing Chatbot Responses Based on Natural Language Processing Techniques

Yuelong Li

Software College, South China Normal University, Guangdong, 510631, China
20202033030@m.scnu.edu.cn

Abstract. Advancements in conversational Artificial Intelligence (AI) models, exemplified by ChatGPT, New Bing, and Claude, have substantially improved real-time interaction and virtual assistant capabilities of chatbot systems. The growing recognition of the significance of conversational AI has led to an increased focus on incorporating Natural Language Processing (NLP) methodologies in chatbot training. NLP, a pivotal field within computer science and AI, empowers machines to comprehend, parse, and generate human language. In the context of machine learning, NLP facilitates the understanding and utilization of vast amounts of unstructured textual data, a crucial aspect for data-driven decision making and predictive modeling. This research contends that the excellence of a chatbot should be evaluated based not on the complexity of its responses to human statements but on its ability to closely emulate human conversational logic. The present study aims to explore a series of natural language processing techniques to enhance chatbot responses, rendering them more human-like.

Keywords: Chatbot, Natural Language Processing, Machine Learning

1 Introduction

In recent years, dialogue robots have attracted widespread attention in the field of artificial intelligence research. An increasing number of dialogue robots are being designed as assistants to help people handle tasks more efficiently. However, a common issue with current dialogue robots is that they can misinterpret people's intentions in certain contexts, leading to illogical responses during conversations.

The research background for training dialogue models using Natural Language Processing (NLP) can be traced back to the early stages of artificial intelligence. Early dialogue models were primarily based on rules and templates, lacking flexibility and adaptability. However, with the development of machine learning and deep learning, researchers began exploring the use of NLP techniques to improve the capabilities of dialogue models.

One important aspect in the research background is the development of language models. A language model is a model that can predict the next word or sentence. Traditional language models were based on n-gram statistical methods, but these methods were unable to capture long-distance dependencies. With the rise of neural networks, neural network-based language models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) were proposed [1, 2], which can better model the contextual information of language.

Another important research direction is the Sequence-to-Sequence (seq2seq) model [3-6]. This model uses an encoder-decoder structure to map input sequences to output sequences. In dialogue models, the input can be a user's question, and the output can be the machine's answer. By training on large-scale dialogue datasets, seq2seq models can learn the grammar, semantics, and contextual information of language, thereby generating more accurate and coherent responses [7]. In addition, Generative Adversarial Networks (GANs) have also been applied in dialogue model research [8]. GAN models consist of a generator and a discriminator, and they improve the performance of the generator through adversarial training. In dialogue models, the generator can generate realistic responses, while the discriminator evaluates the quality of the responses. Through iterative training, dialogue models can generate more natural and fluent conversations.

This research aims to use a seq2seq model to train a conversational chatbot and applied a series of preprocessing techniques to the training dataset. The objective of this study was to enhance the naturalness, accuracy, and adaptability of the dialogue model, enabling it to interact and communicate better with humans.

2 Method

2.1 Data Preprocessing

This study utilized the Cornell Movie-Dialogs Corpus as the training dataset for the conversational chatbot. The Cornell Movie-Dialogs Corpus is a widely used dialogue dataset released by researchers at Cornell University in 2011. It consists of 220,579 dialogues from 617 movies, involving 10,292 movie characters portrayed by 9,035 actors. The main characteristic of this dataset is its extensive collection of natural language dialogues, covering various topics and situations, providing rich training data for dialogue models. The conversations range from casual everyday chitchat to emotionally intense exchanges, enabling the model to be trained on different types of dialogues.

During training with this dataset, a common approach is to use the previous sentence of a dialogue as the input and the subsequent sentence as the target output. This allows the model to learn how to generate appropriate responses based on the context. By employing this sequence-to-sequence training method, the model can better understand the semantics and contextual information of the conversation, resulting in more accurate and coherent replies. In order to facilitate the processing of the dataset, this study will create a well-formatted data file where each line contains a pair of query and response sentences.

The initial step in this study entails parsing the text file encompassing the dialogue data. Each line within the file undergoes dissection into a dictionary of distinct fields, which are subsequently integrated into a cohesive dialogue data structure. During successive training phases, sentence pairs are extracted from the dialogue data to facilitate additional natural language processing tasks. Subsequently, the imperative task involves constructing a vocabulary and loading the query-response sentence pairs (dialogues) into memory. It is pertinent to highlight that the word sequences in consideration are not yet mapped to a discrete numerical space. Hence, for the present investigation, the creation of an index from the words within the dataset is deemed indispensable. To complete this, a class is created that will store the mapping from words to indices, the reverse mapping from indices to words, the count of each word, and the total number of words. This class provides methods to add words to the vocabulary, add all words from a sentence, and clean up infrequent words in preparation for further data cleaning. Prior to utilizing the data, certain preprocessing steps are necessary. Initially, Unicode strings are converted to ASCII format. Subsequently, all letters are uniformly transformed to lowercase, and any non-alphabetic characters, with the exception of basic punctuation, are removed. Furthermore, to facilitate training convergence, a predefined upper limit on the sentence length, denoted as MAX_LENGTH, is established. Sentences that exceed this maximum length are filtered out, thereby effectively expediting the training process of the dialogue model. In the preprocessing phase of the subsequent model training, an additional strategy was introduced to expedite convergence by eliminating infrequently used words from the vocabulary. This reduction in the feature space contributes to alleviating the complexity of the model's learning objective function. This study accomplishes this through the following two steps: 1) This study defined a value representing the frequency of word usage and set MIN_COUNT as the minimum word frequency value. Then, this study removed words below the MIN_COUNT threshold. This means that if a sentence contains words with frequencies below the threshold, the entire sentence will be filtered out.

2.2 Preparing Data for the Conversation Model

During the preprocessing stage, considerable effort was dedicated to preparing and cleansing the data, resulting in the creation of a coherent vocabulary object and a collection of sentence pairs. However, the model's ultimate requirement necessitates the input data to be in the form of numerical torch tensors. Utilizing a batch size of 1 facilitates the straightforward conversion of words within the sentence pairs into their corresponding indices in the vocabulary, thereby providing the necessary input for the model. During the experiments, this study aims to accelerate training and take advantage of the parallel computing capabilities of GPUs. The training process necessitates the adoption of mini-batches, which represent smaller subsets of the data [9]. Utilizing mini-batches introduces the need to address variations in sentence lengths within a batch. In order to accommodate sentences of diverse sizes within the same batch, the batch input tensors are resized. Specifically, for sentences shorter than the maximum length (max_length), they are zero-padded after the End-of-Sentence token (EOS_token).

If simply converting the English sentences into tensors by converting words into indices and zero-padding, the size of the tensor will be (batch_size, max_length), and the index dimension will return the complete sequence across all time steps. However, the indexing of batch data along the time dimension necessitates the inclusion of all sequences within the batch. Consequently, the input batch size will be transposed to (max_length, batch_size), thereby facilitating the retrieval of time steps for all sentences in the batch through indexing along the first dimension. This transposition operation will be implicitly handled during the implementation.

2.3 Model Definition

The chatbot's brain will utilize a sequence-to-sequence (seq2seq) model. The goal of a seq2seq model is to take variable-length sequences as input and return variable-length sequences as output using a fixed-size model.

Sutskever et al. discovered that by using two separate RNNs together, this task can be completed [4]. The first RNN acts as an encoder, encoding the variable-length input sequence into a fixed-length context vector. In theory, this context vector (the final hidden state of the RNN) will contain semantic information about the query statement inputted to the chatbot. The second RNN is a decoder, which takes the input text and the context vector and returns the probability of the next word in the sequence and the hidden state to be used in the next iteration.

Encoder. The Encoder RNN constitutes a pivotal component in natural language processing systems, wherein it receives a sentence as input and successively generates tokens, exemplified by words, throughout its iterative operation. Simultaneously, it produces an "output" vector and a "hidden state" vector during each iteration. The hidden state vector is subsequently propagated to the subsequent step, while the output vector is duly recorded. In effect, the encoder facilitates the transformation of the contextual information observed at each position within the sequence into a sequence of points within a high-dimensional space. These points serve as vital inputs to the decoder, facilitating the generation of meaningful output tailored to a given task.

At the heart of the constructed encoder lies a multi-layered GRU, originally proposed by Cho et al. In this context, a bidirectional GRU variant is employed, effectively incorporating two distinct and independent RNNs. One of these RNNs processes the input sequence in its natural order, while the other operates in reverse, thereby capturing both forward and backward temporal dependencies. The outputs of these two networks are combined through summation at each time step. The utilization of bidirectional GRU confers the significant advantage of encoding not only past contextual information but also future context, thus enhancing the comprehensive representation of the input sequence.

To construct the encoder, a sequential series of operations is undertaken. Initially, each word index in the input sequence is transformed into a word embedding, leveraging either a pre-trained or freshly initialized embedding layer. This process enables the model to grasp the semantic representation of individual words. Subsequently, the batched sequences are aggregated, optimizing computational efficiency by reducing the number of RNN steps required for sequences with varying lengths and facilitating the handling of padding within the batch. Following this, the packed sequence is subjected to forward propagation through a GRU layer, which is a specialized type of RNN module adept at capturing and learning long-range dependencies within the data. Upon completion of forward propagation, reverse padding is performed to unpack the output tensor, restoring it to its original padded sequence shape. Subsequently, the outputs of the bidirectional GRU are summed, combining the acquired knowledge from both forward and backward passes, thereby establishing a more comprehensive representation of the input sequence. Ultimately, the summed output and the final hidden state of the GRU layer are returned, providing a basis for subsequent decoding stages or other NLP model tasks.

Decoder. The decoder RNN generates a response sentence token-by-token. It uses the context vector from the encoder and its internal hidden state to generate the next word in the sequence. It continues generating words until the output is the EOS_token, which represents the end of the sentence. A common issue with a vanilla seq2seq decoder is that if relying solely on the context vector to encode the meaning of the entire input sequence, the information may be lost. This limitation becomes especially pronounced when dealing with long input sequences, greatly restricting the decoder's capabilities.

In response to this issue, Bahdanau et al. devised an "attention mechanism," a fundamental construct that enables the decoder to selectively concentrate on specific segments of the input sequence, rather than relying on a static context throughout each iteration [5]. At a conceptual level, the attention mechanism computes attention scores by considering both the prevailing hidden state of the decoder and the outputs generated by the encoder. The resulting attention weights possess a congruent dimensionality as that of the input sequence, facilitating their utilization in forming a weighted sum with the encoder outputs. This weighted sum delineates the segments within the encoder outputs that merit heightened attention during the decoding process.

Luong et al. improved upon the foundational work of Bahdanau et al. by introducing "Global attention [3]." The key difference is that for "Global attention," all the hidden states of the encoder were considered, whereas Bahdanau et al.'s "Local attention" only considers the hidden state of the encoder at the current step [5]. Another difference is that with "Global attention," this study only uses the hidden state of the decoder at the current step to compute the attention weights (or energies). Bahdanau et al.'s attention calculation requires knowledge of the decoder's state from the previous step [5]. Additionally, Luong et al. provided various methods for computing attention weights (energies) between the encoder outputs and the decoder outputs, referred to as "score functions" [3]: 1) Dot Product: The attention score is computed as the dot product between the decoder's hidden state and the encoder's hidden state. 2) General: The attention score is computed by applying a linear transformation to the decoder's hidden state and taking the dot product with the encoder's hidden state. 3) Concat: The attention score is computed by concatenating the decoder's hidden state and the encoder's hidden state, applying a linear transformation, and passing it through a non-linear activation function.

To work with the decoder, this study begins by obtaining the word embedding of the current input word, which is usually the last predicted word or a given starting token. This process transforms the input word index into a dense vector representation. Next, this study forwards propagates this embedding through a unidirectional GRU layer, which aims to learn and capture data dependencies. Upon obtaining the current GRU output, this study computes the attention weights, which reflect the importance of each encoder output with respect to the decoder's current hidden state. This is achieved by employing an attention mechanism tailored for sequence-to-sequence models.

Once this study has the attention weights, this study multiplies them by the encoder output to produce a new "weighted sum" context vector. This step essentially combines specific parts of the encoded input sequence with the decoder's current state, allowing the model to focus on relevant information at each decoding step. Following this, this study uses Luong's equation 5 to concatenate the weighted context vector and the current GRU output, generating a more informed representation for the current decoding step.

Subsequently, this study employs Luong's equation 6 to predict the next word in the target sequence, which utilizes the concatenated vector to produce an output logit without the softmax activation function. The softmax activation is typically applied later in conjunction with a loss function during the training process. Finally, the decoder returns the unnormalized output and its final hidden state. This process is repeated for each decoding step until the model produces an end-of-sequence token or reaches a predetermined maximum sequence length.

2.4 Defining Training Steps

This study will use some clever techniques to better train the dialogue model.

The first technique is called "teacher forcing" [10]. This means that, with a certain probability set by the teacher_forcing_ratio, this study uses the current target word as the next input for the decoder instead of using the decoder's current guess. This technique acts as training wheels for the decoder and helps with more effective training. However, teacher forcing can lead to instability in the model during inference because the decoder may not have enough opportunities during training to truly generate its own output sequence. Therefore, this study needs to be careful about how this study sets the teacher_forcing_ratio and not be misled by quick convergence.

The second technique is gradient clipping. This is a common technique used to combat the "exploding gradient" problem [11]. Essentially, by clipping or thresholding the gradients to a maximum value, this study can prevent the gradients in the loss function from exponentially growing and causing overflow (NaN) or going beyond the gradient.

In the operation process for training a sequence-to-sequence encoder-decoder model, this study first calculates the forward pass of the entire batch of input sequences through the encoder. This generates encoded representations of the input sequences and the final hidden state of the encoder. Then, this study initializes the decoder input with the Start of Sentence (SOS) token, and the decoder's initial hidden state is set to the encoder's final hidden state.

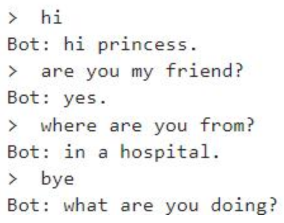
For each step in the target sequence, this study forward propagates the decoder's input through the decoder. The next decoder input is set based on whether this study uses the teacher forcing algorithm or not. If teacher forcing is used, this study sets the next decoder input to the current ground-truth target word. Without teacher forcing, this study sets the next decoder input to the current decoder output.

This study calculates and accumulates losses at each step by comparing the decoder's output with the actual target sequence. After processing the entire sequence, this study performs backpropagation to compute gradients of the loss with respect to the model's parameters. Next, this study clips the gradients to prevent the exploding gradient problem.

Finally, this study updates the encoder and decoder model parameters using an optimization algorithm. The model's parameters are adjusted to minimize the accumulated loss, improving the model's performance on the training data. This process is repeated for multiple epochs, allowing the model to learn patterns and relationships between input and output sequences for accurate translations, summarizations, or any other sequence-to-sequence tasks. Then this study can start training this conversation robot.

3 Results and Discussion

This article uses natural language processing techniques to preprocess the dialogue model, which enables the chatbot to train faster and respond more accurately after training. The conversation results after training completion are shown in the following Figure 1.



```
> hi
Bot: hi princess.
> are you my friend?
Bot: yes.
> where are you from?
Bot: in a hospital.
> bye
Bot: what are you doing?
```

Fig. 1. The conversation results based on the reply.

After training is completed, the relationship between the loss rate and the training progress of this model is shown in the following Figure 2.

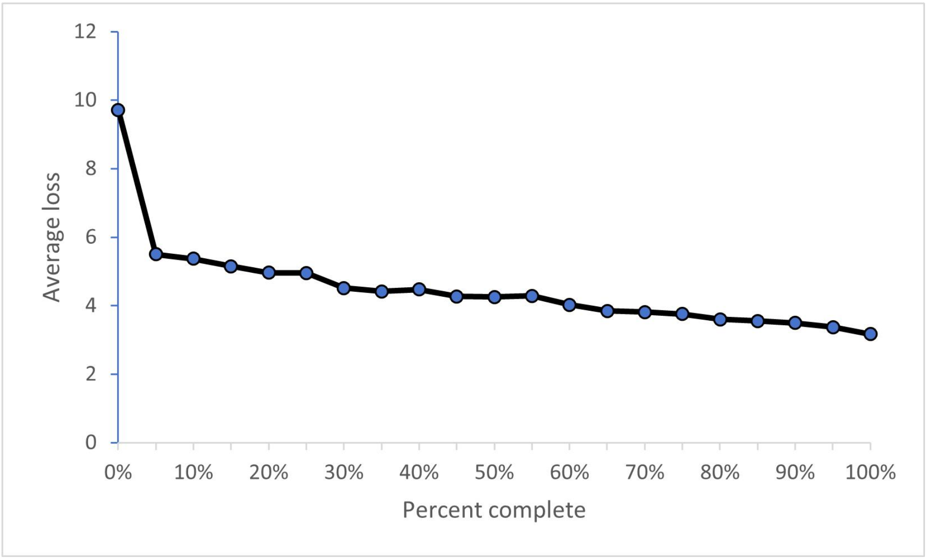


Fig. 2. The variation of the loss rate during the training process.

Evidently discernible is the noteworthy reduction in the rate of loss prior to attaining a 10% advancement within the training regimen. However, within the interval spanning from 20% to 60% progression, a discernible deceleration in the pace of loss reduction becomes manifest. Subsequent to traversing the threshold of 60% progress, the attenuation in the rate of loss diminution becomes even more pronounced, culminating in a state where the loss rate achieves a state of near constancy.

A conjecture may be posited regarding potential determinants contributing to the gradual abatement in loss reduction during the later stages of the training procedure:

Inceptive among these factors could be attributed to intrinsic attributes pertaining to the training dataset itself. The corpus employed in this investigation emanates from cinematic dialogues, replete with contextual and conversational underpinnings that are germane to the filmic medium. However, direct transposition of these dialogic excerpts into the training milieu designed to imbue a chatbot with aptitude for quotidian dialogues may inadvertently introduce perturbations, thereby exerting a deleterious influence upon the efficacy of loss rate mitigation during the latter epochs of training.

Secondarily, one may contemplate that the conversational AI architecture under scrutiny may have attained a state of saturation, denoting that the model has ostensibly internalized a substantial gamut of rudimentary conversational paradigms and attributes, gleaned from an extensive repository of cinematic dialogues. Consequently, the discernment of novel informational substrates within these cinematic dialogues for the purpose of further enhancing predictive fidelity with respect to everyday conversational utterances might be considerably curtailed. This state of equipoise arguably contributes to the observed lethargy in the reduction of loss rates. In addition, some important conclusions can be derived based on the results:

1. Conversion into Tensors: By converting sentences into tensors, it becomes possible for dialogue robots to process and model linguistic information more effectively. Tensors are multi-dimensional arrays that can store and manipulate data efficiently, making them ideal for natural language processing tasks.

2. Word Indexing and Zero Padding: By indexing words in sentences and padding with zero vectors, it becomes possible to standardize input data size for the seq2seq model. This makes it easier to train models across multiple input lengths and, in turn, ensures better generalization of the model.

3. Using the Seq2Seq Model: The sequence-to-sequence (seq2seq) model is ideal for training dialogue robots because it facilitates the encoding and decoding of complex linguistic structures. This model can handle a range of tasks such as machine translation, summarization, and conversational agents.

4. Using Teacher Forcing: Teacher forcing is a technique used in training dialogue robots where the expected output (target sequence) is provided as an input to the model during training. This helps the model learn quickly and make better predictions in shorter periods.

5. Using Gradient Clipping: Gradient clipping is a technique used in training that helps prevent the "exploding gradient" problem where the gradients may become too large during training. It is used to tweak the learning curve and ensure that the model converges more quickly and effectively.

Overall, these techniques help improve the accuracy and fluency of dialogue robots, making them more effective at understanding and responding to user requests.

These score functions allow the decoder to focus on different aspects of the encoder outputs during the attention calculation. The attention weights are then used to compute a weighted sum of the encoder outputs, which is combined with the decoder's hidden state to generate the context vector for the current decoding step. This context vector is then used to predict the next word in the sequence.

4 Conclusion

This study delves into the realm of enhancing the efficacy of training dialogue robots through more efficient methodologies. The prevalent issue of dialogue robots misunderstanding human intentions, thereby producing illogical responses, serves as the impetus for this study. The primary objective entails devising refined natural language processing techniques to optimize the training strategies employed for dialogue robots, thereby augmenting the precision and fluency of their responses. This pursuit is underpinned by the application of advanced machine learning and deep learning methodologies. The study culminates in the utilization of a seq2seq model to successfully train a conversational chatbot, fostering improved interaction and communication with human interlocutors.

References

1. Cho, K., et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724-1734, (2014).
2. Chung, J. Y., et al.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746-1751, (2014).
3. Luong, M. T., et al.: Effective Approaches to Attention-based Neural Machine Translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1412-1421, (2015).
4. Sutskever, I., et al.: Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems 27 (NeurIPS 2014), pages 3104-3112, (2014).
5. Bahdanau, D., et al: Neural Machine Translation by Jointly Learning to Align and Translate. Proceedings of the 2015 International Conference on Learning Representations (ICLR), (2015).
6. Vaswani, A., et al.: Attention Is All You Need. Proceedings of the 2017 Conference on Neural Information Processing Systems (NeurIPS), pages 6000-6010, (2017).
7. Vinyals, O., et al.: A Neural Conversational Model. Proceedings of the 2015 Conference on Neural Information Processing Systems (NeurIPS), pages 2296-2304, (2015).
8. Goodfellow, I., et al.: Generative Adversarial Nets. Proceedings of the 2014 Conference on Neural Information Processing Systems (NeurIPS), pages 2672-2680, (2014).

9. Keskar, N. S., et al.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. Proceedings of the 2017 International Conference on Learning Representations (ICLR), (2017).
10. Bengio, S., et al.: Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. Advances in Neural Information Processing Systems 28 (NeurIPS 2015), pages 1171-1179, (2015).
11. Pascanu, R., et al.: On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning (ICML), pages 1310-1318, (2013).