Enhancing Response Generation Using Chat Flow Identification

Ruirui Li
UCLA
rrli@cs.ucla.edu

Jyun-Yu Jiang
UCLA
jyunyu@cs.ucla.edu

Chelsea J.-T. Ju
UCLA
chelseaju@cs.ucla.edu

Cheryl Flynn
AT&T Lab
cflynn@research.att.com

Wen-ling Hsu
AT&T Lab
hsu@research.att.com

Jia Wang
AT&T Lab
jiawang@research.att.com

Wei Wang
UCLA
weiwang@ucla.edu

Tan Xu
AT&T Lab
tanxu@research.att.com

ABSTRACT

Conversational artificial intelligence (AI) has been widely applied to different products and services in the past few years. As a common engagement channel between customers and businesses, live chat can benefit substantially from conversational AI. Currently, businesses spend a tremendous amount of time and money to train and educate chat agents. The training and educating processes include providing agents with chat flows, FAQs, and answer templates to guide chat agents answering customer questions efficiently and properly. However, creating such training materials for chat agents is time-consuming. In addition, the materials need to be constantly updated as new services and products are deployed to customers. In this work, we study the problem of extracting common chat flow from online customer care chats to enhance question response generation, with the goal of using them to facilitate chat agent training.

KEYWORDS

Customer Care, Chat flow, Response Generation

ACM Reference Format:
Ruirui Li, Jyun-Yu Jiang, Chelsea J.-T. Ju, Cheryl Flynn, Wen-ling Hsu, Jia Wang, Wei Wang, and Tan Xu. 2019. Enhancing Response Generation Using Chat Flow Identification. In *Proceedings of ACM KDD conference (KDD'18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Conversational AI has been widely deployed in different products and services in industry in the past few years. Google launched Smart Reply, an email response recommender system that suggests short replies for 10% of Gmail volume in the Inbox mobile application [4]. Amazon applied a dual encoder network to generate question responses, which covered more than 70% of customer inquiries in delivery related issues [11]. In addition to introducing the chatbots Xiaoice and Rinna in China in 2014 and in Japan in 2015, respectively, Microsoft integrated Cortana, its AI assistant, into most of the products. These conversational AIs strengthen the communication, the connection, and the engagement between customers and businesses.

KDD'18, July 2018, London, United Kingdom

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

Live chat services, as a common customer engagement channel, not only meet customers' basic requirements but also give businesses a way to increase satisfaction, loyalty, and advocacy from customers. According to [16], 90% of customers consider live chat helpful and 63% of them prefer to return to a website that offers a live chat service. Therefore, a key to the survival and success of businesses is to improve live chat qualities. Conventionally, to best serve customers, the chat agents are expected to be extremely knowledgeable about inventories, services, and even website navigation. Therefore, businesses have to spend a tremendous amount of effort and money on training and educating chat agents, especially the new agents. In the training process, well designed guidelines and templates are the major resources to assist agents to answer customer questions properly and efficiently. Unfortunately, customer questions evolve over time as new products and services are deployed. The solutions to the previous tasks may also change. Therefore, it is challenging to come up with approaches that can automatically generate and update solutions to customer questions.

Historical chat logs register interactions between customers and chat agents. These logs are valuable sources for generating meaningful solutions for live chat training in that they allow us to tap into the wisdom embedded in the chat interactions between customers and agents. We can leverage this wisdom to mine potential solutions to customer questions. Each chat involves one customer and one chat agent. Table 1 shows an example of a raw chat dialogue. The chat dialogue is composed of 12 sequential chat turns. Each turn refers to a message conveyed by either an agent or a customer. The chat starts with greetings between a customer and a chat agent. Then the customer expresses his need to arrange a late payment, followed by an acknowledgement from the agent and the corresponding payment arrangement operations. At the end of the chat, there are appreciations exchanged between the customer and the chat agent. As the example shows, historical chats contain customer inquiries and corresponding agent actions. Learning the relationships between them allows us to generate appropriate solutions to customer questions.

There are several challenges in extracting the relationships between customer inquiries and the corresponding agent actions. First, customers use different words to express the same issue. In addition, the adopted construction (e.g., the sentence structure) of the expression also vary from one customer to another. Second, the true intention of a customer is often expressed and clarified in multiple chat turns. Third, the chat messages related to the agent actions do not always immediately follow customer questions. For example, the agent may repeat customer inquiries as confirmations or update the customer on his/her progress as the agent works on a resolution.

Table 1: An example of a raw chat dialogue between a customer and an agent. A and C represent agent and customer, respectively. For privacy reasons, all agent and customer names are replaced by Bob and Alice. Dates are replaced by MM/DD/YYYY. Device makes and models are replaced by device-make and device-model, respectively.

Turn Id	Dialogue content
1	A: good morning thank you for chatting with XXX my name is Bob how may I help you today?
2	C: I want to set up a late payment
3	A: Hi Alice
4	A: I understand that you need to set a late payment on the account
5	A: no problem i am here to help you with that and will be more than glad to assist you
6	A: please allow me a moment while I get access to your account right away.
7	A: Alice what date would you like to set for payment?
8	C: MM/DD/YYYY
9	A: thank you for that information
10	A: Alice I completed the process and set the payment for MM/DD/YYYY
11	C: ok thank u
12	A: you are so welcome.

Table 2: List of symbols

Symbol	Description
t_i	a chat turn from either a customer or a chat agent
D_i	a chat dialogue i composed of a sequence of chat turns <t_1, t_2, ...t_m>
sd_i	a sub-sequence of turns in D_i
ms	the score of the best alignment between two chats
i_{loc}, j_{loc}	the ending positions of the best alignment
$s(t_i, t_j)$	the similarity between turn t_i and turn t_j
h	the turn similarity threshold
$b_{i,j}$	the reward for appending turns t_i and t_j to previous sub-sequences
p	the penalty for skipping turns in chat alignments
x_i/y_i	a sequence of words as inputs/outputs of a training instance
N	total number of the training instances

These issues make it challenging to pinpoint customer questions, agent actions, and the relationships between them.

To address the above challenges and generate appropriate and meaningful question solutions, we first extract common chat subsequences through pairwise chat comparisons. The extracted subsequences incorporate common chat flow in the chats. Next, we apply a generative sequence-to-sequence model to generate question responses. The contributions of this work are two-fold: (1) we propose identifying common chat flow by extracting partial alignments between pairwise chat dialogues, and (2) we apply a hierarchical sequence-to-sequence generative model to generate question responses as potential solutions.

2 METHODOLOGY

In this section, we discuss the methodology to generate responses based on input customer questions. Our proposed approach consists of three stages, including dialogue cleaning, common chat flow extraction, and response generation. In Section 2.1, we discuss dialogue cleaning, which filters out the majority of uninformative chat messages and keeps the essential information. To extract the common flow of conversations in live chat, we propose using dialogue alignment to identify the common chat sub-sequences in Section 2.2. Finally, we discuss our approach of response generation by incorporating the common chat flow in Section 2.3. Table 2 lists the notations we use in this paper.

2.1 Dialogue Cleaning

As shown in Table 1, some turns in the dialogue are greeting messages (e.g., turns 1 and 3), empathy/reassurance messages (e.g., turns 5 and 6), and appreciation messages (e.g., turns 9, 11, and 12). In addition, apology and branded closing messages are also prevalent in customer care chat. These emotion, etiquette padding turns are essential for good customer care but provide no information to help us identify agent actions. In this paper, we refer to these turns as "uninformative". Removing these uninformative turns makes the chat interactions more concise without altering the general understanding of the conversation. Therefore, the tasks of dialogue cleaning focus on identifying the uninformative turns and removing them from the dialogues.

We employ a supervised learning approach and consider identifying the uninformative turns as a binary classification task. We manually label a subset of turns from all dialogues to be either informative or uninformative. These labeled turns are used as training instances to learn a classifier. After training, we apply the classifier to the remaining turns to infer their statuses. Turns labeled as uninformative are filtered out from the original chat.

A classifier can be sensitive to the training distribution [1]. Randomly sampling the turns from all dialogues often results in an unbalance training set, especially when the sample space is large. This is mainly because different numbers of turns sharing similar syntax and semantics. To mitigate this issue, we first cluster all turns into K groups with mini-batch K-means [14] based on their distributed representations trained by *paragraph2vec* [7]. The optimal value of K is determined by the Elbow justification [6]. Turns are proportionally sampled from clusters and manually labeled as the training data. A binary classifier is then trained and applied to predict the labels of the remaining unlabeled turns (See Section 3.2 for model selection). Finally, all labeled and predicted uninformative turns are filtered out, thereby constructing the clean chats.

2.2 Common Chat Flow Extraction

Even though some chat dialogues involve the same types of customer inquires, variations of word choices and sentence structures can increase the difficulty of learning the relationships between customer questions and agent actions in response generation. Hence, there is a necessity to extract common dialogue sub-sequences concisely reflecting the common chat flow. In light of this, we identify common dialogue sub-sequences via pairwise dialogue comparisons, which aims to extract the sub-sequences from two dialogues that are the most similar to each other.

Given two dialogues D_1 and D_2, modeled as two sequences of turns <$t_1^1, t_2^1, \cdots, t_m^1$> and <$t_1^2, t_2^2, \cdots, t_n^2$>, respectively, the goal of common dialogue sub-sequence extraction is to identify two

dialogue sub-sequences, sd_1 from D_1 and sd_2 from D_2, such that the similarity between them is optimized in terms of chat semantics and flow. Here we assume the quality of an alignment can be represented as the sum of the similarities between aligned turn pairs deducted by the penalties of skipped turns in two sub-sequences.

Algorithm 1: Common dialogue sub-sequence extraction

Input: dialogue 1: $D_1 =< t_1^1, t_2^1, \ldots, t_m^1 >$,
dialogue 2: $D_2 =< t_1^2, t_2^2, \ldots, t_n^2 >$,
the penalty for turn skip p, and the similarity threshold h.
Output: dialogue sub-sequences sd_1 and sd_2.

1 **Initialization:** $DP = 0$, $ms = 0$, $i_{loc} = 0$, $j_{loc} = 0$;
2 **for** i *in range of* $|D_1|$ **do**
3 **for** j *in range of* $|D_2|$ **do**
  //Reward for appending turns t_i^1 and t_j^2;
4 $b_{i,j} = tan(s(t_i^1, t_j^2) - h)$;
5 $DP_{i,j} = max(DP_{i-1,j-1} + b_{i,j}, DP_{i,j-1} - p, DP_{i-1,j} - p, 0)$;
  //Update if better alignment found;
6 **if** $DP_{i,j} > ms$ **then**
7 $ms = DP_{i,j}$;
8 $i_{loc} = i$;
9 $j_{loc} = j$;
10 Backtrack from i_{loc} and j_{loc} to find sd_1 and sd_2;
11 **Return** sd_1 and sd_2;

Based on dynamic programming, Algorithm 1 summarizes this process of identifying sd_1 and sd_2. $DP_{i,j}$ indicates the highest score among all partial alignments ending with either t_i^1 or t_j^2. The algorithm aims to correctly fill the whole table DP and find the position (i_{loc}, j_{loc}) with the highest score ms. Note that a higher ms indicates a longer and closer alignment. Since the task of computing $DP_{i,j}$ has the optimal substructure, the computation of $D_{i,j}$ only depends on $D_{i-1,j-1}$, $D_{i-1,j}$, and $D_{i,j-1}$. More precisely, updating $DP_{i,j}$ from $DP_{i-1,j-1}$ involves appending t_i^1 and t_j^2 to the aligned sub-sequences identified at $(i-1, j-1)$. We first calculate the reward of appending these two turns, $b_{i,j} = \tan\left(s(t_i^1, t_j^2) - h\right)$, to be added into $DP_{i-1,j-1}$. The reward $b_{i,j}$ is a function of the similarity $s(t_i^1, t_j^2)$ between the two turns with a similarity threshold h. The function tan is chosen as the reward function because of its smoothness and the ability to reward/penalize extremely high/low similarities. Therefore, $DP_{i,j}$ increases from $DP_{i-1,j-1}$ by a positive reward when t_i^1 and t_j^2 are similar. On the other hand, dissimilar turns lead to a negative reward and decrease the score. Another situation happens when t_i^1 or t_j^2 is skipped in the alignment. The score $DP_{i-1,j}$ or $DP_{i,j-1}$ is deducted by a penalty score p for the skipping. As shown in the fifth line of Algorithm 1, $DP_{i,j}$ is updated to the largest value of the three potential options. In addition, if $DP_{i,j}$ becomes less than 0, $DP_{i,j}$ will be reset to 0, indicating that (i, j) is a new starting position to find the best partial alignment between the two dialogues.

Table 3 shows a toy example of two dialogues regarding late payment and Table 4 shows their corresponding turn-level pairwise similarities based on their latent representations. Based on these two dialogues, Figure 2 shows the computation of DP matrix according to Algorithm 1 with different similarity thresholds. The parameter settings are discussed in Section 3.3. In addition to the DP scores, we also highlight the path giving the best alignment. Through backtracking, we can identify the two sequences of the turns providing the best partial alignment between D_1 and D_2; i.e., $< t_1^1, t_2^1, t_3^1, t_5^1 >$ and $< t_1^2, t_2^2, t_3^2, t_4^2 >$.

2.3 Response Generation

Incorporating the common chat flow together with the clean dialogues, we apply a generative sequence-to-sequence model to learn the dependency between turns. The objective function of our model is shown in Equation 1.

$$J(\Theta) = \frac{1}{N} \sum_{i=1}^{N} P(y_i|x_i) \tag{1}$$

where Θ is a set of parameters defining the model; x_i, as inputs, is a sequence of words; y_i, as corresponding outputs, is also a sequence of words; N is the number of training instances. The goal is to maximize $J(\Theta)$ through considering the relationship between the previous chat turns, x_i, and the next chat turn, y_i.

Figure 1 shows the network structure. It is in the form of an encoder-decoder architecture. The encoder network ingests incoming chat contents, and the decoder network generates outgoing responses. More precisely, the turn-level encoder encodes the word sequence information in a chat turn and updates the internal recurrent state. After the last word has been processed, the recurrent state can be considered as a compact order-sensitive encoding of the chat turn. Given that dialogues are multiple-turn interactions between customers and chat agents, a session-level encoder is adopted to further capture this information. The session-level encoder takes the outputs of the turn-level encoder as inputs and its recurrent state encodes the entire interactions in the chat. The decoder takes the session-level recurrent state as inputs, deciphers the encoded information, and generates outputs sequentially.

Each training instance, denoted as (x_i, y_i), is composed of an input text x_i and an output text y_i. The training instances include the data generated from the cleaned dialogues, and the common dialogue sub-sequences extracted from pairwise comparisons as described in Section 2.2. The goal of incorporating these common subsequences is to slightly change the conditional distribution, $P(y_i|x_i)$, of the text in the dialogue, which optimizes the sequence dependency and consequently reflect the common chat flow. Here we use an example to demonstrate this process. Consider two dialogues $D_1 =< t_1^1, t_2^1, t_3^1, t_4^1, t_5^1 >$ and $D_2 =< t_1^2, t_2^2, t_3^2, t_4^2 >$ in Table 3, and two sub-sequences $sd_1 =< t_1^1, t_2^1, t_3^1, t_5^1 >$ and $sd_2 =< t_1^2, t_2^2, t_3^2, t_4^2 >$ the sources to generate training instances. Including training instances generated from sd_1 and sd_2 will decrease the likelihood of generating less common chat content, t_4^1, and encourage the generation of the common chat content, t_5^1, conditioned on the previous chat interactions. As shown in the example, incorporating extra training instances from common chat flow help us generalize the generated responses.

3 EXPERIMENT

In this section, we first describe the dataset and the performance of dialogue cleaning in Sections 3.1 and 3.2, respectively. Then we show the common sub-sequence extraction results in Section 3.3. Lastly, we show the response generation results in Section 3.4.

3.1 Dataset

In this paper, we use customer care chat logs collected from a large U.S. cellular service provider over a period of several months. Each record in the logs corresponds to a customer care chat, where an example chat dialogue is shown in Table 1. Sensitive and private

Table 3: Two dialogues about late payment

Turn Id	Dialogue content in D_1	Dialogue content in D_2
1	C: I was wondering if I could arrange late payment	C: I want to set up a late payment
2	A: I understand that you would like to setup a payment arrangement	A: I understand that you need to set a late payment on the account
3	A: What date do you want to pay?	A: Bob what date would you like to set for the payment?
4	C: it is really great. :0	A: Bob I completed the process and set the payment for MM/DD/YYYY
5	A: I completed the process	

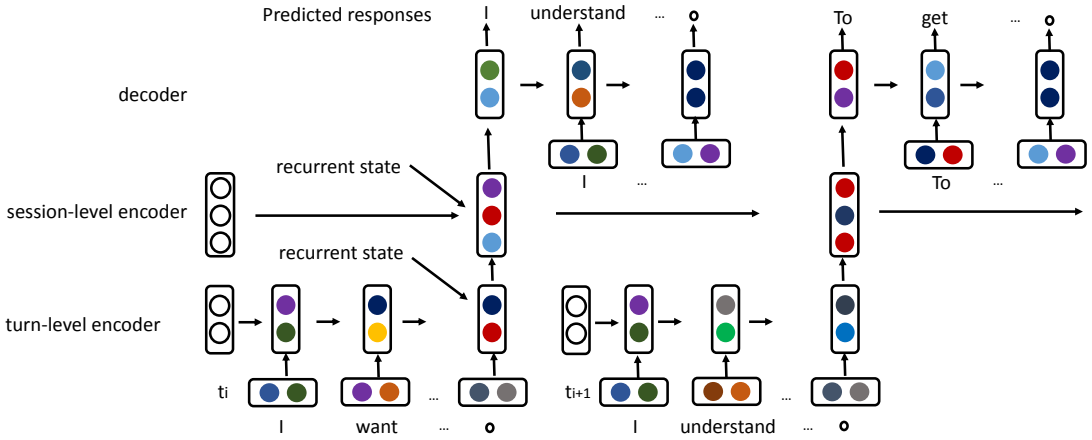

Figure 1: Network structure of the generative model

Table 4: Pairwise turn similarities between D_1 and D_2

	t_1^1	t_2^1	t_3^1	t_4^1	t_5^1
t_1^2	0.328	0.167	0.109	0.356	0.197
t_2^2	0.371	0.374	0.133	0.192	0.081
t_3^2	0.148	0.328	0.464	0.130	0.135
t_4^2	0.034	0.202	0.049	0.321	0.558

customer information was removed from the chat dialogues prior to conducting any analysis. The chat content field in each chat registers how an agent interacts with the customer, including information related to service issues the customer would like to report as well as actions the agent performed to resolve the issues. Altogether, there are about half a million chat records in our data sample.

3.2 Performance of Dialogue Cleaning

To filter out uninformative chat turns, we manually labeled 6,000 chat turns, where 3,000 were labeled as uninformative turns and the other 3,000 were labeled as informative turns. We used 90% of the labeled turns as training data. The remaining 10% were withheld as test data for evaluation.

We experimented with five different classification methods: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Multilayer Perceptron (MLP), and Gaussian Naïve Bayesian (GNB). Table 5 shows the classification performances of the five classifiers[1] in terms of precision, recall, F1 score, and accuracy. Among the five classifiers, SVM with rbf kernels achieved the best classification performance. We applied SVM to infer the labels of unlabeled turns.

[1]Parameters in these models were tuned through grid search.

Table 5: Classification Performance

Metrics	LR	SVM	RF	MLP	GNB
Precision	0.89	0.94	0.89	0.94	0.77
Recall	0.89	0.94	0.88	0.94	0.77
F1	0.89	0.94	0.88	0.94	0.77
Accuracy	0.887	0.943	0.883	0.940	0.765

Table 6: An example of the cleaned chat dialogue

Turn Id	Dialogue content
2	C: I want to set up a late payment
4	A: I understand that you need to set a late payment on the account
7	A: Alice what date would you like to set for payment?
8	C: MM/DD/YYYY
10	A: Alice I completed the process and set the payment for MM/DD/YYYY

Table 6 shows the cleaned version of the chat example in Table 1 after filtering out the uninformative turns. The turns involving greetings, empathy/reassurance, and appreciations were successfully removed.

3.3 Performance of Common Dialogue Sub-sequences Extraction

Table 7 shows four extracted pairwise dialogue sub-sequences. The first column shows the dialogue turns in one dialogue and the second column shows the corresponding turns aligned in another dialogue. Different dialogue sub-sequences are separated by a blank line. For example, the first four lines of the table show two dialogue sub-sequences that reflect the common chat flow in pin verification. The next sub-sequence pair expresses the common starting actions regarding adding a new cellphone to mobile services. We list two more extracted sub-sequence pairs for readers' interests.

Table 7: Pairwise common sub-sequences identified by local alignments

Sub-sequence in dialogue D_1	Sub-sequence in dialogue D_2
A: For security purposes I will send you a pin to your cell phone via text.	A: Alice to get started I will send you a pin to your cell phone via text.
A: Once you receive the pin please post it in your chat	A: Once you receive the pin please post it in our chat.
A: You can receive the pin in any line of your email	A: Which line can I send the pin Alice?
A: Pin sent successfully	A: Pin sent.
C: I need to add a different phone to the line ending in XXXX	C: I need to activate my network connection on the new phone. it's still attached to my old one.
A: to get started may I get your wireless number?	A: to get started please provide me with your full name and mobile number.
A: may I please have the imei of the new device and the iccid of the sim card.	A: Alice is this your new device?
A: do you mean cancel the line Alice?	A: it's sad to know that you want to cancel your services. if you don't mid me asking may i know why?
A: I'll make this quick and seamless. by the way are you using wifi while are chatting?	A: I'll make this quick and seamless. by the way are you using wifi while are chatting?
A: please make sure your wifi connection is with range so we can avoid getting disconnected.	A: please make sure your wifi connection is with range so we can avoid out chat from getting disconnected.
A: lastly chats are likely to disconnect if you navigate outside this chat window. so stay until we complete everything, okay?	A: lastly chats are likely to disconnect if you navigate outside this chat window. so stay until we complete everything, okay?
A: may I have the phone model please so I can check it for you	A: may I know please the brand and the model of the phone?
C: device-make device-model	C: device-make device-model
A: you can click here to open the page in a new window tab I will be there on this window tab when you are ready to continue.	A: you can click here to open the page in a new window tab I will be there on this window tab when you are ready to continue.

		t_1^2	t_2^2	t_3^2	t_4^2
	0	0	0	0	0
t_1^1	0	1.274	1.689	0.689	0
t_2^1	0	0.274	2.993	2.960	1..960
t_3^1	0	0	1.993	5.577	4.577
t_4^1	0	1.551	0.993	4.577	6.781
t_5^1	0	0.551	0	3.577	8.014

(a) Alignment result when h = 0.2

		t_1^2	t_2^2	t_3^2	t_4^2
	0	0	0	0	0
t_1^1	0	0.281	0.704	0	0
t_2^1	0	0	1.015	0.982	0
t_3^1	0	0	0.015	2.644	1.644
t_4^1	0	0.563	0	1.644	2.854
t_5^1	0	0	0	0.644	4.168

(b) Alignment result when h = 0.3

		t_1^2	t_2^2	t_3^2	t_4^2
	0	0	0	0	0
t_1^1	0	0	0.205	0	0
t_2^1	0	0	0.235	0	0
t_3^1	0	0	0	1.374	0.374
t_4^1	0	0.063	0	0.374	1.084
t_5^1	0	0	0	0	2.423

(c) Alignment result when h = 0.35

		t_1^2	t_2^2	t_3^2	t_4^2
	0	0	0	0	0
t_1^1	0	0	0	0	0
t_2^1	0	0	0	0	0
t_3^1	0	0	0	0.642	0
t_4^1	0	0	0	0	0
t_5^1	0	0	0	0	1.566

(d) Alignment result when h = 0.4

Figure 2: Alignment results for different settings of h

Here we discuss the parameter settings in Algorithm 1. There are two key parameters, the similarity threshold h and the penalty p for turn skips. $h \in [0, 1]$ defines the similarity boundary that determines whether to reward or penalize when appending additional chat turns to previously aligned sub-sequences. A larger h requires higher turn similarities for rewards during alignments, which leads to closer but shorter aligned sub-sequences. A smaller h relaxes turn similarity requirements during alignments, which leads to longer aligned sub-sequences but takes the risk of including dissimilar chat contents. Figures 2a, 2b, 2c, and 2d show the alignment results of dialogues D_1 and D_2 in Table 3 when setting h equal to 0.2, 0.3, 0.35, and 0.4, respectively. The red arrows and shaded entries show the best alignments between the dialogues. For example, when setting h equal to 0.2 and 0.3, the sub-sequences identified are both $<t_1^1, t_2^1, t_3^1, t_5^1>$ and $<t_1^2, t_2^2, t_3^2, t_4^2>$. When increasing h to 0.35, the common sub-sequences become $<t_2^1, t_3^1, t_5^1>$ and $<t_2^2, t_3^2, t_4^2>$. The first turns t_1^1 and t_1^2 in each dialogue are excluded, since their similarity, 0.328, is less than the threshold h. When further increasing h to 0.4, the extracted sub-sequences reduce to only one pair of matched turns, t_5^1 and t_4^2. In experiments, we calculated similarities between sampled turn pairs. h was selected based on the similarity distribution. The second parameter, p, defines a non-negative penalty of skipping a chat turn during the alignments. Selecting a larger p is more likely to chop the aligned sequences into pieces of shorter ones; while selecting a smaller p is more tolerant to skipping dissimilar chat messages. The selection of p also depends on the adopted reward function such that the penalty for turn skips and the penalty for appending dissimilar turns are in the same scale. In our experiments, we set h and p to 0.3 and 0.1, respectively.

3.4 Examples of Generated Dialogues

Table 8: Examples of generated dialogues

Input 1	**I need to cancel the insurance of the plan**
Re:1	May I have your name please?
Re:2	I am now checking the insurance on your account.
Re:3	I have successfully removed the insurance on your account
Re:4	I have successfully applied the credit on your account
Input 2	**Is there an outage nearby?**
Re:1	I can help you with the network issue.
Re:2	May I have your wireless number?
Re:3	I am now checking the network area.
Re:4	I am now checking the network bars.
Input 3	**I want to check my account**
Re:1	May I have your name please?
Re:2	I can help you with the concern about the charges on your account.
Re:3	May I have your permission to access your online account with you today in order to assist you?
Re:4	I am now checking the details on your account.

Table 8 shows three dialogues generated based on different input questions.

Given the first question "I need to cancel the insurance of the plan", the generated agent actions start with customer name verification. Then a sequential operations regarding customer account checking, insurance removal, and credit recovery are conducted. For the second question "Is there an outage nearby?", the generated agent actions start with concluding it as a network issue, followed by user wireless number acquisition, and concludes by investigations of network area and bars. The third example is related to checking customer accounts. The generated agent actions include the acquisition of customer names and permissions to access the account. Then account details are investigated by the agent.

4 RELATED WORK

The ability to generate coherent and semantically meaningful text plays a key role in natural language processing applications.

Long short-term memory (LSTM) [3] and other recurrent neural network (RNN) models have demonstrated excellent performance on generating meaningful and grammatically correct sentences in sequence generation tasks, such as machine translation [12, 15], poem generation [19], and image captioning [5, 17]. The majority of generative methods use the deep learning technique known as sequence-to-sequence translation [15]. A LSTM model is used to encode an input sequence to a vector, and then another LSTM model is used to decode the vector to a target sequence. However, the decoding is often quite brittle, as errors may accumulate over time. [13] addresses this issue by proposing a sequence level training algorithm that directly optimizes metric used at test time (such as BLEU or ROUGE). [8] applies a hierarchical LSTM model, which builds embeddings for words and sentences, and then decodes the embeddings to reconstruct the original paragraph.

[9] first attempts to apply adversarial training on sequence generation. SeqGAN [18] extends generative adversarial network with reinforcement learning. It models the text generation as a sequential decision-making process, where the state is previously generated words, the action is the next word to be generated, and the generator is a stochastic policy that maps the current state to a distribution over the action space. The discriminator, a classifier, takes in the real and generated text and generates reward signals to update the generator. LeakGAN [2] is proposed to address the sparse reward issue in previous GAN solutions. It further uses features extracted from the discriminator as a step-by-step guidance to train the generator. The performance improves significantly, especially in long text generation. RankGAN [10] claims that the richness inside the sentences constrained by binary prediction is too restrictive and relaxes the training of the discriminator to a learning-to-rank optimization problem.

In this paper we experiment with using hierarchical sequence-to-sequence method as our generative model, and plan to incorporate adversarial training in future work.

5 CONCLUSIONS

In this work, we study the problem of response generation for customer care chat by applying generative models. In the data preparation part, we extract common sub-sequences by pairwise dialogue comparisons, which allow the generative model to optimize more on common chat flow in the conversation. In the model part, we apply a hierarchical encoder to encode input information, where the turn-level RNN encodes the sequential word information while the session-level RNN encodes the sequential interaction information. The decoder deciphers the encoded input conversation and generates output responses. In the future work, we would combine adversarial training and reinforcement learning to further improve the learning performance of the generative model.

REFERENCES

[1] George Forman and Ira Cohen. 2004. Learning from little: Comparison of classifiers given little training. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 161–172.

[2] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long Text Generation via Adversarial Training with Leaked Information. *CoRR* abs/1709.08624 (2017). arXiv:1709.08624 http://arxiv.org/abs/1709.08624

[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[4] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 955–964. https://doi.org/10.1145/2939672.2939801

[5] Andrej Karpathy and Li Fei-Fei. 2017. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 4 (2017), 664–676. https://doi.org/10.1109/TPAMI.2016.2598339

[6] David J Ketchen Jr and Christopher L Shook. 1996. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal* (1996), 441–458.

[7] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. 1188–1196. http://jmlr.org/proceedings/papers/v32/le14.html

[8] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 1106–1115. http://aclweb.org/anthology/P/P15/P15-1107.pdf

[9] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial Learning for Neural Dialogue Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. 2157–2169. https://aclanthology.info/papers/D17-1230/d17-1230

[10] Kevin Lin, Dianqi Li, Xiaodong He, Ming-Ting Sun, and Zhengyou Zhang. 2017. Adversarial Ranking for Language Generation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 3158–3168. http://papers.nips.cc/paper/6908-adversarial-ranking-for-language-generation

[11] Yichao Lu, Phillip Keung, Shaonan Zhang, Jason Sun, and Vikas Bhardwaj. 2017. A practical approach to dialogue response generation in closed domains. *CoRR* abs/1703.09439 (2017). arXiv:1703.09439 http://arxiv.org/abs/1703.09439

[12] Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 11–19. http://aclweb.org/anthology/P/P15/P15-1002.pdf

[13] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence Level Training with Recurrent Neural Networks. *CoRR* abs/1511.06732 (2015). arXiv:1511.06732 http://arxiv.org/abs/1511.06732

[14] D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. 1177–1178. https://doi.org/10.1145/1772690.1772862

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks

[16] teve Olenski. 2016. It's Alive: Why Live Chat Is So Important For Brands. https://www.forbes.com/sites/steveolenski.

[17] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 3156–3164. https://doi.org/10.1109/CVPR.2015.7298935

[18] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 2852–2858. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14344

[19] Xingxing Zhang and Mirella Lapata. 2014. Chinese Poetry Generation with Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 670–680. http://aclweb.org/anthology/D/D14/D14-1074.pdf