arXiv:2104.12755v1 [cs.CL] 26 Apr 2021

Highlights

Auto Response Generation in Online Medical Chat Services

Hadi Jahanshahi, Syed Kazmi, Mucahit Cevik

- An auto-response mechanism for medical conversation is developed.
- A clustering algorithm is proposed to create a canned response set of doctors' messages.
- We tailor deep learning algorithms to address the problem in a dynamic chat conversation.
- Medical word embeddings are incorporated to improve the accuracy of the suggested replies.

Auto Response Generation in Online Medical Chat Services

Hadi Jahanshahi, Syed Kazmi, Mucahit Cevik

Data Science Lab at Ryerson University, Toronto, ON M5B 1G3, Canada

Abstract

Telehealth helps to facilitate access to medical professionals by enabling remote medical services for the patients. These services have become gradually popular over the years with the advent of necessary technological infrastructure. The benefits of telehealth have been even more apparent since the beginning of the COVID-19 crisis, as people have become less inclined to visit doctors in person during the pandemic. In this paper, we focus on facilitating the chat sessions between a doctor and a patient. We note that the quality and efficiency of the chat experience can be critical as the demand for telehealth services increases. Accordingly, we develop a smart auto-response generation mechanism for medical conversations that helps doctors respond to consultation requests efficiently, particularly during busy sessions. We explore over 900,000 anonymous, historical online messages between doctors and patients collected over nine months. We implement clustering algorithms to identify the most frequent responses by doctors and manually label the data accordingly. We then train machine learning algorithms using this preprocessed data to generate the responses. The considered algorithm has two

Email address: `hadi.jahanshahi@ryerson.ca` (Hadi Jahanshahi)

steps: a filtering (i.e., triggering) model to filter out infeasible patient messages and a response generator to suggest the top-3 doctor responses for the ones that successfully pass the triggering phase. The method provides an accuracy of 83.28% for precision@3 and shows robustness to its parameters.
Keywords: Natural language processing, AI and healthcare, Smart chat reply, Medical services, Deep learning

---

1. Introduction

Online chat services have been used across various sectors for providing customer service, tech support, consultancy/advisory, sales support, and education. Compared to in-person and over-the-phone encounters, live chat provides the highest level of customer satisfaction [1]. As more people join online chat platforms, and with the use of smartphones and smartwatches, as well as an increase in on-the-go communication, smart response generation has become an integral part of online chat platforms.

The smart response suggestions have made businesses more productive as well. Since customer inquiries follow a predictable pattern, which is especially true for domain-specific businesses, smart replies allow for quick and accurate responses. The improved efficiency reduces customers' wait times and thereby results in service satisfaction. Smart response systems also enable employees to handle multiple chats simultaneously, and as a result, businesses can save on additional hiring costs as they grow.

As healthcare is moving towards online chat services, the smart response system plays a prominent role in allowing smooth and effective doctor-patient interactions. According to the Association of American Medical Colleges

(AAMC), the demand for physicians will exceed supply in the U.S. by 2032, leading to an approximate shortage of 46,900 to 121,900 full-time physicians [2]. Hawkins [3] reports that the average wait time for a physician appointment for 15 major metropolitan areas in the U.S. is 24.1 days, representing a 30% increase over 2014. Furthermore, Mehrotra et al. [4]'s findings suggest a 60% decline in the number of visits to ambulatory care practices, whereas there has been a rapid growth in telehealth usage during the COVID-19 pandemic. Due to this high imbalance in the doctor-to-patient ratio and the increase in peoples' reluctance to visit doctors in-person for various reasons (e.g., during the pandemic), telehealth has the potential to become an essential component of our daily lives.

To facilitate the patient-doctor e-conversations, we develop a smart response generation approach for an online doctor-patient chat service. We use historical doctor-patient anonymous chats to develop a method applicable to any online doctor consultation service and apps. There exist certain challenges regarding these types of datasets. First, in many cases, patients take multiple chat-turns to convey a message, and it needs to be manually determined what part of the chat must be used to match the corresponding doctor's reply. In addition, extensive data preprocessing is required to correct misspellings, punctuation misuses, and grammatical errors. In our response generation mechanism for the medical chats, we consider various machine learning and deep learning models. Specifically, our algorithm has two steps: a triggering model to filter out infeasible patient messages and a response generator to suggest the top-3 doctor responses for the ones that successfully pass the triggering phase. We observe that response generation mechanisms

benefit considerably from the high performance of the deep learning models for the natural language processing tasks at both phases.

The rest of the paper is organized as follows. In Section 2, we present the related literature and summarize our contributions with respect to the previous studies. We define our problem and solution methodology for smart response generation in Section 3. Afterwards, we summarize our numerical results with the smart response generator using actual patient-doctor conversations in Section 4. The paper concludes with a summary, limitations, and future research suggestions in Section 5.

2. Related work

The effectiveness of the smart response systems has made them popular in industries where user communication is deemed significant. The speed and convenience of simply selecting the most appropriate response make it suitable for high volume and multitask settings, e.g., when an operator has to chat with multiple customers simultaneously. A diverse set of suggested options presents users with perspectives they might otherwise have not considered. The correct grammar and vocabulary in machine-generated responses enhance communication clarity and helps users avoid confusion over the context of a message. These attributes can be crucial for businesses that rely on the speed and accuracy of the information and, most importantly, users who lack English proficiency. Additionally, smart reply systems mitigate risks associated with messaging while driving and some health concerns such as De Quervain's tenosynovitis syndrome [5].

Google's smart reply system for Gmail [6] serves as a means of convenience for its users. With an ever-increasing volume of emails exchanged along with

the rise in smartphone use, generating responses on-the-go with a single tap of the screen can be very practical. One aspect of the end-user utility discussed in this paper is the diversification of the suggested replies. To maximize usability, Google employs rule-based approaches to ensure that the responses contain diverse sentiments, i.e., covering both positive and negative intents. The paper also suggests using a triggering mechanism to detect whether a reply needs to be generated beforehand to save from unnecessary computations and make the model scalable. Uber also devised a one-click-chat model to address driver safety required for responding to customer texts while driving [7]. Their proposed algorithm detects only the intention of the user message, and, using historical conversations, it suggests the most likely replies. The replies are kept short to reduce the time spent reading, and thereby maximizing safety and utility. Galke et al. [8] analyzed a similar problem of response suggestion where users of a digital library ask librarians for support regarding their search. They used information retrieval methods such as TF-IDF and word centroid distance instead of sequence-to-sequence models, noting that such algorithms are more accurate when the training data is limited.

While the models above are task-oriented and designed to accomplish industry-specific goals, they do not address user engagement issues and the motive to make conversations seem more natural. Microsoft XiaoIce used an empathetic computing module designed to understand users' emotions, intents, and opinions on a topic [9]. It can learn the user's preferences and generate interpersonal responses. Yan [10] proposed social chatbot models that serve the purpose of conversing with humans seamlessly and appropri-

ately. Yan et al. [11] devised a conversation system in which there were two tasks involved: response ranking and next utterance suggestion. The response ranking aimed to rank responses from a set given a query, whereas the next utterance suggestion was to proactively suggest new contents for a conversation about which users might not originally intend to talk. They used a novel Dual-LSTM Chain Model based on recurrent neural networks, allowing for simultaneous learning of the two tasks. Similarly, Yan and Zhao [12] designed a coupled context modeling framework for human-computer conversations, where responses are generated after performing relevance ranking using contextual information.

The studies mentioned above discuss the applicability of smart reply and other AI-enabled conversational models in various settings and domains. Smart reply models that are built specifically for online conversations have to adhere to a distinct criterion. In online conversations, users may adopt words and sentence structures differently. The very intention of a user is often expressed and clarified in multiple chats-turns, and responses do not always immediately follow questions or inquiries. To overcome these challenges, Li et al. [13] extracted common sub-sequences in the chat data by pairwise dialogue comparisons, which allow the generative model to optimize more on common chat flow in the conversation. They then applied 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.

Another challenge with regards to smart reply models built specifically for online conversational chats is scalability. Large-scale deployment of online

smart reply models requires energy and resource efficiency. Kim et al. [14] presented the idea of using sentiment analysis to determine the underlying subject of a message, deciding between character vs. word vs. sentence level tokenization, and whether to limit queries to only nouns without affecting the quality of the model. Jain et al. [15] discussed the idea of using conversational intelligence to reduce both the time and the number of messages exchanged in an online conversation. It includes presenting intelligent suggestions that would engage the user in a meaningful conversation and improve dialogue efficiency. Lastly, Lee et al. [16] proposed using human factors to enable smooth and accurate selection of the suggested replies.

Concerning doctor-patient conversation, there have been several studies in recent years to help doctors with artificial intelligence-based diagnostics and treatment recommendations [17, 18, 19, 20]. Nevertheless, to the best of our knowledge, there is no specific example of a smart response mechanism in the healthcare domain. Related studies focused on language models such as chatbots and not on real-time chat conversations. Oh et al. [21] proposed a chatbot for psychiatric counseling in mental healthcare service that uses emotional intelligence techniques to understand user emotions by incorporating conversational, voice, and video/facial expression data. In another study, Kowatsch et al. [22] analyzed the usage of a text-based healthcare chatbot for the intervention of childhood obesity. Their observations revealed a good attachment bond between the participants and the chatbot.

As more emphasis is being placed on the quality of patient-physician communication [23], an AI-based communication model can facilitate direct and meaningful conversations. However, it is essential to consider the ethical

issues related to AI-enabled care [24, 25] as well as the acceptability of AI-led chatbot services in the healthcare domain [26]. There is hesitancy to use this technology due to accuracy in responses, cyber-security, privacy, and lack of empathy.

Our study differs from the aforementioned works in the literature in various ways. For instance, messages exchanged on the Uber platform are typically short and average between 4-5 words. The average length of messages exchanged on an online medical consultation service tends to be longer, e.g., 10 to 11 words on average, and can be up to 100 words, due to the necessity to clearly describe a certain medical condition. Similarly, the suggested responses created on Gmail are shorter. In terms of the corpus size, Uber and Google's general-purpose datasets might reach millions of instances, whereas a typical training data is substantially smaller (e.g., in tens of thousands), especially for a start-up company or local clinics. This challenge augments the complexity of our model in that it should learn proper responses using a smaller dataset. Galke et al. [8] work with a domain-specific dataset consisting of 14k question-answer pairs and generate responses using retrieval-based methods. On the other hand, their model does not consider diversity and only generates one suggested response for every message. Our model prompts multiple responses with different semantic intents, resulting in better utility for the users. Zhou et al. [9], Yan [10] and Yan and Zhao [12] have developed models that are successful in making natural and human-like conversations. However, they are generic and not suitable for a domain-specific task such as ours that should take into account the medical jargon.

In this study, we develop an algorithm for smart response generation in

online doctor-patient chats. Our analysis is aimed at addressing the challenge of generating smart responses in the medical domain with a limited and constantly evolving (e.g. due to entry of new patients and diseases) dataset. We summarize the contributions of our study as follows.

- To the best of our knowledge, this is the first study to propose an auto-response suggestion in a medical chat service. As conversations include medical jargon, we use medical word embeddings and retrain them on our large conversational corpus.
- Our method involves employing a novel clustering approach to create a canned response set for the doctors.
- Our detailed numerical study shows the effectiveness of the proposed methodology on a medical chat dataset. Moreover, our method demonstrates robustness to its parameters. Accordingly, our study provides an empirical analysis of smart auto-response generation mechanisms.
- The proposed method can be used to generate fast smart responses and can be easily integrated into the chat software.

3. Smart Response Suggestion

AI-assisted tools have become increasingly prevalent in the medical domain over the years. As services such as appointment scheduling and doctor consultations move online, there is an increasing need for auto-reply generation methods to increase the overall system efficiency. However, as is the case in any domain-specific application, there are certain challenges in developing smart response mechanisms for online medical chat services. While particular challenges such as scalability and response quality have been addressed in

previous works [6, 7], there has not been much focus on the speed of response generation and disorderly chat flows. We summarize these two issues within the context of a medical chat service as follows.

- **Speed**: As online chats between doctors and patients follow a rapid pace, the model must generate a response instantaneously (i.e., within a second) to be of practical use. This issue does not persist for systems that generate a reply in an offline setting.
- **Disorderly chat flows**: In a chat platform, a message may or may not be followed by an immediate response. There are instances where messages are exchanged in various turns with their orders being completely random. A message may be replied to immediately or at a later turn. This issue does not apply to the works that deal with email exchanges, as most of the emails are a direct response to the previous email, or they have a reply-to option to circumvent this challenge. However, the impact of disorderly chat flows might be magnified in doctor-patient chats as the doctors, who might be overwhelmed by conversations, are typically slower than the patients.

We address both of these challenges through our comprehensive analysis. According to the challenges and the available data, we consider the below-described steps to construct our suggested response mechanism.

3.1. Data preparation

In this study, we use a dataset obtained from Your Doctors Online[1], which is an online application that connects patients with doctors. The dataset

[1] https://yourdoctors.online

includes a collection of anonymized doctor-patient chats between October 6, 2019, and July 15, 2020. We extracted 38,135 patient-doctor conversations, consisting of 901,939 messages exchanged between them. Note that the in-depth data exploratory analysis, e.g., n-gram analysis, is excluded due to information sensitivity.

Each chat between a patient and a doctor has two characteristics: the number of messages and the number of turns, i.e., the back and forth messages between them. The violin plot in Figure 1 shows the distribution of the number of turns and messages per chat. The number of turns in each chat has a distribution with a mean of 15.5 and a standard deviation of 11.5. On the other hand, each chat includes on average 23.8 messages with a standard deviation of 19.3.

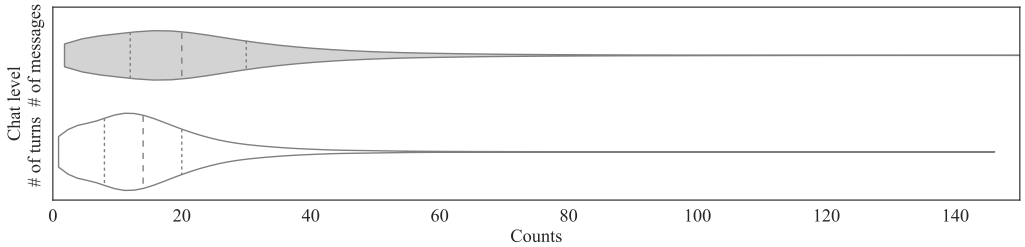


Figure 1: The distribution of the number of turns and messages per chat.

In the next step, we divide the chat into pairs of patient-doctor messages. Each paired message is manually labeled as "feasible" or "infeasible", indicating whether a paired message should trigger a smart reply or not. Figure 2 shows the distribution of feasible and infeasible patient-doctor messages' lengths. We note that the average length of infeasible messages is 20.3 (and $\sigma = 46.3$), whereas the average length of feasible messages is 11.2 (and $\sigma = 7.9$), indicating a significantly higher length for the infeasible ones.

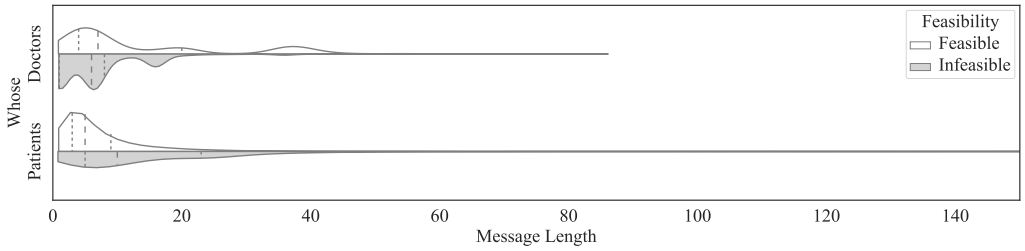


Figure 2: The distribution of the number of words in each message

3.1.1. Data cleaning

We define the following filtering conditions to maintain high-quality messages.

- We remove any patient/doctor message that is longer than 200 words. That is, we choose not to trigger any responses for those long messages.
- As we face a chat environment, there is a plethora of idioms, abbreviations, and mispronounced vocabularies. Therefore, we create a dictionary of abbreviations and replace each abbreviated word with its long-form, e.g., "by the way" as a substitute for "BTW" or "do not know" in place of "dunno". Moreover, using the "pyspellchecker" package in Python, we generate a comprehensive dictionary of typos in the medical domain. This misspelling dictionary includes 30,295 words extracted from the chats and is used to clean the dataset. Nevertheless, not all the misspelled words are retrievable. We are unable to suggest the proper replacement for some typos that are not similar to any words in the package's corpus.
- We decide to keep stopwords in the dataset as the final response needs to be grammatically correct. Therefore, we examined our method

with and without stopwords and found that keeping them enhances the replies' quality. Similarly, we do not apply lemmatization because it is detrimental to syntactic comprehension.

- Other preprocessing steps include removing extra white spaces, deleting punctuation and URLs, converting all characters to lower case, and, finally, removing non-Unicode characters and line-breaks.

3.1.2. Creating canned response set

As the response generation task requires labeled data, and considering that pairing patient and doctor messages is a tedious task, we select a portion of the data that captures the most significant characteristic of the desired output. Hence, we divide the work into two folds. First, we explore the similarity between doctors' messages and cluster them, and second, we find the patient pair for each doctor message in only dense, frequent clusters.

After data cleaning, we pinpoint the most frequent responses by doctors. However, we cannot make it by solely exploring response occurrence since many responses deliver the same message. For instance, "you're welcome", "happy to help", "no problem" and "my pleasure" are different possible answers to the same patient message. Therefore, we create a semantic cluster of the responses and examine the total frequency of responses in each cluster. In other words, the model should only learn the messages most commonly sent to the patients. Figure 3 demonstrates the steps in the manual labeling process.

As shown in Figure 3, we convert each textual message to numeric vectors through the weighted average word embedding. TF-IDF value for each word generates its weight, and its word embedding is treated as its value.

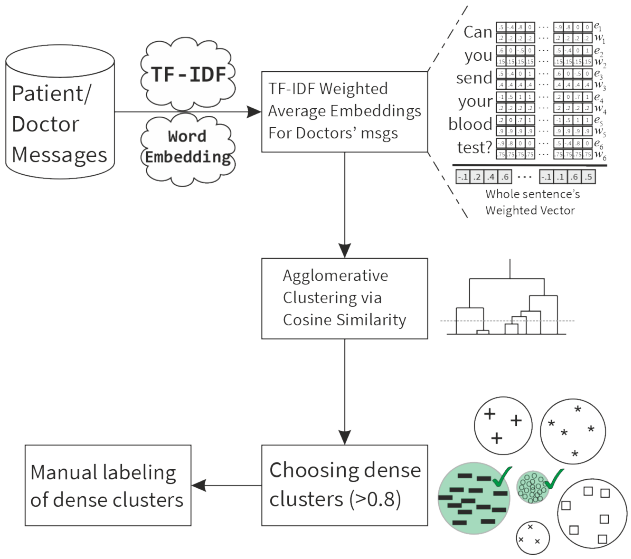


Figure 3: A flowchart for the manual labeling process

As there are many medical terms in messages exchanged, we use Wikipedia PubMed word embedding[2]. This word2vec dataset is induced on a combination of PubMed and PMC corpus, together with the texts extracted from an English Wikipedia dump. Therefore, it is suitable for both medical terms and daily language. We compared the performance of this word embedding with the Glove embedding [27] and found that many vocabularies that the Glove embedding does not support do exist in Wikipedia PubMed embedding. Moreover, the manual exploration of the generated responses points to a better performance of the Wikipedia PubMed embedding (i.e., it is improved by 3.4% in our case). After finding the proper embedding, we use the weighted average word embedding for doctor's messages and apply agglomerative clustering on the responses through the cosine similarity. Next,

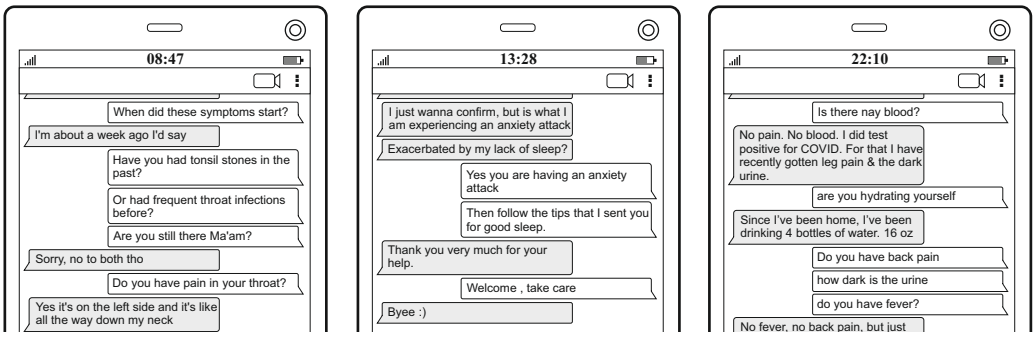
[2] https://bio.nlplab.org/

using average silhouette width, we found the optimal number of clusters as 158. Among them, we chose the clusters whose densities are more than 80%. Unlike dense clusters that include distinct message types, sparse ones contain a high volume of irrelevant messages with little or no similarity, and hence, we exclude the non-dense clusters.

3.1.3. Pairing responses and manual labeling

After obtaining the possible clusters for doctor messages, we pair each doctor message to its related patient message. During the manual labeling process, we encounter some challenges in the chat context. First, not all messages are a response to their previous message. For instance, a doctor may give some information regarding possible drugs without being asked to, or in some cases, a response is too generic or too specific and cannot be considered feasible. In such cases, instead of finding the paired patient message, we mark them as "infeasible". Second, message flow is not always in order. For instance, a patient may ask a question, and then in the following messages, give some additional information. Then, the doctor starts responding by asking something about the patient's first message. As the dataset does not include the "reply-to" option, we need to manually trace chats back to find a relevant patient message given each doctor's reply. It is a cumbersome task in the labeling process, which does not exist in previous works. Figure 4a and Figure 4b show examples of disorderly and correct flow, respectively. In some cases, a doctor's response may be relevant to something asked much earlier. For instance, in Figure 4c, the question of "how dark is the urine" is related to a message sent earlier. Also, the doctor's response is related to only parts of the text sent by the patient, and not all of it. In such cases,

we check only the last message's by a patient and trim the irrelevant part of a text. Wherever we are unable to find the desired match, we label it as "infeasible". Note that the entire manual labeling process is repeated by multiple experts to ensure reaching a consistent canned response set.



(a) Example 1 - disorderly flow (b) Example 2 - correct flow (c) Example 3 - disorderly flow

Figure 4: Examples of patient-doctor chats; the gray boxes indicate patient messages, and the white ones belong to doctors.

Ultimately, the manual labeling process leads to a set of paired patient-doctor chats and some infeasible cases. In total, we obtain 31,407 paired messages, 23.1% of which are "infeasible".

3.2. Response diversification

After finding the appropriate responses and associating them with patient messages, we consider strategies for diversifying the generated responses. Based on the comprehensive rules that we identified, we generate a set of diverse canned messages. Note that determining such rules for response diversification requires domain expertise and interaction with the stakeholders (e.g., physicians and end-users). Table 1 shows an example of our rule-based response diversification. We diversify the response "You are welcome" based

on some predefined rules. For instance, if the patient message implies the end of the conversation, we use "You are welcome. Take care. Bye." instead. As we consider a platform that suggests the top-3 responses to the doctors, our algorithm can benefit considerably from a more diverse set, including all possible situations. Otherwise, many irrelevant messages might pop-up on the platform, all pointing to the semantically identical response.

Table 1: An example of rule-based response diversification for the message "You are welcome."

Diversified response	**Adopted rules for the response diversification**
You are welcome.	A general answer to thanks, thank you, etc.
You are welcome. Take care. Bye.	Answer to thanks at the end of conversations. The patient message should imply the end of the chat.
You are welcome. Have a great day.	When a patient ends the chat by wishing a nice day.
You are welcome. Have a great night.	When a patient ends the chat by wishing a good night.
Take care, Happy to help! If you liked our service, please leave us a Google review :)	When a patient ends the chat implying a satisfactory service.

3.3. Smart response generation approach

In real-time chat conversations, it is typically not required to generate a response for all the received messages, which is unlike a chat-bot. Therefore, after preprocessing the messages, we define a triggering model that decides whether or not to trigger a reply for a given patient message. Triggering is a binary classification task based on the "feasible" / "infeasible" manual labeling explained in Section 3.1.3. If a patient message passes the triggering model with a prediction probability greater than a predetermined value p, then it enters the smart response generator phase; otherwise, we do not generate a reply for it.

Figure 5 illustrates the processes of triggering and response generation. The reply suggestion phase integrates different models to generate a proper

suggested response. Since a typical usage in practice involves recommending top-k responses (e.g., $k = 3$), our main aim is to propose the most appropriate response within the first k suggestions.

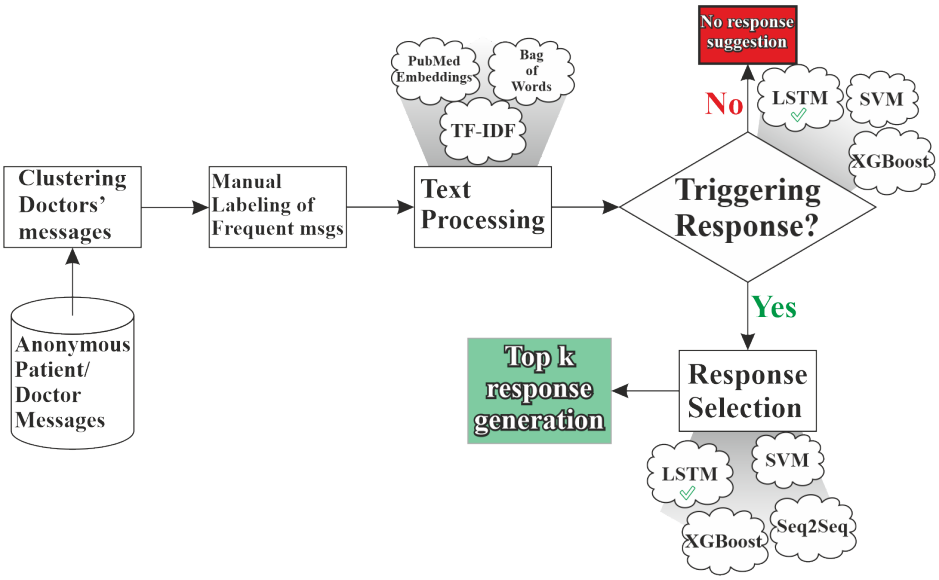


Figure 5: Triggering filter and response generation

3.4. Machine learning models for triggering and response generation

In the triggering phase, we aim to find the feasibility of the response generation. If a patient's message is too specific, (i.e., not applicable to other people), too generic (e.g., "OK" or "done"), or not seen in the training set (i.e., the chance of irrelevant suggestion is high), then there is no need to trigger any smart response. Furthermore, the triggering model ignores messages that are too complex or lengthy. On the other hand, the system should facilitate a doctor's job since they might be busy with multiple chats. The triggering should pass a message to response generation only if a proper response suggestion is likely. We experiment with different binary classification

methods to identify the most suitable model for the triggering phase. Accordingly, the value 0 for the dependent binary variable represents patient messages for which it is not ideal to generate a reply, and the value 1 indicates feasible patient messages. We use the preprocessed patient messages along with their length as the independent variables. The textual feature is converted to numeric values in different ways for each algorithm; therefore, we discuss the data conversion process within each model's explanation.

Unlike the triggering phase that deals with a binary output, the response generation decides on the proper reply only if it passes the first phase. Hence, the diversity of responses is higher, reducing the accuracy of the multi-class classification task. In this phase, we used all possible replies that are manually labeled as the dependent feature. However, we only used patients' messages as the dependent variables. We did not find any significant correlation between the length of a message and the generated response; therefore, we did not include patients' message length as a feature.

Although there are many machine learning (ML) algorithms for text classification, we chose to experiment with those commonly used in different domains. Moreover, in our preliminary analysis, we experimented with other ML methods (e.g., Random Forest and Naive Bayes); however, we did not find those to outperform the methods we summarized below.

XGBoost enhanced with weighted embedding. XGBoost, as a scalable tree boosting system [28], builds an ensemble of weak trees by incrementally adding new instances that contribute the most to the learning objectives. To accommodate the distributed text representation in numeric format, we average the embedding of each word per message as proposed by Stein et al.

[29] with some modifications. First, as we deal with medical conversations, we use Wikipedia PubMed as our word embedding representation. Second, since simple averaging does not reflect the importance of each word, we use a weighted average where TF-IDF values of the words are the weights. By these slight adaptations, we ensure that unimportant words do not have an impact on the averaged output for a given message [30]. Finally, we append the length of the patient message as a new independent feature. Hence, the text representation along with its length contributes to 201 independent attributes for each patient message. For XGBoost, while we include the message length in the triggering phase, we exclude it in the response generation phase. It is also important to note that we compared this approach with both simple TF-IDF representation [31] and unweighted word embedding average and found it to perform better.

SVM enhanced with weighted embedding. Support Vector Machine (SVM) has been widely used for text categorization and classification in different domains [32, 33, 34]. It identifies support vectors – i.e., data points closer to the hyperplane – to position a hyperplane that maximizes the classifier's margin. SVM learns independently the dimensionality of the feature space, which eliminates the need for feature selection. It typically performs well for text classification tasks with less computational effort and hyperparameter tuning while also being less prone to overfitting [35]. Hence, we consider SVM as a baseline and compare it with other classification approaches.

Bi-directional LSTM enhanced with Wikipedia PubMed embedding. Long short-term memory (LSTM) units, as the name suggests, capture both the long-term and the short-term information through the input, forget, and output

gate. Therefore, it has the ability to forget uncorrelated information while passing the relevant ones [36]. Since the patient messages consist of long sentences, such gates are ideal to have the least information loss. They can detect message contents stored as the long-term memory inside the cell while keeping invaluable information provided towards the end of a sentence. In our algorithm, we use Bi-directional LSTM (BiLSTM) units that learn information from both directions, enabling them to access both the preceding and succeeding contexts [37]. This way, equal weight is provided towards the beginning and the end of a sentence. BiLSTM units are an appropriate remedy for our problem since a patient message may contain useful information either at the beginning of a sentence or at the end. Using the attention mechanism, BiLSTM disregards generic comments and concentrates on more pertinent information [38].

Seq2Seq enhanced with Wikipedia PubMed embedding. Sequence-to-sequence models turn one sequence into another. It is used primarily in text translation, caption generation, and conversational models. Therefore, we only apply it to the reply suggestion phase as it is not generalizable to the triggering phase. Our Seq2Seq model consists of an encoder, decoder, and an attention layer [39]. The encoder encodes the complete information of a patient message into a context vector, which is then passed on to the decoder to produce an output sequence of a doctor's reply. Since our data consists of long sentences, we use an attention mechanism to assign more weight to relevant parts of the context vector to improve computational efficiency as well as accuracy [40].

To prepare our data for the Seq2Seq model, we tokenize both doctors'

and patients' messages and pad them to match the length of the longest sentence in our data. Start and end tokens are added to each sequence. Furthermore, we use a pre-trained Wikipedia PubMed embedding layer to capture the text semantics. The encoder additionally uses a Bi-directional LSTM layer for enhanced learning of encoded patient messages. We train it using the Adamax optimizer and sparse categorical cross-entropy to calculate the losses.

We employ beam search [41] to retrieve the predicted outcomes of the model using a beamwidth of three and apply length normalization to avoid biases against lengthier doctor replies. We rank replies according to their beam scores and choose the top-k responses. As the model generates responses word by word, there is a tendency for the model to suggest inappropriate or grammatically incorrect sentences. To overcome this issue, we apply cosine similarity to match the generated responses with our canned response set and select the ones with the highest cosine similarity score. Hence, we ensure that the proposed options will have proper word choice and grammar. Nevertheless, when the final top-k suggested replies overlap, we iteratively cycle through their cosine scores and pick the next best response until we reach k unique suggestions.

3.5. Experimental setup

We use 5-fold nested cross-validation and tune the most important parameters of the models by dividing the training dataset into validation and training sets. In each grid search procedure of the hyperparameters, we identify the best models for text classification. In the testing phase, we use the models that perform best on the validation set. We provide the final model

configurations as follows.

- We obtain the learning rate as 0.3 and the number of trees as 200 for XGBoost, whereas the other parameters are set to the default values of the XGBoost library in Python.
- Our SVM model uses a linear kernel with a degree of 3 and is implemented using the scikit-learn package in Python.
- We use TensorFlow to create our sequential LSTM model, which has an embedding layer powered by Wikipedia Pubmed. The model has a bidirectional layer of size 200, a dense layer of size 100 with the Relu activation, and a dense output layer with sigmoid and softmax activation functions for triggering and response generation, respectively. We train the model for 20 epochs using the Adam optimizer.
- For our Seq2Seq model, we initiate our encoder with the Wikipedia Pubmed embedding layer, followed by a bidirectional LSTM layer of size 1024. Next, we use an LSTM layer for our decoder, along with the Luong Attention Mechanism. We calculate sparse categorical cross-entropy loss, and the model is trained for 15 epochs using the Adamax optimizer.

4. Results

In this section, we first briefly define the performance metrics used to evaluate different algorithms. Then, we report the performance of the smart response suggestion together with sample generated responses.

To investigate the performance of two different phases of the algorithm, we relied on two different sets of metrics. For the triggering phase, we used

"accuracy", the ratio of correct predictions to the number of instances, "precision", how many instances predicted as class c belongs to the same class, "recall", how many data points that belong to class c are found correctly, and "F1-score", which is the harmonic mean of precision and recall. These four performance metrics are threshold-dependent (i.e., model predictions constitute a probability distribution over class labels, and binary predictions are determined based on a probability threshold, e.g., 0.5). Therefore, we also utilize the area under the ROC curve (AUC-ROC) as a threshold-independent approach to mitigate the problems with threshold settings [42].

We employ different metrics to assess the performance of response generation models. We mainly rely on the "precision@k" metric to report the accuracy of the suggestions [6]. If the suggested response is among the top k responses, we call it a correct suggestion; otherwise, it is not a suitable suggestion. We take the number of generated responses as $k = 3$. Therefore, if the model is adept enough to include the proper reply among the top 3, it will be considered an appropriate suggestion. We also report "precision@1" and "precision@5" to gain more insights regarding the models' performances. Another useful metric is the rank of the suggested response. If a model puts forward a reply in rank 4, there is a likelihood that it can be improved further by some parameter tuning. On the other hand, if the response is ranked 20, the model is unlikely to suggest a proper response. Consequently, we report the Mean Reciprocal Rank (MRR) metric, that is,

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i}$$

where N is the total number of messages. MRR ranges from 0 to 1, where 1

indicates the optimal performance (all the suggestions are ranked first).

4.1. Triggering performance

Accurate triggering is important since an infeasible patient's message passing this filter not only leads to an irrelevant message but also increases the computational complexity. Consequently, an inferior model will reduce the quality of the suggested replies and degrade the performance of the overall response generation mechanism.

Table 2 shows the performance of different models for the triggering phase. All the models outperform the baseline approach, which generates the response based on frequency. The reason for the relatively high accuracy of the frequency-based approach is the imbalanced ratio of feasible and infeasible cases. Therefore, by overestimating the majority group, it still can reach acceptable performance. However, when it comes to the threshold-independent metric, AUC-ROC, the frequency-based approach performs as poorly as a random guess with almost 50% AUC-ROC. On the other hand, other approaches show significant improvement over the frequency-based suggestion.

Table 2: Performances of different models for the triggering task (reported values are all in percentages)

Method	**Accuracy**	**Precision**		**Recall**		**F1-score**		**AUC-ROC**
		Infeasible	Feasible	Infeasible	Feasible	Infeasible	Feasible	
BiLSTM†	**87.4**	72.1	**92.1**	**73.7**	91.4	**72.9**	**91.8**	91.5
XGBoost$^{\dagger *}$	86.0	73.7	88.9	61.1	93.5	66.8	91.2	**92.4**
SVM$^{\dagger *}$	85.7	**74.8**	88.0	57.0	**94.3**	64.7	91.0	92.2
Frequency	64.7	23.2	77.0	23.0	77.2	23.1	77.1	50.0

†: Wikipedia-PubMed embedding; *:TF-IDF values as weight

Table 2 also shows that BiLSTM, XGBoost, and SVM perform similarly in predicting the majority class. All the models have acceptable values for precision, recall, and F1-score for the feasible cases. However, XGBoost and SVM perform relatively poorly in recalling the infeasible messages. They overfit the majority class, leading to passing so many messages to the next phase and, consequently, an abundance of irrelevant reply suggestions. BiLSTM shows the best aggregate performance over various performance metrics. Nonetheless, SVM and XGBoost demonstrate slight superiority in some metrics. It mainly stems from overestimating one class and neglecting the other one. Therefore, we choose BiLSTM as our primary triggering model. We also note that BiLSTM provides the best performance when combined with its response suggestion algorithm; however, we do not provide a detailed analysis with combined triggering and response generation algorithm for the sake of brevity.

4.2. Response suggestion performance

After a message successfully passes the triggering filter, it enters into the response suggestion model. Response suggestion aims to suggest proper messages within the top responses to facilitate the patient-doctor conversation. Here, we only concentrate on doctors' response generation processes.

We compare machine learning algorithms with the baseline, frequency-based suggestion. The baseline selects doctor responses from the canned messages based on their occurrence probability in the training set. As our response set includes certain frequent categories, the precision of the baseline might seem relatively high. However, the difference between machine learning algorithms and the baseline is statistically significant. Table 3 summarizes

the accuracy of each algorithm. BiLSTM enhanced by Wikipedia-PubMed embedding significantly outperforms other machine learning and deep learning algorithms. It suggests more correct responses in the first rank (precision@1) than other alternative approaches. In 61.95% of the suggestions, the proper reply does exist in its first suggestion, and in 87.73% of the times, the model can generate a response desired by a doctor. In a software application that suggests top-3 responses, the model accuracy based on precision@3 is promising. Table 3 also reports the Mean Reciprocal Rank of the models. According to the MRR values, BiLSTM significantly outperforms others. It is important to note that the model is able to generate an instantaneous response, i.e., less than a second, which is desired by the application. Therefore, the proposed model can suggest the proper reply options in a timely manner.

Table 3: Accuracy of the suggested responses for different models

Method	**precision@1 (%)**	**precision@3 (%)**	**precision@5 (%)**	**MRR**
BiLSTM†	**58.98**	**83.28**	**85.37**	**0.75**
Seq2Seq†	53.21	61.48	68.63	0.60
XGBoost$^{\dagger *}$	51.33	79.59	82.83	0.69
SVM$^{\dagger *}$	47.62	78.97	82.25	0.68
Frequency	16.71	32.20	42.80	0.35

†: Wikipedia-PubMed embedding; *:TF-IDF

We note that in a previous work by Kannan et al. [6] for Gmail, thousands of possible responses were available, and they reached the precision@10 value of 48.3%. This seemingly lower performance can be attributed to the high number of messages existing in their canned response set. Nevertheless,

in their work, even the frequency-based method was able to fulfill the precision@10 equal to 32.1%, implying that the number of frequent responses was dominant. In another work for in-app communication of drivers by the Uber team [7], a slightly different metric for accuracy is used. They found 77.2% accuracy in intent detection of the generated replies. However, our work mainly concentrates on the precision of the suggested responses and not just their intentions. In comparison to these previous studies, we can conclude that our adopted algorithm for response generation in online medical services is successful and surpasses the baseline (i.e., frequency-based response generation) by enhancing its precision more than 2.5 times (see Table 3).

We demonstrate the distribution of the actual frequency of the most frequent medical responses (i.e., excluding casual responses such as "You are welcome." and "Thanks.") in Figure 6. The ground-truth frequency is shown in black, while the predicted frequency is grey. We observe that both the prediction and the ground truth follow a similar distribution. For the casual responses, which are excluded from the graph, the generated frequencies exhibit a similar pattern as the actual ones.

Figure 7 shows the accuracy of suggested responses given their rank. If a message passes the triggering phase, the model will recommend a reply whether the original message is feasible or not. Therefore, the plot shows the ratio of precise suggestions per rank for all feasible and infeasible cases entering the response generator. BiLSTM has the best overall performance for the top-3 responses. It prompts the highest ratio of correct replies in the first rank while having the least suggestions in the fourth position and above. Surprisingly, seq2seq, which is the second-best algorithm consider-

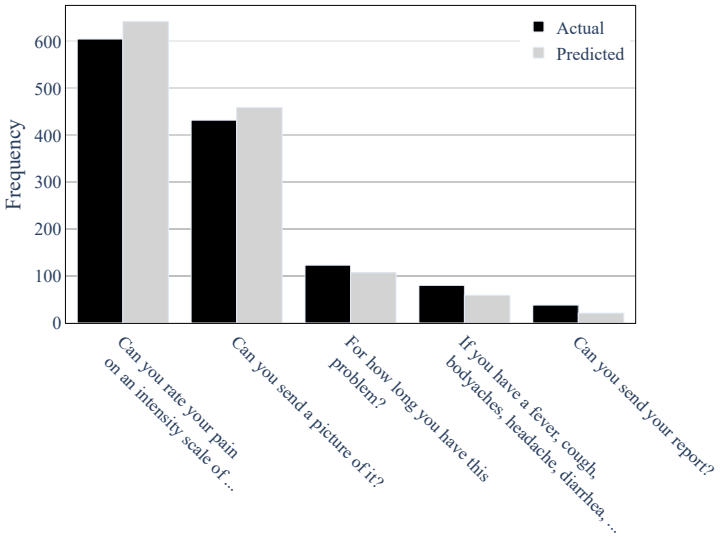


Figure 6: The distribution of the actual frequency of the doctor messages compared to the predicted responses by BiLSTM

ing the suggested messages ranked first, loses its superiority shortly after. One reason for the performance drop is the beam search associated with the response selection. It does not have the option to diversify the message, and adding rule-based diversification increases its computation time. Therefore, the model fails to generate high-quality responses considering the pace needed to output a reply. All the analyses highlight the advantage of using BiLSTM in automated doctor response recommendations.

4.3. Sample generated responses

Figure 8 demonstrates sample replies generated by the BiLSTM model. If a patient's message passes the triggering phase, the algorithm suggests top-3 possible responses in order. In Figure 8a, the proper response is ranked first. The model learned from the training set to select an automated message from the canned set to apologize for delays. As these kinds of less frequent

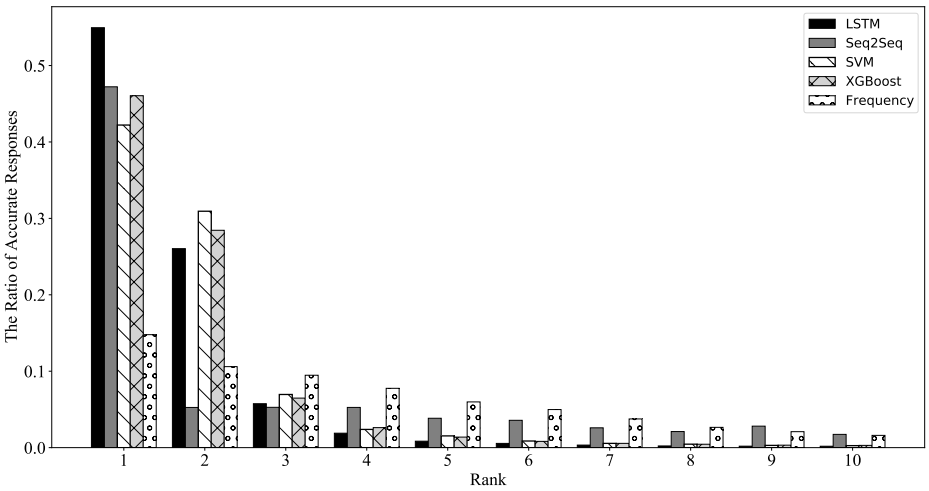
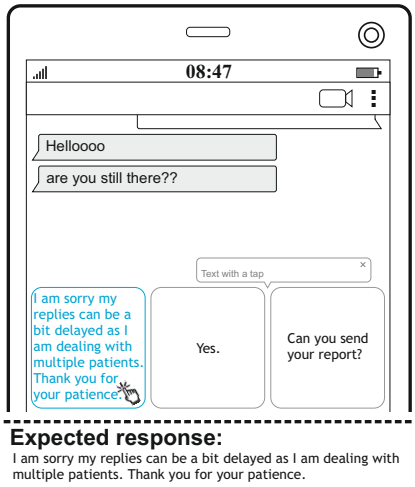


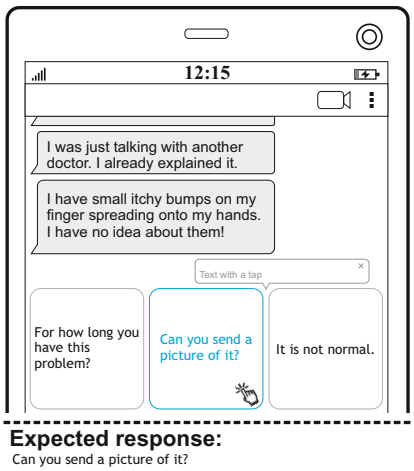
Figure 7: The distribution of correct responses ranked top-10 for different methods

messages are not diversified, and we do not have other alternatives for them, the third suggested response might seem less relevant. Figure 8b depicts a situation in which the expected response is ranked second. However, the other two responses are pertinent to the patient's message. As the patient asks about itchy bumps, the doctor may either request a picture to have a better idea or ask about its longevity. In Figure 8c, the patient implies Covid-19 condition by referring to "virus", "testing", and "positive", and the suggested response inform the patient about the symptoms of Covid-19. The other two responses show compassion for the patient and ask for a report if there is any clinical test available. Although the response does not appear at the first rank, others still provide an appropriate alternative for the ground truth. Lastly, Figure 8d shows a case in which the proper response is not provided. The patient asks for the place where he/she can find weight gain supplements, and the model cannot suggest the location. We manually checked for the rank of the correct reply and found it at rank six.

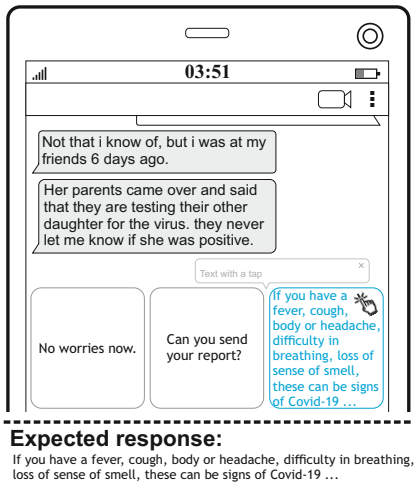
Nonetheless, there is a typo in the patient's message that can be addressed, and the wrong responses can be potentially avoided.



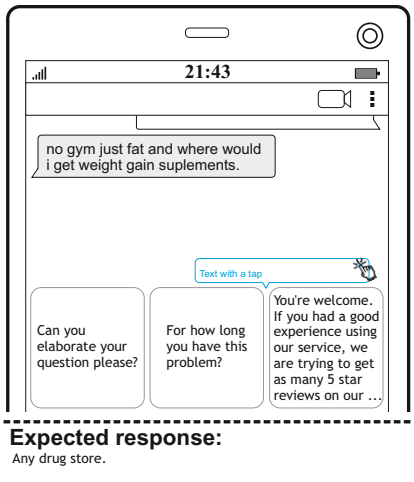
(a) Example 1: A response appears at the first position as expected.



(b) Example 2: The expected response is still among the top-3 suggestions.



(c) Example 3: The expected reply is ranked third. Still easy to be tapped.

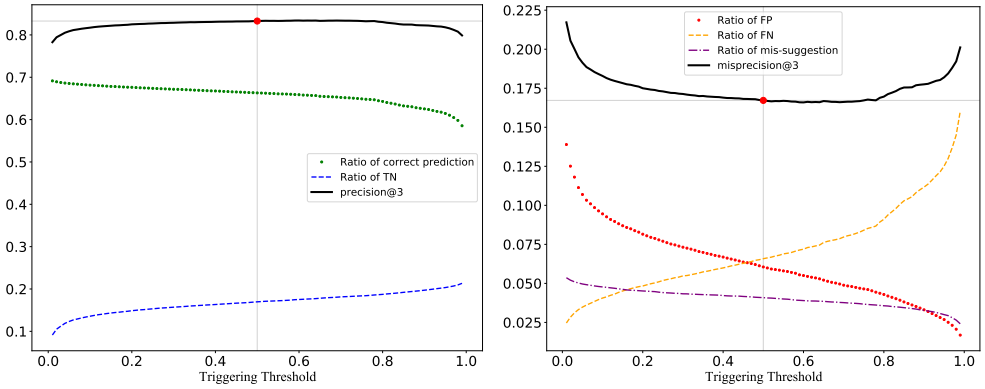


(d) Example 4: A wrong suggestion. The model does not suggest the expected reply.

Figure 8: Examples of smart responses generated by BiLSTM

4.4. Sensitivity of the model to the triggering threshold

One of the most significant parameters of the algorithm is the triggering threshold. As the triggering model suggests the probability of generating a response, it is important to determine how to convert that probability to a binary decision. As a rule of thumb, we round numbers greater than or equal to 0.5 to 1 and smaller ones to 0. However, the question is whether the threshold of 0.5 provides the best-suggested replies. When the threshold is too small, the model tends to generate responses for most infeasible cases; on the other hand, when it is close to 1, the model becomes more conservative as it avoids generating inappropriate responses.



(a) Correct predictions in detail

(b) Wrong predictions in detail

Figure 9: The sensitivity of the algorithm to the threshold of the triggering phase

Figure 9 demonstrate the sensitivity of the performance of the BiLSTM model to its triggering threshold. Figure 9a demonstrates the source of precision. It can come either from the correct filtering of infeasible responses (TN) or the correct suggested replies that fall in the top-3 suggestions. Therefore, the total precision@3, the sum of the other two predictions, is shown in black. We observe that the algorithm is robust to the threshold parameter as

there is only a negligible fluctuation in the total precision for different values of the triggering threshold. In Figure 9b, we explore the effect of triggering threshold on the ratio of mispredictions. We divide the misprecision@3 into three folds: the incorrect predictions originating from passing infeasible messages (FP), the ratio of incorrectly filtered feasible messages (FN), and missuggestions of feasible responses, meaning a suggested response is not among the top 3. Same as the previous plot, we depict the sum of these three subdivisions in black. The misprecision@3 has the least effect on the ratio of missuggestions; however, it highly affects both False Positive and False Negative cases. Altogether, misprecision@3 does not oscillate within the range of 0.3 and 0.8. Hence, the method seems robust under different threshold values. Accordingly, we choose the common threshold of 0.5 for all the experiments.

5. Concluding remarks

Considering the rapid growth of online medical chat services, telehealth companies may choose to either expand their capacity — e.g., the number of physicians — or facilitate the communication for the existing employees to maximize their utilization. Instead of employing an expensive workforce, the cost of enhancing the current application with Artificial Intelligence is almost negligible. Accordingly, smart response suggestions can relieve the doctors' burden by facilitating patient-doctor communication, proposing appropriate replies, and saving their valuable time. To the best of our knowledge, we investigate the feasibility of having smart reply suggestions in medical contexts for the first time.

We use the actual conversations between patients and doctors coming from an online medical chat service. Accordingly, after exploratory data analysis, we clean the dataset by devising a canned response set. Using clustering techniques, we find the densest clusters of doctors' messages and extract frequent responses from those. Afterward, we match the patient and doctor messages being aware of the complexity of disorderly exchanged chats, which results in 31,407 paired messages. Not all patient messages require smart replies; therefore, we also label the pairs as "feasible" or "infeasible". Our algorithm proceeds in two steps: predicting whether we need to trigger a smart reply and suggesting the proper response given a message passes the triggering phase. We explore different combinations of machine learning and deep learning algorithms to address each step. Furthermore, we tune the parameter and report the performance using 5-fold nested cross-validation. We assess each algorithm's performance using threshold-dependent and -independent metrics and observe that Bidirectional LSTM is the best method for the triggering phase. It has a balanced score for both majority and minority class labels, i.e., feasible and infeasible cases. In addition, its suggested replies are also the most appropriate in the response generation phase. Moreover, we tested its robustness to the triggering threshold and found it to be resilient to its parameter changes.

A relevant venue for future research would be to improve the method by including more data points (i.e., more labeled-conversations). To the best of our knowledge, there is no publicly available dataset for medical conversations. Therefore, we only apply the algorithm to our proprietary dataset. Besides, in response to the COVID-19 pandemic, our dataset is continuously

being updated. Specifically, we find constant changes in patient queries and doctor answers. For instance, with regards to the modeling symptoms' questions, we observe that the vaccine queries become dominant. Accordingly, an automated mechanism to retrain the models according to unprecedented challenges can be developed. We note that the overall response generation mechanism becomes feasible by introducing enough paired messages and updating the model weights. Moreover, as manual labeling is a tedious task, we plan to investigate semi-supervised learning for semantic clustering and labeling big datasets.

6. Acknowledgement

The authors would like to thank Your Doctors Online for funding and supporting this research. This work was also funded and supported by Mitacs through the Mitacs Accelerate Program. The authors would also like to thank Gagandip Chane for his help with the data labeling.

References

[1] G. Charlton, Consumers prefer live chat for customer service: stats, 2013. URL: https://econsultancy.com/consumers-prefer-live-chat-for-customer-service-stats/.

[2] AAMC, Physician supply and demand. a 15-year outlook: Key findings, 2019. URL: https://www.aamc.org/media/45976/download.

[3] M. Hawkins, Survey of physician appointment wait times and medicare and medicaid acceptance rates, 2017. URL: https://www.aristamd.com/wp-content/uploads/2018/11/mha2017waittimesurveyPDF-1.pdf.

[4] A. Mehrotra, M. Chernew, D. Linetsky, H. Hatch, D. Cutler, The impact of the covid-19 pandemic on outpatient visits: A rebound emerges, 2020. URL: https://www.commonwealthfund.org/publications/2020/apr/impact-covid-19-outpatient-visits.

[5] H.-A. B. Epstein, Texting thumb, Journal of Hospital Librarianship 20 (2020) 82–86. doi:10.1080/15323269.2020.1702846.

[6] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, M. Ganea, P. Young, et al., Smart reply: Automated response suggestion for email, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 955–964.

[7] Y. Weng, H. Zheng, F. Bell, G. Tur, Occ: A smart reply system for efficient in-app communications, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2596–2603.

[8] L. Galke, G. Gerstenkorn, A. Scherp, A case study of closed-domain response suggestion with limited training data, in: International Conference on Database and Expert Systems Applications, Springer, 2018, pp. 218–229.

[9] L. Zhou, J. Gao, D. Li, H.-Y. Shum, The design and implementation of xiaoice, an empathetic social chatbot, Computational Linguistics 46 (2020) 53–93. doi:10.1162/coli_a_00368.

[10] R. Yan, " chitty-chitty-chat bot": Deep learning for conversational ai., in: IJCAI, volume 18, 2018, pp. 5520–5526.

[11] R. Yan, D. Zhao, W. E, Joint learning of response ranking and next utterance suggestion in human-computer conversation system, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 685–694.

[12] R. Yan, D. Zhao, Coupled context modeling for deep chit-chat: towards conversations between human and computer, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2574–2583.

[13] R. Li, J.-Y. Jiang, C. J.-T. Ju, C. Flynn, W.-l. Hsu, J. Wang, W. Wang, T. Xu, Enhancing response generation using chat flow identification (2019).

[14] J.-G. Kim, C.-W. Wu, A. Chiang, J. Ko, S.-J. Lee, A picture is worth a thousand words: Improving mobile messaging with real-time autonomous image suggestion, in: Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications, HotMobile '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 51–56. doi:10.1145/2873587.2873602.

[15] M. Jain, P. Kumar, R. Kota, S. N. Patel, Evaluating and informing the design of chatbots, in: Proceedings of the 2018 Designing Interactive Systems Conference, 2018, pp. 895–906.

[16] S.-C. Lee, J. Song, E.-Y. Ko, S. Park, J. Kim, J. Kim, Solutionchat: Real-time moderator support for chat-based structured discussion, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1–12. doi:10.1145/3313831.3376609.

[17] P. Hamet, J. Tremblay, Artificial intelligence in medicine, Metabolism 69 (2017) S36–S40.

[18] E. J. Topol, High-performance medicine: the convergence of human and artificial intelligence, Nature medicine 25 (2019) 44–56.

[19] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, K. Zhang, The practical implementation of artificial intelligence technologies in medicine, Nature medicine 25 (2019) 30–36.

[20] L. Tudor Car, D. A. Dhinagaran, B. M. Kyaw, T. Kowatsch, S. Joty, Y.-L. Theng, R. Atun, Conversational agents in health care: Scoping review and conceptual analysis, J Med Internet Res 22 (2020) e17158. doi:10.2196/17158.

[21] K.-J. Oh, D. Lee, B. Ko, H.-J. Choi, A chatbot for psychiatric counseling in mental healthcare service based on emotional dialogue analysis and sentence generation, in: 2017 18th IEEE International Conference on Mobile Data Management (MDM), IEEE, 2017, pp. 371–375.

[22] T. Kowatsch, M. Nißen, C.-H. I. Shih, D. Rüegger, D. Volland, A. Filler, F. Künzler, F. Barata, D. Büchter, B. Brogle, K. Heldt, P. Gindrat, N. Farpour-Lambert, D. l'Allemand, Text-based healthcare chatbots supporting patient and health professional teams: Preliminary results of a randomized controlled trial on childhood obesity, in: Persuasive Embodied Agents for Behavior Change (PEACH2017) Workshop, co-located with the 17th International Conference on Intelligent Virtual Agents (IVA 2017), 2017.

[23] C. Cuffy, N. Hagiwara, S. Vrana, B. T. McInnes, Measuring the quality of patient–physician communication, Journal of Biomedical Informatics 112 (2020) 103589. doi:https://doi.org/10.1016/j.jbi.2020.103589.

[24] T. Davenport, R. Kalakota, The potential for artificial intelligence in healthcare, Future healthcare journal 6 (2019) 94.

[25] J. T. Hancock, M. Naaman, K. Levy, AI-Mediated Communication: Definition, Research Agenda, and Ethical Considerations, Journal of Computer-Mediated Communication 25 (2020) 89–100. doi:10.1093/jcmc/zmz022.

[26] T. Nadarzynski, O. Miles, A. Cowie, D. Ridge, Acceptability of artificial intelligence (ai)-led chatbot services in healthcare: A mixed-methods study, Digital health 5 (2019) 2055207619871808.

[27] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[28] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794. doi:10.1145/2939672.2939785.

[29] R. A. Stein, P. A. Jaques, J. F. Valiati, An analysis of hierarchical text classification using word embeddings, Information Sciences 471 (2019) 216 – 232. doi:https://doi.org/10.1016/j.ins.2018.09.001.

[30] J. Zhao, M. Lan, J. F. Tian, ECNU: Using traditional similarity measurements and word embedding for semantic textual similarity estimation, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado, 2015, pp. 117–122. doi:10.18653/v1/S15-2021.

[31] Z. Qi, The text classification of theft crime based on tf-idf and xgboost model, in: 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2020, pp. 1241–1246. doi:10.1109/ICAICA50127.2020.9182555.

[32] J. Hartmann, J. Huppertz, C. Schamp, M. Heitmann, Comparing automated text classification methods, International Journal of Research in Marketing 36 (2019) 20–38. doi:https://doi.org/10.1016/j.ijresmar.2018.09.009.

[33] S. Günal, Hybrid feature selection for text classification, 2011.

[34] F. Ren, M. G. Sohrab, Class-indexing-based term weighting for automatic text classification, Information Sciences 236 (2013) 109–125. doi:https://doi.org/10.1016/j.ins.2013.02.029.

[35] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: C. Nédellec, C. Rouveirol (Eds.), Machine Learning: ECML-98, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 137–142.

[36] J. Du, C. M. Vong, C. L. P. Chen, Novel efficient rnn and lstm-like architectures: Recurrent and gated broad learning systems and their applications for text classification, IEEE Transactions on Cybernetics (2020) 1–12. doi:10.1109/TCYB.2020.2969705.

[37] G. Liu, J. Guo, Bidirectional lstm with attention mechanism and convolutional layer for text classification, Neurocomputing 337 (2019) 325–338. doi:https://doi.org/10.1016/j.neucom.2019.01.078.

[38] D. S. Sachan, M. Zaheer, R. Salakhutdinov, Revisiting lstm networks for semi-supervised text classification via mixed objective function, Proceedings of the AAAI Conference on Artificial Intelligence 33 (2019) 6940–6948. doi:10.1609/aaai.v33i01.33016940.

[39] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 27, Curran Associates, Inc., 2014, pp. 3104–3112.

[40] T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1412–1421. doi:10.18653/v1/D15-1166.

[41] A. Kumar, S. Vembu, A. K. Menon, C. Elkan, Beam search algorithms for multilabel learning, Machine learning 92 (2013) 65–89.

[42] J. Hernández-Orallo, P. Flach, C. Ferri, A unified view of performance metrics: Translating threshold choice into expected classification loss, Journal of Machine Learning Research 13 (2012) 2813–2869.