Isa: Intuit Smart Agent, A Neural-Based Agent-Assist Chatbot

Zijun Xue[1], Ting-Yu Ko, Neo Yuchen, Ming-Kuang Daniel Wu, and Chu-Cheng Hsieh[2]

***Abstract*— Hiring seasonal workers in call centers to provide customer service is a common practice in B2C companies. The quality of service delivered by both contracting and employee customer service agents depends heavily on the domain knowledge available to them. When observing the internal group messaging channels used by agents, we found that similar questions are often asked repetitively by different agents, especially from less experienced ones. The goal of our work is to leverage the promising advances in conversational AI to provide a chatbot-like mechanism for assisting agents in promptly resolving a customer's issue. In this paper, we develop a neural-based conversational solution that employs BiLSTM with attention mechanism and demonstrate how our system boosts the effectiveness of customer support agents. In addition, we discuss the design principles and the necessary considerations for our system. We then demonstrate how our system, named *Isa* (Intuit Smart Agent), can help customer service agents provide a high-quality customer experience by reducing customer wait time and by applying the knowledge accumulated from customer interactions in future applications.**

I. Introduction

One of the primary factors contributing to business success is exceptional customer service[1]. Research[2] has shown that B2C companies such as Amazon and Zappos have invested many resources to improve the quality of customer service and achieve business success. Similar to these B2C companies, Intuit (`http://www.intuit.com`) hires thousands of seasonal workers in our call centers across the world to provide quality customer support, especially during peak seasons.

A large volume of service calls floods into our call centers every day. To address an almost endless variety of customer requests and needs, our customer service agents utilize an internal communication tool, Slack (`http://www.slack.com`), to communicate and discuss customer problems when speaking with customers. Specifically, they rely on Slack *channels*, a mechanism of sending group messages, to ask for help from other senior agents and employees. Front-line agents and seasonal workers alike rely heavily on these Slack channels to consult senior agents and to escalate issues to managers.

Based on our study, agents often ask identical or similar questions repetitively. For example: "What does this Error code: 41424 (connection timeout) mean?" and "I cannot delete the very last credit card on file. Please help." In addition, many frequently-asked questions are actual system error or warning messages (such as an internal error code) directly copied and pasted into the channel by an agent. While a well-designed agent onboarding and training process helps reduce occurrences of basic questions, memorizing answers to all frequently asked questions is not a practical solution.

In this study, we propose a chatbot-like experience for answering repetitive questions. The advances in neural-based technology have fueled the attempts to construct powerful conversational AI. We believe that a task-oriented chatbot can best meet our need for automatically answering repetitive questions. The imminent benefits brought about by our system are manifold. On the one hand, our system reduces a significant amount of workload for senior customer services agents who assist seasonal workers; on the other, all knowledge learned over time is accumulated in the AI system to enable many future research and applications.

Our goal is to build a task-specific chatbot for internal use to assist agents in solving customer's issues promptly. Our chatbot system differs from a general purpose chatbot in two ways. First of all, for a question with a known answer, the system should provide the most probable answer previously hand-crafted by a domain expert such as a senior service agent, aiming at quick task completion instead of encouraging conversations. That is to say, "friendliness" and "human-like" characteristics are less relevant. Secondly, we would like the system to gauge its own confidence in answering a given question correctly. For a question with a low confidence answer or without an answer, our system should *hand over* the question to human experts or simply remain silent.

We summarize our contributions as follows:

- We develop a neural-based conversational system that employs BiLSTM with attention mechanism and demonstrate how it boosts the effectiveness of customer support agents.
- We research principles and propose considerations for developing agent-assist chatbot applications.

II. Platform Overview

A. Slack

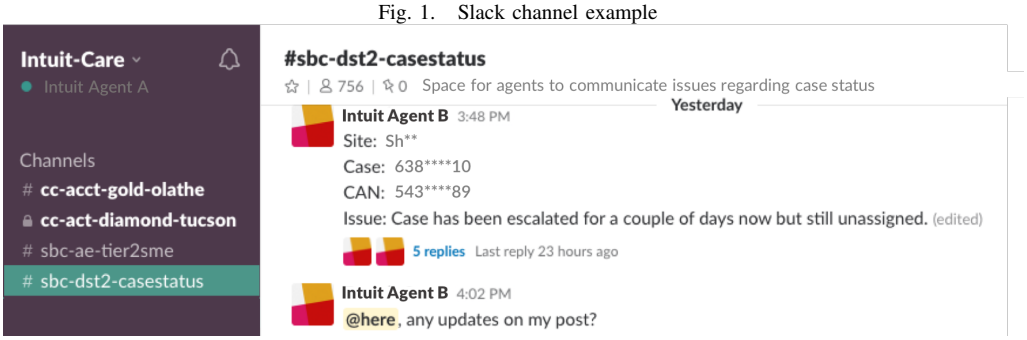
Slack is one of the mainstream organizational communication platforms comprising a set of cloud-based team collaboration tools and services. Launched in 2014, Slack claims a customer base of 8 million daily active users from more than

[1]Zijun Xue is a PhD candidate of Computer Science at University of California, Los Angeles, CA 90095 `xuezijun@cs.ucla.edu`

*This work is supported by Intuit during Zijun's internship employment with Intuit from Jun. 2018 to Sep. 2018.

[2]Ting-Yu Ko, Neo Yuchen, Ming-Kuang Daniel Wu and Chu-Cheng Hsieh are with the Customer Success Team of Central Data Science Organization, Intuit Inc., 2700 Coast Ave, Mountain View, CA 94043 `jessica_ko, daniel_wu, neo_yuchen, chu-cheng_hsieh@intuit.com`

Fig. 1. Slack channel example



70,000+ paying companies and 500,000 organizations [3]. One of the major features of Slack is messaging which offers both channel-wide and direct messaging. Organizations can set up different channels to segregate messages, discussions and notifications by purpose, department or topic. Direct messages are like traditional instant messaging intended for a person instead of a group. Intuit adopts Slack as its internal communication platform.

B. Slack Bot User

Slack Bot User is Slack's built-in feature that can be programmatically controlled through Slack's APIs to interact with users of a workspace in a conversational manner. Developers use the *Events API* as the primary way for building applications that receive and respond to events. Examples of such applications include monitoring and processing channel activities, posting messages and responding to users, and creating interactive messages by adding UI widgets such as buttons. A *Slack Bot User* application can be installed for different channels and deployed to different Slack workspaces, which enables us to create a Slackbot that serves multiple customer service Slack channels and workspaces[4].

III. Dataset Overview

A. Intuit Slack Channel Chatlog

Raw Slack channel data can be obtained through Slack's API. Specifically, one can use `channel.history` method to fetch message and event history of a channel. Each channel history log, all of which can be exported in JSON format, contains meta information such as main channel post type, timestamp, and user in addition to the message text. If there exist replies to a main channel post, a field `replies` is added to list associated user and timestamp for all replies to that original channel post. Here, the timestamp is a unique key that can be used to join the message text with meta information for a reply message. For instance, the threaded response object has a field `thread_ts` which is the timestamp of the original channel post associated with this particular reply post. The first dataset we use is the chatlog from the Slack channel `"sbc-ae-tier2sme"`, a channel for agents to discuss user account issues. This dataset contains 20,764 posts generated between April 2017 and May 2018. An initiating question is posted following a predefined convention where "Site", "Case" (case number), "CAN" (customer account number) and "Issue" are specified, as shown in Figure 1.

B. Customer Support Knowledge Base

Intuit has licensed customer support *knowledge base (KB)* to power customer service operation. Subject domain experts such as experienced customer service agents and technical writers contribute the knowledge articles in the *KB*. These articles contain information for processes, historical issues and solutions, FAQs and tutorials. Customer service agents regularly use the *KB* to search for information to resolve customers' issues. At the time of writing, there are more than 20,000 *KB* articles covering about 40 products. Each article's metadata containing information such as topics and view counts also serve as a valuable data source.

C. Exploratory Analysis

Across Intuit's Customer Care organization, there are more than 500 customer service agent Slack channels. About 60 questions are asked daily in a typical channel, which amounts to nearly 20,000 questions daily. This huge volume of questions demand a significant amount of time and efforts from expert agents, therefore present an opportunity for optimization.

Table I shows data collected from 6 agent Slack channels in `Intuit-Care` workspace between April 2017 and June 2018. It is worth noting that usage patterns vary across different Slack channels, and *general agents* outnumber *expert agents* [1]. Based on the number of questions that were answered by the top 5 expert agents (those who have answered the most questions), we see that at least 65% of the questions are in fact answered by these top 5 expert agents.

Take channel `#sbc-ae-tier2sme` as an example: during the same time period, 1,354 members asked questions, and only 145 members replied to at least one question. Within the latter group, only 37 members replied to at least 10 questions, and the top 3 expert agents answered 61.3% of the 20,764 questions. Clearly there is an urgent need for automating as much as possible the Q&A tasks for expert agents.

IV. System Design

In this section, we describe our system workflow as shown in Figure 2. There are five components: data preprocessing

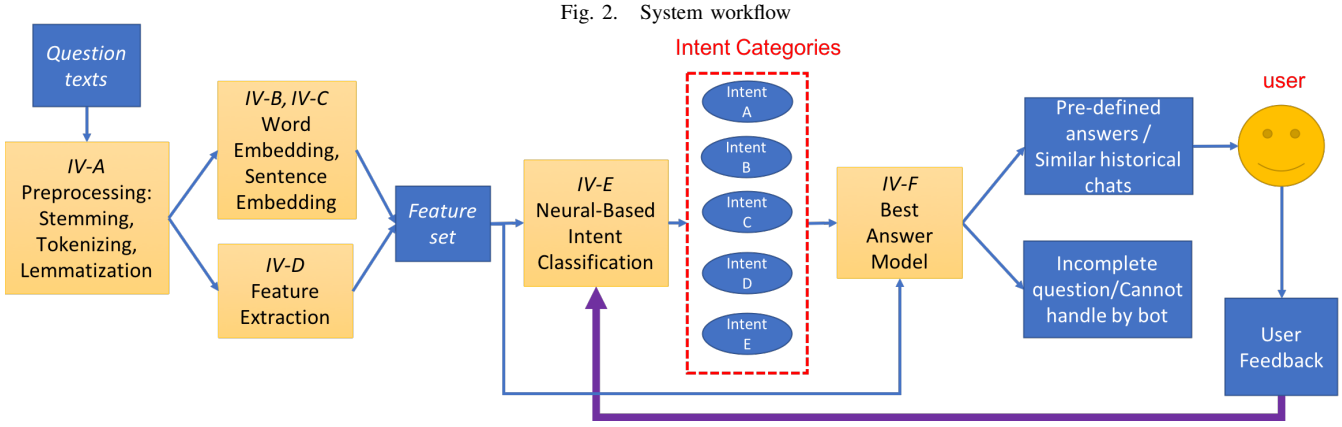
[1] agents who have answered more than 10 questions

TABLE I

STATISTICS OF INTUIT-CARE SLACK CHANNELS

Channel	Questions	General Agents	Expert Agents	Questions Answered by Top 5 Experts (#)	Questions Answered by Top 5 Experts (%)
sbc-ace-tier2	8,491	385	38	6,465	76.13
sbc-ae-tier2sme	20,764	1,354	37	14,544	70.04
cc-act-supportability	6,628	220	19	5,172	78.03
sbc-qdbt-pro-t1-tech	21,753	1,602	25	15,923	73.19
sbc-qdbt-pro-t1-bc	9,888	1,220	15	9,356	94.61
cppl-posa-license-t1	46,721	3,400	25	30,482	65.24

Fig. 2. System workflow



(IV-A), inner representation (IV-B, IV-C, IV-D), intent classification (IV-E) , best answer model (IV-F), feedback loop (IV-G) and application deployment (IV-H).

A. Data Preprocessing

The Slack channel data dump is in JSON format. Every post is indexed by a timestamp. We follow the general text analytic practice for processing the Slack post text data. The posts are first converted to lowercase before we perform stemming, lemmatization and tokenizing. We also use timestamps to join reply posts with the original question posts. In addition, as covered in IV-D, we use "*term replacers*" to replace recognized name entity, email addresses and currency numbers for better word embedding results. To illustrate: company name "Intuit" is replaced by "<company name>"; currency number "$12.54" is replaced by "<currency number>".

B. Word Embedding

Word embedding is a powerful distributed representation method introduced by Mikolov at [5]. A notable property of `word2vec` is that vectors of similar words tend to have close *cosine similarity*.

One challenge for us is that agents often use Intuit-specific acronyms and jargons in Slack channel communication. For example, "cx" is used to mean "customer" and "sf" refers to "Salesforce". To tackle this challenge, we train word embeddings over the Intuit Slack chatlog. We use *gensim*[6] to get the word embeddings for our dataset with window size 5 and negative sampling rate 0.1. By using word embeddings trained from Intuit corpora, we are able to effectively alleviate the acronym problem. For example, the cosine similarity between "cx" and "customer" reaches 0.88; similarly for word pairs ('pto', "paymentech"), ('sf','salesforce'), and ('qbo', 'qbdt'), cosine similarity scores are 0.88, 0.95 and 0.91 respectively.

C. Sentence Embedding

In order to quantify the meaning of each sentence, we use `sentence2vec` to build the embedding for each sentence. The `sentence2vec` model is proposed by [7]. Here we denote each sentence as S. The `sentence2vec` model considers each n-gram inside a sentence and generates the overall sentence embedding by averaging all n-gram's embeddings.

Let $R(S)$ represent the set of all possible n-grams for sentence S and v_w denote each n-gram embedding, the sentence embedding v_S can be computed by using the formula:

$$v_S := \frac{1}{R(S)} v_{R(S)} = \frac{1}{R(S)} \sum_{w \in R(S)} v_w \tag{1}$$

D. Feature Extraction

Customer service is a complex domain where the domain knowledge accumulated over time has great value, especially for a sophisticated financial software product like QuickBooks. Besides relying on their knowledge of prior cases and solutions, agents often recognize the type of customer service questions from *keywords* or *key sentences* in the question texts. For example, a question that contains currency numbers and keywords such as "bills" or "receipts" usually

refers to the case where the customer is confused about a monthly payment increase after a discount period expires. The presence of specific keywords or entities is usually a strong indication of certain types of questions. Therefore, a carefully hand-crafted feature set can lead to better model performance. We introduced the following features to augment our word embedding features:

1) Specific Name Entity: We use *Name Entity Recognition (NER)* techniques to extract meaningful entities such as company names and people names. For this NER task, we use SpaCy[8], a Python compatible natural language processing package.

2) Email Addresses and Currency Numbers: For email addresses and currency numbers, we use pattern-based regular expressions to detect their presence in the text.

3) Identification Numbers: As shown in the example in figure 1, when agents ask questions in a Slack channel, they sometimes provide several associated identification numbers such as Customer Account Number(CAN), case number, and license number. These identification numbers can be extracted by matching prefixes such as "CAN", "Case", or "license".

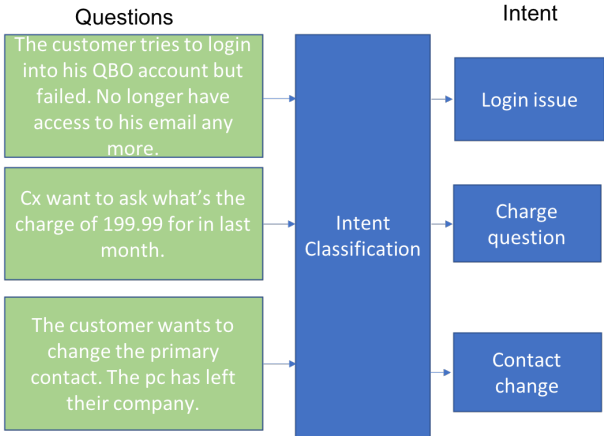
4) Intuit Specific Terms: Intuit specific glossaries and their synonyms are important features. They include terms such as product names (QuickBooks/QBO/QBOA/QBDT), internal information processing platform (paymentech/PTO), OS version (Mac, Windows) and the like.

E. Intent Classification

The goal for the intent classification component is to map each question to a specific predefined intent class. An example of the intent classification is depicted in Figure 3. We define for our model a number of intent classes from each dataset. Features combining word embeddings and hand-crafted features are used by the intent classifier. We employ the attention-based model to generate a vector representation for each question and predict the correct intent class for the question.

Our neural-based intent classification model consists of the following layers:

Fig. 3. Intent classification example



*1) **Input Layer**:* For the input layer, we use the pretrained word embedding from our corpora. We replace unseen words with an `UNSEEN` token which translates to a predefined special embedding for the token.

*2) **BiLSTM Layer**:* We use a bi-directional Long-Short term memory (BiLSTM) network layer to generate a hidden representation for each token.

*3) **Self-Attention Layer**:* We use the self-attention layer to generate an output of a uniform length from sentences of varying lengths. For the self-attention layer, we use the layernorm and residual structure to encapsulate each attention layer.

*4) **Output layer**:* We use a densely connected layer to generate the output label with the *softmax* function.

F. Best Answer Model

The ability to evaluate the similarity between two questions is critical for building our task-specific chatbot. To this end, we adopt the neural-based model to calculate the distributed representation of each sentence from word embedding of every token in the sentence.

Word Mover Distance: When matching a given question with historical questions, we use *word mover distance (WMD)*[9] to evaluate the similarity between two sentences. *Word mover distance* is defined as the minimum value of the sum of the word embedding's consine distance between a set of word pairs. Each word pair consists of one word from the first sentence and the other word from the second sentence. For example, given two sentences: `"The boy eats bananas."` and `"The child enjoys candies."`, we can construct three word pairs: `("boy", children)`, `("eats","enjoys")`, and `("bananas","candies")`. We denote the distance between word i and word j as $c(i,j) = \|x_i - x_j\|_2$. Assuming word i is mapped to word j for T_{ij} times, the word mover distance can be represented as the following constrained optimization problem:

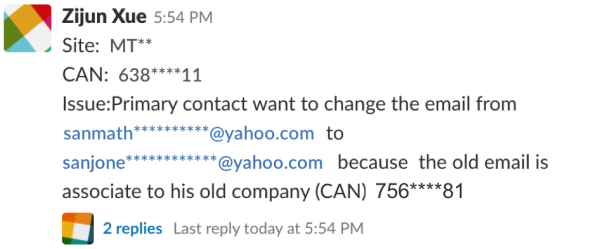
$$\min_{T>0} \sum_{i,j=1}^{n} T_{ij} c(i,j) \tag{2}$$

We calculate the WMD in two phases by quickly filtering out a large number of impossible sentences before calculating the accurate WMD. Specifically, the best answer is generated following these steps: (1) All historical questions in the chatlog are transformed into their sentence embeddings. The new question's sentence embedding is used to select the top 500 closest embeddings within its intent class. (2) Calculate the WMD between the new question and each of the top 500 historical questions identified in step (1) and return the answer associated with the historical question of minimum WMD.

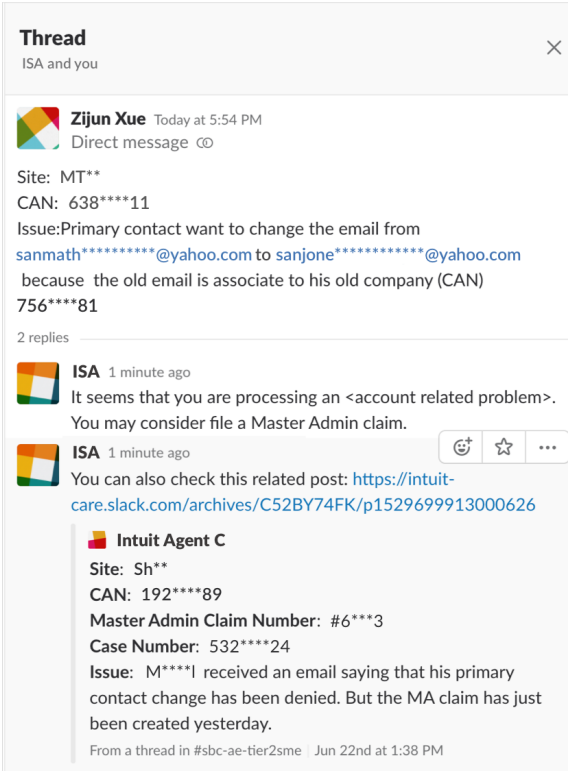
G. User Feedback in the Loop

For continuous improvement of the system, we add thoughtfully designed feedback buttons to the GUI to solicit valuable feedback from users without disrupting the conversation flow. The acquired user feedback is used for

Fig. 4. An example of the issue reported in Slack channel and the extended thread



(a) An example of the issue reported in an agent Slack channel



(b) The example reply text by Slackbot

retraining the core machine learning model to improve model performance.

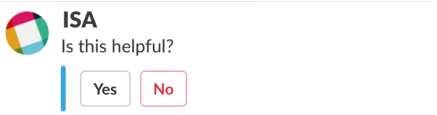
H. Slack Application Deployment

Amazon Web Services (AWS) provides the capability for hosting a Slack application through its *ChatOps* collaboration model[10]. It involves building a Slack app with webhooks and interactive components, writing Lambda functions to handle the interactive messages, and creating a service API to interact with AWS EC2 instances. This part of the system is built with Python 3.5.

V. Demonstration

In this section, we demonstrate the behaviors of *Isa*, our Slack bot. 5(a) illustrates an example of an issue posted by a customer service agent regarding an email contact change request. 5(b) shows Isa's replies to the posted issue. Isa first provides a possible solution for the issue: *file a Master Admin claim for the issue*. Subsequently, links to historical threads of posts similar to this issue are also provided. In the event that the issue is too complicated for the system to find a best answer for, Isa is designed to remain silent. This is because *accuracy* is much more important than *recall* for our system: providing wrong or irrelevant information to the inquiring agents would be counterproductive.

Fig. 5. an example of the response button design



VI. Related Work

The research streams of conversational AI fall into two main categories: general purpose conversational systems (chitchat systems) and task-specific or domain-specific chat systems. Chitchat systems attempt to simulate casual conversations without any specific purpose. In contrast, task-specific chat systems focus on certain subject domains aiming to solve specific tasks.

In this work, we focus on task-oriented conversational systems. Typical techniques for building conversational systems include *sequence-to-sequence* models and *reinforcement learning* frameworks. Conversational AI designers also leverage the information extracted from external databases and previous chat history.

A prominent research direction for task-oriented systems is to adopt end-to-end sequence-to-sequence models. The related work can be classified into two categories. The first category [11][12][13][14] is mainly based on the Memory Network[memory network]. In [15], the author introduces a RNN model with recurrent attention over a large external memory to predict the answer. In [16], the author adopts the memory network from Q&A in the dialogue where the answer is selected by the model from a given set of answers. In [13], the author uses multiple RNNs to construct a dialogue system: the first RNN is used to capture the user's intent and the second one serves as a belief tracker to maintain a multinomial distribution over a given value set.

The second category of end-to-end task-oriented system research focuses more on accessing the external libraries and generating internal representations for the input sentences.[17][18][19]. In [17], Lowe et al use two RNNs to encode both the input and the external unstructured text information while TF-IDF and hashing techniques are used to decide the relevance. [19] uses Freebase as the external knowledge base where named entity recognition is applied to identify the key entity in a sentence followed by extracting relevant entity from the knowledge base to answer more specialized questions.

VII. Discussion

A. Cold Start Problem

During the development of our chatbot, we encounter the cold start problem where we do not have enough labeled data for training the intent classification model. To address this problem, we use *feature-based annotating* method. We create several keyword phrases as feature and apply these keyword phrase filters to the responses of each thread. For example, `"Master Admin claim"` is a common solution for resolving account related issues. Thus, we define a filter to detect `"Master Admin claim"` and its synonyms such as `"MA claim"` to find all matched conversations and label them as `"account related issue"`. We successfully acquire 7,073 labeled data points using this method.

The assumption behind using *feature-based annotation* method is that many questions are asked repetitively. Since answers are mainly provided by a small number of experts, the word selection by this small group of experts should be highly consistent for these repetitive questions. Our *feature-based annotator* can successfully label a large number of questions which serve as the initial data points to train our intent classifier. We believe this approach can also be applied to other domains where answers to repetitive questions are provided by a small group of users.

B. Interaction Design

Although machine learning plays an increasingly important role in improving user experience, UX design is often deemed as important in successful chatbot designs[20].

Since *Isa* is an agent-assist chatbot, we make several special decisions for its interaction design.

First of all, *Isa* does not initiate private individual chats with agents. Instead, it joins a predefined channel and listens to every chat between agents. Such a design enables seamless integration into the existing workflow without requiring agents to change their habit.

Secondly, we choose to have *Isa* send its reply as a follow-up message in the same message thread instead of sending a direct private chat to the inquiring agent or posting a new thread to the channel. This is because a private chat message requires the recipient agent to switch between chat frames, which is not very user-friendly. Similarly, a new thread posted to the channel is broadcast to every member of the channel, which can become unnecessarily distracting for other agents not involved in the message thread.

Lastly, *Isa* displays a feedback widget at the end of the chat to collect feedback that is used to incrementally improve our model over time. This is illustrated in Figure 5. If the inquiring agent does not find the provided answer useful, a follow-up widget containing our intent categories is presented to the agent to obtain the correct intent label from the agent.

VIII. Conclusion

In this work, we present a system that provides a chatbot-like experience for answering repetitive questions to assist seasonal workers and general agents. We discuss the design principles and necessary considerations for such system. We show that a task-specific chatbot suits our needs and propose a solution that employs BiLSTM with attention mechanism. Our system helps customer service agents in providing delightful customer experience by reducing customer wait time and by preserving knowledge generated from customer interactions to power other future applications. Our system also sheds light on how to combine human intelligence and modern machine learning techniques together in delivering high quality customer experience.

References

[1] Anton, Jon and Petouhoff, Natalie L., Customer Relationship Management: The Bottom Line to Optimizing Your ROI (NetEffect Series), Prentice-Hall, Inc., 2001.

[2] Ba, Sulin, and Wayne C. Johansson. "An exploratory study of the impact of eservice process on online customer satisfaction." Production and Operations Management 17.1 (2008): 107-119.

[3] *Slack* http://www.slack.com

[4] *Slack Bot User* https://api.slack.com/bot-users

[5] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[6] *gensim* https://github.com/RaRe-Technologies/gensim

[7] Matteo Pagliardini, Prakhar Gupta, Martin Jaggi, Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features NAACL 2018

[8] Honnibal, Matthew AND Montani, Ines, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,2017.

[9] Kusner, Matt, et al. "From word embeddings to document distances." International Conference on Machine Learning. 2015.

[10] https://aws.amazon.com/blogs/devops/use-slack-chatops-to-deploy-your-code-how-to-integrate-your-pipeline-in-aws-codepipeline-with-your-slack-channel/

[11] Bordes, Antoine, Y-Lan Boureau, and Jason Weston. "Learning end-to-end goal-oriented dialog." arXiv preprint arXiv:1605.07683 (2016).

[12] Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in neural information processing systems. 2015.

[13] Wen, Tsung-Hsien, et al. "A network-based end-to-end trainable task-oriented dialogue system." arXiv preprint arXiv:1604.04562 (2016).

[14] Williams, Jason D., and Geoffrey Zweig. "End-to-end lstm-based dialog control optimized with supervised and reinforcement learning." arXiv preprint arXiv:1606.01269 (2016).

[15] Sainbayar Sukhbaatar, JasonWeston, Rob Fergus, et al. 2015. End-to-end memory networks. In Advances in neural information processing systems. 24402448.

[16] Bordes, Antoine, Y-Lan Boureau, and Jason Weston. "Learning end-to-end goal-oriented dialog." arXiv preprint arXiv:1605.07683 (2016).

[17] Lowe, Ryan, et al. "Incorporating unstructured textual knowledge sources into neural dialogue systems." Neural Information Processing Systems Workshop on Machine Learning for Spoken Language Understanding. 2015.

[18] Ghazvininejad, Marjan, et al. "A knowledge-grounded neural conversation model." arXiv preprint arXiv:1702.01932 (2017).

[19] Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue. 129133.

[20] Flstad, Asbjrn, and Petter Bae Brandtzg. "Chatbots and the new world of HCI." interactions 24.4 (2017): 38-42.