COBOTS - A Cognitive Multi-Bot Conversational Framework for Technical Support

Industrial Applications Track

Sethuramalingam Subramaniam
IBM Research AI
Bangalore, Karnataka, India
sethsubr@in.ibm.com

Pooja Aggarwal
IBM Research AI
Bangalore, Karnataka, India
Aggarwal.Pooja@in.ibm.com

Gargi B Dasgupta
IBM Research AI
Bangalore, Karnataka, India
GaargiDasgupta@in.ibm.com

Amit Paradkar
IBM Research AI
Yorktown Heights, New York, US
paradkar@us.ibm.com

ABSTRACT

Human technical support agents spend significant time interacting with customers via various channels of voice, email and chat. There is a massive incentive to automate support with autonomous agents with the goal of reducing manual effort and time taken for problem resolution. As technical support questions are complex and diverse, building a generic agent capable of solving multiple domains is implausible. In this paper, we describe a scalable conversational framework that automates the process of guided troubleshooting called COBOTS (COgnitive BOts for Technical Support). Our underlying premise is that scalability in such frameworks can be achieved by control and co-ordination across multiple domain expert bots. These bots co-ordinate to (a) understand user problems from natural language queries (b) engage in conversation and (c) provide assistance with troubleshooting. All of the above is done with minimum human assistance. COBOTS framework comprises of User Bots that monitor customer infrastructure for issues, the Orchestrator bot which co-ordinates and controls various request-response pairs and Domain Expert bots which handle issues pertaining to their domains, respectively. In a real environment, we have deployed an implementation of our COBOTS framework which can co-ordinate and control user queries across 11 different technical support domains. When evaluated by two different teams of expert support users, it was observed that more than 75% of the time our application was able to provide relevant solutions for their queries.

KEYWORDS

Conversational Agents; Chat bots; Dialog systems

ACM Reference Format:
Sethuramalingam Subramaniam, Pooja Aggarwal, Gargi B Dasgupta, and Amit Paradkar. 2018. COBOTS - A Cognitive Multi-Bot Conversational Framework for Technical Support. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 8 pages.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

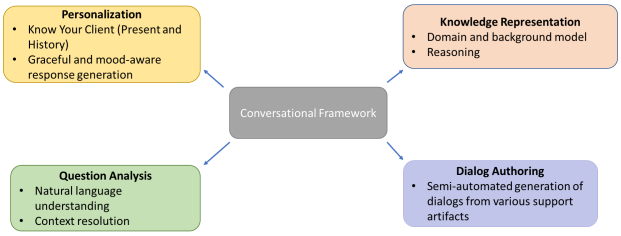

Figure 1: Conversation framework

1 INTRODUCTION

Guided troubleshooting in technical support is the process of assisting customers with step-by-step troubleshooting of their problems. Questions in this domain tend to be complex. Consider a sample query in this domain: *SRC 2b4c8009 failed cache battery. On dc01 dc03 will fail in 123 days. Please replace both.* It contains the attributes of **symptom**:"failed cache battery", **error code**:"2b4c8008" and **intent**:"please replace both". The solution to this problem comprises of multiple steps of validating the error, finding the root cause and executing a fix. Guided troubleshooting through a conversational interface entails automated understanding of the question and its attributes, deciding the next course of action and creating automatic responses for them.

Companies have found conversation as a new and engaging means to connect to their users [4, 25].Most of the existing bot frameworks [8, 10, 26] contain natural language interfaces that can be plugged in with existing applications enabling them to be conversational. Even in the guided troubleshooting space, conversational bot frameworks like Amelia [7] and Robotic automation [13] have been around for a few years now. Each one of these conversational systems at a high level comprises of the main components of natural language understanding, personalization and retrieval mechanisms from one or more knowledge repositories as shown in Figure 1

However for technical support, seamless scaling of conversational systems across multiple domains becomes a very important need. A significant amount of manual work is needed in creating conversation flows in these solutions and repeating the manual work for multiple domains does not scale. Hence, automatically

extracting domain knowledge and representing them in a suitable way to create conversational flows automatically is an interesting research problem.In this paper we present COBOTS, a multi-bot framework for technical support that leverages interactions and coordination among several bots to achieve complex tasks in guided troubleshooting. The main contribution of our work is to create this as a scalable and repeatable architecture across thousands of technical support products and services.

The rest of the paper is organized as follows: Section 2 presents some related work and a brief background of the various conversational frameworks. Section 3 describes our system architecture. In Section 4, 5, and 6, we discuss our three bots i.e User Bot, Orchestrator bot, and Domain Expert bot in detail. Section 7, we briefly describe the way COBOTS framework is implemented. Section 8 shows the experiments done to test the COBOTS framework with help of expert agents. Section 9 concludes the work done in the paper and highlights the scope for future work.

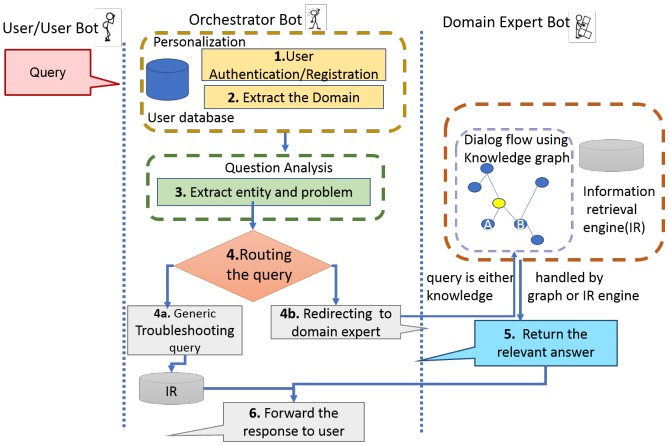

Figure 2: System Architecture

2 BACKGROUND AND RELATED WORK

Conversational bots have their origin dated back to the 1960's with the famous ELIZA bot [29] built by Joseph Weizenbaum, generated responses based on simple keyword matching rules. Following decades saw several attempts to build chat bots with more sophisticated bot architectures like MegaHAL [10] (Using Hidden Markov Models), CONVERSE system [3], ELIZABETH [24]. The A.L.I.C.E (Artificial Linguistic Internet Computer Entity) or Alice bot [26] is another famous patterns-based system by Wallace that involves building intelligent bots using rules defined using the Artificial Intelligence Mark-up Language (AIML).

Though building conversational systems for technical support can be highly advantageous, it consists of some challenging problems like multitude of problem categories, identifying actual root cause from vague user inputs, technical complexity of the inherent issues and limited knowledge of customers on those issues [1]. There have been a plethora of conversational platforms and systems like IPSoft's Amelia [7], Api.ai , Microsoft's Language Understanding Intelligent Service (LUIS) , Wit.ai etc., in the recent years but none of them is known to comprise or discuss a similar scalable, multi-bot framework for technical support as discussed in the current work.

The spoken language research community refer these conversational bots as Dialog Systems and the early research dates back to late 1990s when Dialog Systems [16] and Dialog modules [2] were introduced to generalize conversational flows and re-usability of dialog components. The current research in Dialog systems can be broadly categorized into three main streams - Natural Language Understanding (NLU), Dialog Management (DM) and Natural Language Generation (NLG) [20].

Natural Language Understanding refers to analyzing user input and producing a representation of its meaning which can be used by the Dialog Management module. NLU includes identifying the domain of the utterance and categorizing the user intent. There have been earlier work on categorizing intents as Topics in Customer Service Domain[23].In addition to intent categorization, systems use several approaches to obtain the semantic representation of the input. Some of the techniques involve Syntax-driven methods which are language-specific e.g. The Phoenix System[27], Statistical and Machine Learning based methods including the recent Deep Learning based ones [9],[15],[17],[6],[21]. Our proposed architecture has capabilities to extract relevant entities such as symptom, intent etc. using Watson Deep parsing [19]. Dialog management is achieved using state-less, context-based traversals over the Knowledge Graphs and the dialog authoring capabilities serve in automatic response generation.

3 SYSTEM ARCHITECTURE

In this section, we explain the overall architecture of our conversational framework. The three main actors of our framework are User Bots, the Orchestrator Bot, and Domain Expert Bots. User Bots are deployed at customer sites for monitoring one or more user services. A conversation in COBOTS could be triggered by either a user coming to the platform with a problem or a User Bot when it encounters some error event. The Orchestrator bot acts as the front-end to User Bots or to users for understanding their queries, specified in natural language, and provide a relevant answer. The Domain Expert bot handles domain specific user queries using its Knowledge Graph (KG), passed to it by the Orchestrator bot, to provide a relevant solution. Figure 2 shows the sequence of actions that gets triggered in case of a problem resolution.

(1) *User Authentication/Registration*: On receiving a user query, the Orchestrator bot at first authenticates the user and asks for a new sign-in if not already registered. Once authenticated, question analysis is performed to: (a) identify the domain which the question belongs to and (b) understand the problem being faced by the user.
(2) *Domain Identification*: The Orchestrator bot identifies the domain either from the user query or from the past interactions saved in the user database, making the service truly personalized. Further details are explained in the Section 5.
(3) *Entity and Problem Extraction*: The next step of the Orchestrator bot is to determine what the user is asking for i.e *intent* and what are the key entities being talked about in the user query as shown in the Table 1 . We consider these

entities and intent as enriched data along with the original user query.

(4) *Query Routing*: Based on the query and the extracted attributes, the Orchestrator bot routes the query either to its own Information Retrieval (IR) engine or to a domain specific bot if the query is generic troubleshooting or specific to a domain respectively. **Generic Troubleshooting query:** Queries with low classification scores across all domains are considered as generic troubleshooting queries. The Orchestrator bot handles these queries by passing them to its own IR engine and depending upon the attributes and user query, it provides with relevant solution URLs. **Redirect to Domain Expert:** For the domain specific queries, the Orchestrator bot initiates the conversation with the corresponding Domain Expert bot by forwarding the user query along with the enriched data.

(5) *Answer Extraction* When a Domain Expert bot receives the user query from the Orchestrator bot, its corresponding knowledge graph is traversed, using the query and the enriched data, and a particular node (*nodeMatch*) matching the user query is highlighted. The Domain Expert bot asks the next question based on the child nodes of the node *nodeMatch* as show in the Figure 4. Conversation between the Orchestrator bot and the Domain Expert bot is reasoned around its domain-specific knowledge graph. User or the User Bot shares the necessary details, with the Domain Expert bot, through the Orchestrator bot. The Domain Expert bot either answers using its knowledge graph or in some cases uses its IR engine to provide the relevant response.

(6) *Graceful Handing-off to a human agent*: Due to dynamic nature of the dialog, it is possible that neither the Orchestrator bot's IR engine nor the Domain Expert bot is able to resolve the user query. In such cases, the Orchestrator bot redirects the query to a human agent. The human agent takes over from there and provide a solution to the user query. The chat session between the user and the expert is saved and can be used for future analysis.

(7) Forward the response to the user: Orchestrator bot collects the response from its IR engine or the Domain Expert bot or at times from the human agent and forwards it to the user or her bot.

Note that one of the main motivations of the COBOTS framework is to abstract out all the bot-to-bot conversational complexities from the user and to provide a seamless conversational experience. The user is totally unaware of the presence of multiple bots behind and the numerous back and forth messaging between the Orchestrator and Domain Expert bots. Whether it is the interaction of a customer with the Orchestrator bot or a conversation initiated by the Orchestrator bot with a Domain Expert bot, a seamless conversation is provided across all channels by the COBOTS framework. The co-ordinated control and hand-off between the Orchestrator and the Domain Expert bots help COBOTS scale up to multiple domains without affecting rest of the conversation flow. Using automatically extracted knowledge graphs to drive the conversation helps reduce manual work in creating these multi-domain solutions.

Next, we explain the functionality of each of the three actor bots in detail.

4 USER BOT

With the large number of systems that are operational in today's top business organizations, problems may go unnoticed until multiple issues arise. A combination of issues in a large environment can result in either data loss or system downtime. In such scenarios, Call Home [22] like continuous monitoring systems are designed to increase system availability by providing round-the-clock monitoring of the underlying infrastructure. We model User Bots after the Call Home capability, such that they pro-actively monitor users infrastructure. On detecting issues in the system like slow performance, disk failure etc., the user bot determines the problem severity, collects contextual information (like operating system details, machine type, machine model, patch levels, memory and processor foot prints, etc.), collects logs for all contextual information and automatically triggers a chat session with the Orchestrator bot. It share evidences for the errors observed, contextual details collected and if required can also upload error logs. On receiving a resolution from the Orchestrator bot, the User bot applies the recommended solution, validates the success or failure of the solution, notifies the Orchestrator bot on the outcome (for continuous learning if the resolution was successful) or for the next course of action (if failure occurred). The User Bot also notifies the customer via email regarding the issue and the subsequent steps taken so far.

5 ORCHESTRATOR BOT

The Orchestrator bot has the capability of understanding the natural language user query and to provide a relevant response to user either from a Domain Expert bot or from its own IR engine. Based on the input query, it tries to identify the user's intent, key entities being talked about and the domain which the query belongs to. The domain prediction is achieved using trained Machine Learning models for each of the existing domains. Once an input query is categorized to its most likely domain category, the Orchestrator initiates a conversation with the corresponding Domain Expert bot. For example the Domain Expert bot could also request for additional details like log files, hardware details etc. In that case the Orchestrator Bot sends back these requests to the User Bot/user, collects their inputs and forwards them to the appropriate Domain Expert bot. In cases where the initial domain gets wrongly categorized and the corresponding Domain Expert bot declines to handles such queries, the Orchestrator bot forwards the query to the next most likely Domain Expert bot based on the domain prediction score. It is also capable of handling generic troubleshooting questions for which it uses its own IR engine to provide a relevant response. Two key components of this bot are Question Analysis and Personalization. Next, we explain each of these components in detail.

5.1 Question Analysis

The first stage of processing in the COBOTS framework is to perform a detailed analysis of the support question in order to determine the problem being faced by the user (*symptom*), what the user is asking for *intent* and what are the key entities or words from the user message such that the words describe the component of

Table 1: Examples of Question Analysis

Original Query	Entities Extracted	Domain	Intent
I am facing an issue with my tape drive. would like to know how can I get it replaced.	Tape drive	Power	issue with tape drive. replace it
please ship a replacement part for a failed 600 gb sata disk drive (fru 00l4521). no CE is required	Disk Drive	STORWIZE	failed disk drive. ship a replacement
how do i fix windows memory utilization issues	windows, memory utilization	Generic	memory utilization issues
add 2GB of memory to the server id3434	2GB, memory, server id3434	Transaction	memory update

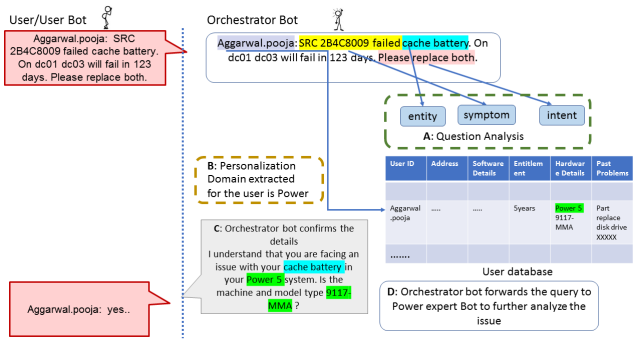

Figure 3: Illustration of Question Analysis and Personalization of User query

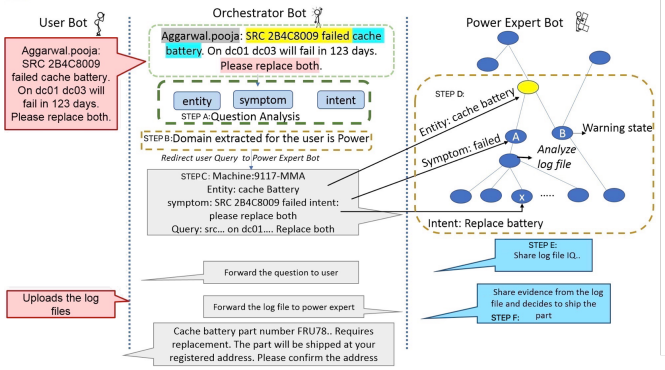

Figure 4: Illustration of the Domain Expert Bot's query handling

concern to the user (*entity*). Since the Knowledge Graph for different domains are also rooted at entities for that domain, extracting entities is the first step of our question analysis pipeline.

We use IBM Watson's parsing and semantic analysis capabilities for linguistic analysis of text in the user query [14] to extract user's intent, symptom and entity. There are several natural language parsers that allow tokenization and POS tagging. Some also provide a simple description of the grammatical relations in a sentence, but this is not sufficient as technical support questions especially the ones that are pertainable to guided troubleshooting are complex from the point of understanding. Watson uses two deep parsing components: English Slot Grammar (ESG) followed by Predicate Argument Structure (PAS) for linguistic analysis of text [18]. This helps in automatically extracting the attributes from the complex and diverse user queries. Table 1 shows the extractions from the actual support queries from various domains. Figure 3 (Step A: Question Analysis) shows that for a given user query *SRC 2B4C8009 failed cache battery. On dc01 dc03 will fail in 123 days. Please replace both.*, the deep parser module extracts the entity *cache battery*, the symptom *SR 2b4c8009 failed* and the intent *please replace both*. When the Orchestrator bot forwards the user query to the Domain Expert bot, then these attributes are used by the Domain Expert bot to traverse the knowledge graph.

5.2 Personalization

Personalized support for users becomes even more important, when users work on time constrained, dynamic service-oriented environment. As technical support questions are complex and diverse, it is important to save user profile so that it can be used in solving future problems. Personalization acts as an enrichment layer which helps in identifying the domain which the user query belongs to and also helps in reducing the dialog between the user and the Orchestrator bot by extracting the information from the past problems.

Once the user's *intent*, *symptom* and *entities* are correctly identified from the query, the Orchestrator bot then tries to identify the domain from the current query itself or from the past conversations stored in user database. For example, user query shown in Figure 3 has no information regarding the product or the domain the query belongs to. In this case, the Orchestrator bot queries the user database to extract the details of the user and identifies that the user is registered with *Power* system machines (Step B:Personalization in Figure 3). It is possible that the same user is registered with multiple domains or a single domain has multiple systems. In such cases, the Orchestrator bot shows the list of domains/systems with which the user is registered and asks the user to confirm the details (Step C:in Figure 3). In case the user specifies a new domain or system, the Orchestrator bot updates the user details for future reference. Orchestrator bot takes these attributes *entity*, *symptom*, *intent* and *domain* and forwards them along with user query to the *Power* Domain Expert bot for further analysis (Step D: in Figure 3).

Once the current issue is resolved by the Domain Expert bot, the Orchestrator bot updates the details regarding the issue in the user database. In case of a recurring issue, the Orchestrator bot along with the new query and attributes, also forwards the similar past issues to the Domain Expert bot enabling it to do a detailed analysis and provide a better resolution.

6 DOMAIN EXPERT BOT

The Domain Expert Bots are considered to be expert agents capable of handling issues related to part replacement, firmware update, service pack installation, configuration of the site, log file analysis, etc.

They interact only with the Orchestrator bot and COBOTS framework ensures seamless interactions between Orchestrator bot and the multiple Domain Expert bots such that the user may not even know about the existence of the latter. Triggering of the Domain Expert happens when it receives a message from the Orchestrator asking it to handle a user query in its domain. The domain bot then uses both the KG and IR engine to handle a user query. Multiple interactions with the user (via the Orchestrator bot) is performed on the KG to return an answer. If a relevant response cannot be provided to the user based on the KG, the domain expert fires a search query to its IR engine.

An example shown in the Figure 4 (Step A), illustrates a case where the Orchestrator bot forwards the user request to Power Expert bot along with the attributes. The message send to the expert bot contains the user query, entity, symptom, intent, and if possible machine model details. The attribute extractions are mapped to the Power Expert bot's KG [11] (Step B:Figure 4). Power Bot understands that the problem is related to cache battery, it has failed and the user wants to get the cache battery replaced. But before going forward with the replacement, Power expert bot asks for log files from the user so that it can diagnose the problem properly and see whether actually the replacement is required or there is some other issue(Step C:Figure 4). From the knowledge graph shown in Figure 4, we can see if the cache battery fails then one needs to analyze the log file based on which one of the following actions reconfiguring the cache battery, check the power supply cable, replace the cache battery, etc can be performed. To understand which node to follow next, Power Expert bot collects evidences from the log files and then continues the conversation with user. In this case, Power Expert bot decides that the cache battery needs to be shipped at the customer's site (Step D:Figure 4). Even though an explicit intent (replace cache battery) was specified by the user, then also the bot follows the complete path to reach the *intent* node in the knowledge graph and then provide a relevant resolution. The two key building blocks of the Domain Expert bot are the knowledge represented in the form of a graph and the dialog flow which gets created using its instances.

6.1 Knowledge Representation

Every Domain Expert bot relies on a Knowledge Graph (KG) built from its domain knowledge sources and historical ticket data. Attributes extracted from user inputs are used by the Domain Expert Bot for traversing its KG to provide the relevant solution. Instances in the KG follow a modified version of the Common Data Model (CDM) ontology [5] for service management. An excerpt from the ontology and a few instances for the IBM Power Systems [11] are presented in Figure 5. The ontology connects entities in the domain (like cache battery) to Symptoms that it can exhibit, via the *exhibits* relation, and Intents that can be executed on it, via the *has_task* relation. Intents are connected to Symptoms that they treat or mitigate via the *treats* relation. Finally, Intents are connected to Solution via the *has_solution* relation. Solutions contains the steps/procedure to perform a task

Instances for the Knowledge Graph are extracted from knowledge sources and historical ticket data leveraging the models created

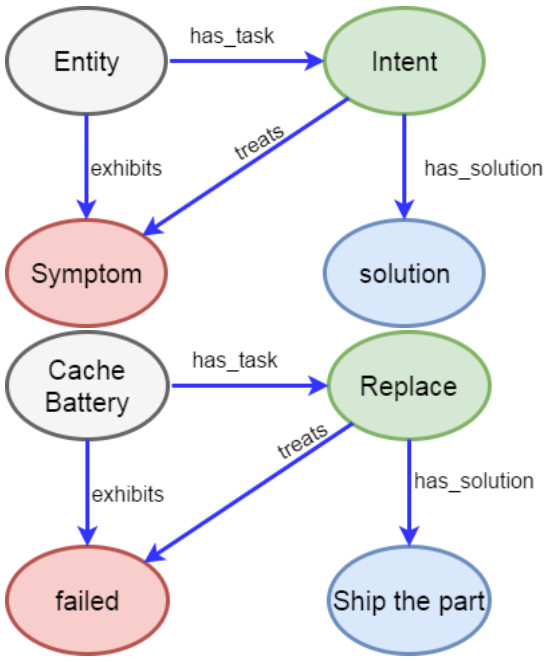

Figure 5: Sample Ontology and Knowledge Graph instances for Technical Support

for question analysis. Existing hierarchical structure in the knowledge sources, like Knowledge Center documents for IBM Power Systems, are also useful for extracting instances.

6.2 Dialog Authoring

Domain Expert Bots might need multiple interactions with the user (via the Orchestrator bot) to reach a resolution. This interaction requires the Domain Expert Bots to converse based on the instances of the Knowledge Graph for their domains. We use Watson Assistant [28] from IBM as our dialog manager. Watson Assistant, formerly known as Watson Conversation Service is a cloud-based dialog management service for creating chat bots using IBM Watson's state-of-the-art conversational framework. A typical Watson Assistant workspace requires hand-coding of the dialog flow and its related components. Hence its a completely human intensive task. But in our approach, based on our domain-specific KGs, we generate Watson Assistant workspace JSON files corresponding to each domain automatically and link them to our master workspace. So, the generated conversation workspace acts as the dialog manager and acts as the backbone of our conversational flow. Figure 6 shows a snapshot of the automatically generated conversation flow for the Storwize product of IBM. The Storwize workspace is generated from its KG, which in turn is generated from the Knowledge Center pages on Storwize[12].

Nodes in the conversation flow roughly correspond to nodes in the Knowledge Graph, with the extractions from user responses being compared to the value of the Knowledge Graph node. For example, the extraction *failed disk drive* from the question might match the KG node *failed drive*. The comparisons are done with a fuzzy match instead of an exact match, to account for various

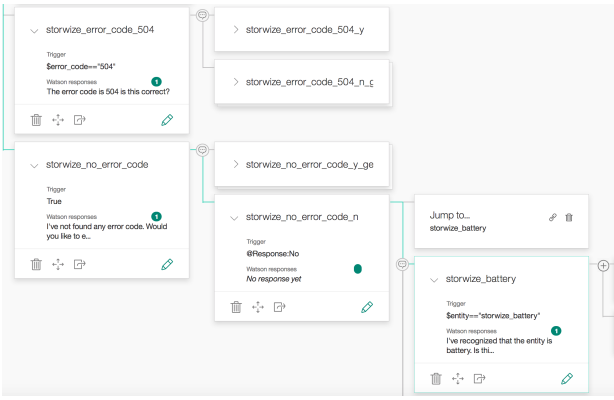

Figure 6: Excerpt of the automatically generated Watson Assistant conversation flow

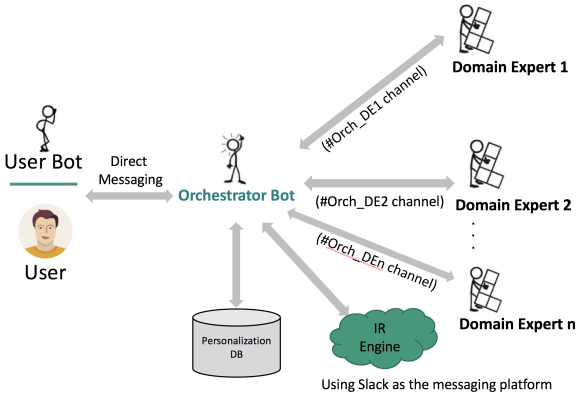

Figure 7: The COBOTS Implementation

ground forms that can appear in user questions. There are a few extra nodes to confirm the matches performed. The response generated from the bot is stored within each node, and is returned back when traversal reaches that node.

7 COBOTS IMPLEMENTATION

In this section, we describe the current implementation of COBOTS. One underlying premise is that scalability in such frameworks can be achieved by control and co-ordination across multiple bots. The current implementation is based on the architecture shown in Figure 2. The framework is implemented in Java with Watson Assistant[28] as dialog framework and Slack being conversational front-end using the SimpleSlack API . Though Slack was just a medium of choice, our framework can be realized using any other chat platforms like Facebook Messenger, Twitter, etc.

In the current implementation, every bot possesses three main events-based functionalities - *Hears*, *Speaks* and *Brain*. *Hears* event fires when there is an input from a user or from any other bots. We were able to achieve the Bot-to-Bot communication via Slack Channels. Every bot listens to any Direct Message or to any new group message in its enrolled channels. In addition to textual messages, *Hears* event gets triggered on file uploads. *Speaks* event gets triggered whenever a bot needs to send response to the messaged user or bot in the respective channel in the form of text or file output. *Brain* method is responsible for executing the business logic corresponding to the current input and context. The main functionalities of intent classification, question analysis, dialog management and dialog authoring etc. get triggered from this method. In our system, we support seamless interactions between the Orchestrator bot and 11 different Domain Expert bots via 11 different Slack channels. A key advantage in our framework is the plug-and-play nature of adding new Domain Expert bots. We have worked with several technical teams with their own domain based services and have plugged them in as Domain Expert bots in the system. Some of these Domain Experts include bots for IBM Power systems, IBM Storage, Network and Middleware etc.

In a real-time environment, we have deployed our current implementation of the COBOTS framework as part of the IBM's first Cognitive Services Platform[1] this year.

8 CASE STUDIES

This section presents the case studies done with two domains of server and storage. We picked the domains which were complex and had about thousands of tickets every year. The main objective of this evaluation was to measure the performance of the major components of our framework.

8.1 Case Study on Server Domain Tickets

We performed an end-to-end evaluation of our implemented COBOTS framework on the tickets data from IBM Power Series servers in terms of its ability to handle direct questions that can be handled with domain-specific rules and more complex open-ended questions. IBM Power Series tickets are maintained across two different ticketing systems namely RETAIN and RCMS by IBM. IBM Storage and Server businesses each have a footprint of 4-5K customers in North America across 30K locations. The total number of tickets considered, and the number of tickets handled by the different components of the COBOTS framework are detailed in the Table 2. Human evaluation was performed 2 by a team of 20 human experts evaluation belonging to the IBM Power domain. Overall, 73% of the IBM Power tickets were either resolved or received relevant solution from COBOTS. It can be noted that around 42% of the tickets hit the IR engine of the Orchestrator bot. Hence we decided to perform a separate evaluation of the Orchestrator bot's IR engine performance. The details of the evaluation are explained in the Section 8.3.

8.2 Case Study on Storage Domain Tickets

Similar to our analysis of our system's performance on IBM Power domain, we evaluated our system's efficacy on IBM Storwize as our next case study. In IBM Storwize domain, queries requesting for new part replacements were considered as our target category. Based on our numbers gathered, 30% of 7000 tickets in a month fall under the *partreplacement* category. From this set of 2100 tickets,

[1] http://www-03.ibm.com/press/us/en/pressrelease/52781.wss

Table 2: Server Domain Tickets Evaluation Results

Criteria	Count of tickets
(a) Total number of tickets considered	23916
(b) Number of tickets with relevant resolution based on domain-specific rules	9524 (40%)
(b1) Number of tickets where logs were required by the Domain Expert bot	1859/9524 (8%)
(b2) Number of tickets handled by the Orchestrator bot based on the previous history	7665/9524 (32%)
(c) Number of tickets enriched and sent to human agents for further analysis	8009 (34%)
(c1) Number of tickets resolved using the IR engine of the Domain Expert bot	4269/8009 (18%)
(c2) Number of tickets where the Orchestrator bot returned a valid link	3740/8009 (16%)
(d) How many where handled by the IR Engine of the Orchestrator bot	10123 (42%)

Table 3: Storage Domain Tickets Evaluation Results

Criteria	Count of tickets
(a) Total number of tickets considered	7000
(b) Number of tickets belonging to *partreplacement* category	2100 (30%)
(c) Number of tickets considered for manual evaluation by experts	900/2100
(d) Number of tickets marked as either *resolved* or *relevant* by experts	695/900 (77%)

we evaluated a sample of 900 tickets by a team of experts from the IBM Storwize domain. We found that 695 tickets (77%) were marked as either resolved or relevant solution was provided by our COBOTS framework.

8.3 Case Study on the Orchestrator Bot's IR Engine Performance

We evaluated COBOTS system through two experiments both targeting IT Service Management domain. In the first experiment, 21 technical services architects with expertise in various IT Service Management sub-domains (such as Storage, Networking, Database etc) asked COBOTS system a total of 202 questions. These architects evaluated each of the 662 responses suggested by COBOTS on a scale of "Resolved" - meaning the suggestion contained the expected answer, "Contributed" - meaning the suggestion had some useful information contained but not the complete solution, and "Not Related" - meaning the suggestion was not useful at all. If one of the top 3 suggestions to each question was rated either "Resolved" or "Contributed", we considered that question to have a good answer. We chose this evaluation metric over others such as Normalized Discounted Cumulative Gain due to its simplicity. Using this evaluation metric, 141 out of the 202 questions had at least one good answers in top 3 resulting in an overall accuracy of 70%. Further, 85 questions (42%) had one answer with a Resolved rating. Figure 8 shows the overall accuracy by topics.

The results also indicated that 37 questions had their intent misclassified and hence no answers were returned. This substantiates our need to attain scalability across multiple domains with this multi-bot framework. Via co-ordinated control and hand-off we are able to handle multiple domains with low error (37 out of 202 questions ~18% error) in accuracy. However we are still working

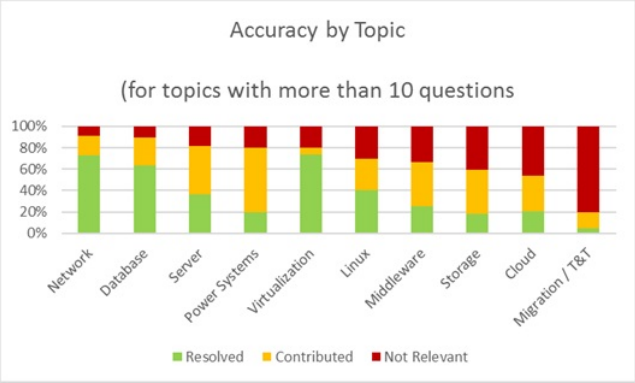

Figure 8: Accuracy of the system by topics

on further improvements for increasing the accuracy of domain understanding.

We also recognized the need to improve content/content coverage. For about 24 questions (~11%), these had no good answers specially for two categories Cloud and MigrationT&T.

In the second experiment, sample questions came from a set of ground truth question-answer pairs prepared for a service desk virtual agent targeted to the sub-domain of a popular email client. A total of 148 unique questions were presented to the COBOTS system, and the resulting 732 suggestions were evaluated in the same manner as before (here the suggestion was rated "Resolved" if it contained the same information as the original answer from the ground truth data). The COBOTS system achieved an accuracy of 81% (120 out of 148 questions had a good answer), with 76% (112 out of 148) questions having a Resolved answer.

9 CONCLUSION AND FUTURE WORK

In this paper, we have presented a scalable, multi-bot framework to build conversational systems for technical support by leveraging interactions and co-ordination between bots to automate the process of guided troubleshooting. Going forward, we can group the current limitations of our system and future work in four directions - 1. Improving the intent categorization accuracy 2. Improving the content coverage for some of the specific topics 3. Currently the queries handed off to human agents by the system are just saved as chat sessions. As our next extension, we plan to mine these chat session to derive new insights on specific topics and issues. 4. We also plan to work on generating personalized responses by analyzing the current tone and emotion of user to make it further more human like interaction.

REFERENCES

[1] Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, and Roberto Pieraccini. [n. d.]. Technical Support Dialog Systems: Issues, Problems, and Solutions. In *NAACL-HLT-Dialog '07*. 25–31.

[2] Barnard, E., Halberstadt, A., Kotelly, C., Phillips, M. [n. d.]. A Consistent Approach To Designing Spoken-dialog Systems. In *Proceedings of IEEE ASRU'99*.

[3] B. Batacharia, D. Levy, R. Catizone, A. Krotov, and Y. Wilks. 1999. *CONVERSE: a Conversational Companion*. Springer US, Boston, MA, 205–215. https://doi.org/10

[4] Jerome R Bellegarda. 2014. Spoken language understanding for natural interaction: The siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 3–14.

[5] CDM 2017. Common Data Model (CDM). (2017). https://goo.gl/AqUhNA.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (almost) from Scratch. *CoRR* abs/1103.0398 (2011).

[7] Karl Flinders. 2015. Amelia, the IPsoft robot. *Computer Weekly* (2015).

[8] Cezar Guimaraes. 2017. Azure SREBot: More than a Chatbot {\textemdash} an Intelligent Bot to Crush Mitigation Time. (2017).

[9] Yulan He and Steve Young. 2006. Spoken language understanding using the Hidden Vector State Model. *Speech Communication* 48, 3-4 (March 2006), 262–275.

[10] Jason Hutchens and Mike Alder. 1998. *Introducing MegaHAL*. Technical Report.

[11] IBM Power servers 2017. IBM Power servers. (2017). https://www-03.ibm.com/systems/power/hardware/

[12] IBM Storwize KC 2017. IBM Knowledge Center on IBM Storwize. (2017). https://www.ibm.com/support/knowledgecenter/ST5Q4U/landing/v7000_unified_welcome.htm

[13] Mary Lacity, Leslie P Willcocks, and Andrew Craig. 2015. Robotic process automation at Telefónica O2. (2015).

[14] Adam Lally, John M Prager, Michael C McCord, Branimir K Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How Watson reads a clue. *IBM Journal of Research and Development* 56, 3.4 (2012), 2–1.

[15] F. Lefevre. [n. d.]. Dynamic Bayesian Networks and Discriminative Classifiers for Multi-Stage Semantic Interpretation. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, Vol. 4. IV–13–IV–16. https://doi.org/10.1109/ICASSP.2007.367151

[16] Bernd Ludwig, Günther Görz, and Heinrich Niemann. 1998. Combining Expression and Content in Domains for Dialog Managers. In *Proceedings of the International Description Logics Workshop DL-98*.

[17] K. Macherey, O. Bender, and H. Ney. 2009. Applications of Statistical Machine Translation Approaches to Spoken Language Understanding. *IEEE Transactions on Audio, Speech, and Language Processing* 17, 4 (May 2009), 803–818. https://doi.org/10.1109/TASL.2009.2014262

[18] Michael C. McCord. 1990. Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In *Proceedings of the International Symposium on Natural Language and Logic*. 118–145.

[19] M. C. McCord, J. W. Murdock, and B. K. Boguraev. 2012. Deep parsing in Watson. *IBM Journal of Research and Development* 56, 3.4 (May 2012), 3:1–3:15. https://doi.org/10.1147/JRD.2012.2185409

[20] Callejas Zoraida McTear, Michael and David Griol Barres. 2016. *The Conversational Interface - Talking to Smart Devices*. Springer.

[21] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23, 3 (March 2015), 530–539. https://doi.org/10.1109/TASLP.2014.2383614

[22] Jim Mitchell, Daniel Henderson, and George Ahrens. 2005. IBM POWER5 processor-based servers: A highly available design for business-critical applications. *IBM White paper* (2005).

[23] Manex Serras, Naiara Perez, M. Inés Torres, Arantza Del Pozo, and Raquel Justo. 2015. *Topic Classifier for Customer Service Dialog Systems*. 140–148. https://doi.org/10.1007/978-3-319-24033-6_16

[24] BA Shawar and E Atwell. 2002. *A comparison between Alice and Elizabeth chatbot systems*. University of Leeds, School of Computing research report 2002.19.

[25] ViV 2012. ViV. (2012). http://viv.ai/

[26] Richard S. Wallace. 2009. *The Anatomy of A.L.I.C.E.* Springer Netherlands, Dordrecht, 181–210. https://doi.org/10

[27] W. Ward. 1991. Understanding spontaneous speech: the Phoenix system. In *Proceedings of ICASSP' 91*. 365–367 vol.1. https://doi.org/10.1109/ICASSP.1991.150352

[28] Watson Assistant 2018. Watson Assistant. (2018). https://www.ibm.com/watson/services/conversation/.

[29] Joseph Weizenbaum. 1966. ELIZA-a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Commun. ACM* 9, 1 (1966). https://doi.org/10.1145/365153.365168