Enabling Interactive Answering of Procedural Questions

Anutosh Maitra(✉), Shivam Garg, and Shubhashis Sengupta

Accenture Technology Labs, Bangalore 560037, India
anutosh.maitra@accenture.com

Abstract. A mechanism to enable task oriented procedural question answering system for user assistance in English is presented in this paper. The primary aim is to create an answering "corpus" in a tree-form from unstructured document passages. This corpus is used to respond to the queries interactively to assist in completing a technical task. Reference manuals, documents or webpages are scraped to identify the sections depicting a "procedure" through machine learning techniques and then an integrated task tree with extracted procedural knowledge from text is generated. The automated mechanism breaks the procedural sections into steps, the appropriate "decision points" are identified, the interactive utterances are generated to gain user inputs and the alternative paths are created to complete the tree. Conventional tree traversal mechanism provides step by step guidance to complete a task. Efficacy of the proposed mechanism is tested on documents collected from three different domains and test results are presented.

Keywords: Procedural question · Clause identification · Question generation

1 Introduction

Task oriented virtual agents aim at potentially automating a wide range of processes within the business. A few of the common examples of such agents include interactive self-service assistance to customers, knowledge assistance to internal help desk, guided selling or personalized guidance to portal navigation [1]. Most of these agents handle only quick, one-off advisories often leading to unsatisfactory resolution of an issue or incomplete assistance to obtain the goal. Service encounters are critical to an organization's image and therefore central to determining the success of the business. Modern task oriented virtual agents try to satisfy personalized constraints. Many service tasks relate to "How to" procedures and need assistance in an interactive manner.

This paper presents an enabling mechanism to support guided response in English in performing day-to-day customer facing operations; and to enable self-service for the customers. A guided response system demands the knowledge repository to be available in a certain form. This is often difficult to attain as such information contains both structured and unstructured data; some of them are client specific, some may be related to internal promotions, products and processes as well as installation, troubleshooting or operational manuals. Answering procedural questions from such disperse and unstructured knowledge sources would require a well-formed text structure. Such texts

E. Métais et al. (Eds.): NLDB 2020, LNCS 12089, pp. 73–81, 2020.
https://doi.org/10.1007/978-3-030-51310-8_7

can be a simple, ordered list of instructions to reach a goal, but mostly they are less linear, outlining different ways to realize a goal, with arguments, conditions, hypothesis or preferences. Hence, information collected from the reference documents must be organized to demarcate each procedure and stored with the corresponding sequential sets of instructions in a form to enable user interaction at intermediate levels. The contexts and conditions those determine the right path to complete the task are collected through interactions. In this work, a tree type structure is generated for each procedure description. The interaction points are the decision nodes of these trees. A conventional tree traversal mechanism is adopted to collect user inputs at every decision node.

2 Procedural Question Answering State of the Art

Recent advances in deep neural networks, language modeling and language generation have introduced stronger capabilities to the conversational agents. However, research on procedural question answering are still restricted to successful identification of procedures or responding to one-off questions. A preliminary structure of a model based on conceptual and linguistic analysis of procedural texts by simple text grammar system in French language was proposed by Delpech [2]. The issues of title identification, tagging and reconstruction via a learning mechanism in a large variety of procedural texts have been addressed by Adam et al. [3]. The challenges of answering procedural questions from procedural text have been investigated by Saint-Dizier [4]. Parsing and analyzing argumentative structures in procedural texts have been addressed successfully by Fontan [5]. Recent work on question answering tasks involving procedural text uses artificial neural networks [6]. Many times, these neural models rely on surface cues alone to make inferences. Such approaches also lead to domain-specific language systems that do not gracefully extend to other tasks. The novel work of Ribeiro et al. [7] allows semantic interpretation of procedural texts for answer retrieval and finding a single response from a procedural passage. Benchmark datasets like bAbI [8], SCoNE [9] and ProPARA [10] have also been created, but they mostly serve the purpose of procedural text comprehension and are not suitable for guided response.

3 Overview of the Proposed Mechanism

Our work first identifies the procedural passages in operational manuals and then we represent the information in the procedural passages into a tree form where the decision nodes provide the scope of interaction and the response nodes contain the advisory. When a conversation is initiated, the agent selects the appropriate process tree and traverses the right branch to generate the sequential set of instructions.

There are three major components in the whole mechanism as shown in Fig. 1. There is a Text Classifier, a Tree Generation Mechanism and an Interactive Chatbot mechanism. The first two components are required to transform an operation manual to a tree representation of the procedural passages. The Interactive Chatbot component

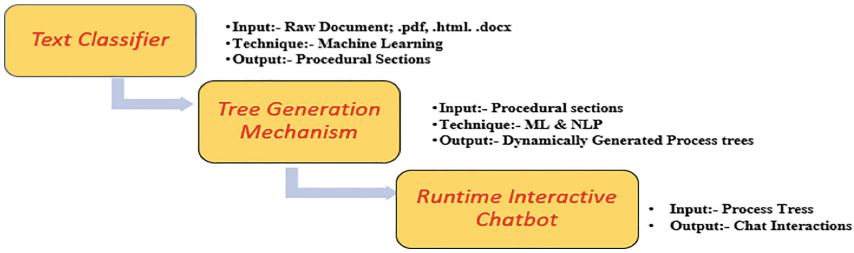

Fig. 1. Schematic representation of the components

works runtime. The Tree Generation Mechanism is the central component of the entire architecture and has five subcomponents; viz., Decision Point Identifier, Clause Identifier, Question Generator, Answer Path Identifier and Tree Structure Representation.

Detailed modeling of each of these subcomponents and the associated experimental results to substantiate the model efficacy are presented in the succeeding section.

4 Detailed Modeling and Experimentation

We have used three practical datasets from three different domains for experimentally developing the components. The datasets were manually curated to create a near equal distribution for procedural and non-procedural paragraphs. The first one is a FAQ document dataset that contains frequently asked questions available on Fitbit [11] and DLink [12] websites. About 900 paragraphs of IT domain specific questions and answers were obtained by web scraping. The second one is Insurance document dataset which contains 650 paragraphs related to banking and insurance domain. The QA pairs were obtained from Insurance document published by Bank of America [13]. The third one is Industrial document dataset which contains 500 paragraphs related to manufacturing and maintenance of industrial equipment published by Jackson [14].

4.1 Text Classifier for Identifying Procedure(s)

Operational manuals usually contain large volume of information out of which only a limited number of sections describe a procedure. Early research [15] to generate responses to "How" and "Why" kind of questions described a variety of rational structures of procedural texts. However, diverse argumentations in this type of texts have been observed and many other subtle language forms like tonality, opinion marks or injunctive forms are to be understood to create a linguistic model for procedure identification. Linguistic rules often do not discriminate between a process and a causality description. Understanding the boundary of the procedural knowledge; especially when the procedure cue has an ambiguous start is also challenge.

We propose simple binary classifiers to identify procedural sections in a document. The beginning and end of each section are the corresponding beginning and end of a

paragraph, title, headings or subheadings, bullet lists or any other marker that might indicate a seclusion. First, a conventional Random Forest (RF) classifier [16], with following two different sets of features was used for study and experimentation.

- Linguistic rule driven features like number of sentences starting with a verb, number of sentences without a subject, average length of steps in a procedure, number of infinitive verbs, number of gerunds, number of co-reference clusters etc.
- Fast-text word representation features including vectors for text of the lists, section marks, titles and subtitles wherever available.

We also built a Long Short Term Memory (LSTM) deep neural network [17] and trained it with glove word embeddings as the input feature. Comparison of the training and test accuracy of the models is shown below in the Table 1. Random forest classifier using only linguistic features was weak since the documents were not written in uniform styles and formats. The results indicated a far better performance of the LSTM model. Subsequently, complete validation of the LSTM model was done on the 3 datasets mentioned in the beginning of this section. An average F1 score of 89% was obtained during validation for all these 3 practical datasets.

Table 1. Comparative training results for text classifiers

Model	Training data	Training accuracy	Testing data	Testing accuracy	Validation data	Validation accuracy
LSTM	Fitbit	91.34	Fitbit	89.76	DLink	73.56
LSTM	Fitbit+DLink	94.60	Fitbit+DLink	92.40	DLink	84.56
RF FastText	Fitbit	80.23	Fitbit	76.89	DLink	58.40
RF FastText	Fitbit+DLink	83.45	Fitbit+DLink	77.34	DLink	65.50
RF Linguistic	Fitbit	61.45	Fitbit	56.76	DLink	54.89
RF Linguistic	Fitbit+DLink	63.45	Fitbit+DLink	61.90	DLink	58.39

4.2 Identifying Decision Points

Decision points are the instructions where user inputs are required to find the next steps as alternative paths are found at these points. The dependency parser Spacy [18] was used to understand the linguistic structure of the sentences, like the Part of Speech (PoS) and Dependency (DeP) tag. Whether the sentence is a decision point can then be identified programmatically. In Fig. 2 below, we show the tags for a simple sentence: *"If using windows 10, download latest version"*. Rules related to the conditional word "if" as starting conjunction (sconj) and the location of "advcl", can identify this as a conditional statement and a decision point. The PoS and DeP tags however, often make all acknowledgement points too as decision points. As shown in Fig. 3, for a sentence *"After completion of download, shutdown server",* no decision is required although the "advcl" is present. This is a mere acknowledgement point. Additional grammatical rules were created to address such scenarios. Additional rules are also used to consider more sparsely used conditional words like "check", "ponder" and "ensure".

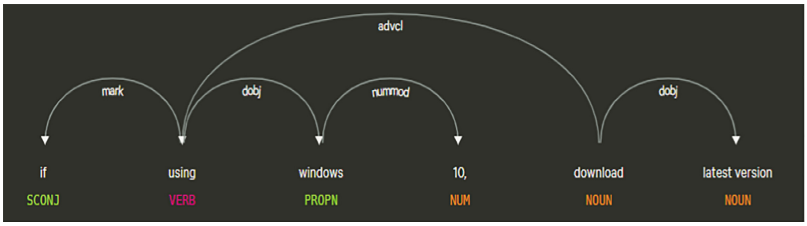

Fig. 2. Parser output for a conditional sentence as a decision point

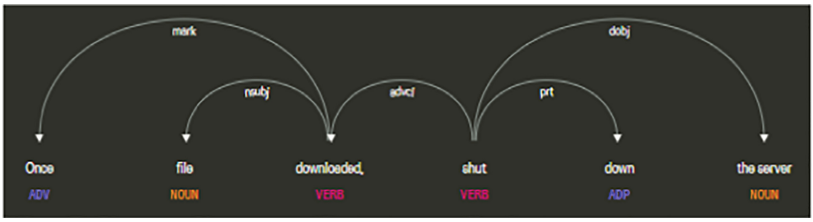

Fig. 3. Parser output for a conditional sentence as an acknowledgement point

The accuracy of decision point identification was enhanced by close to 6% with these additional rules. The accuracy is defined as the ratio of total number of decision points correctly identified to the total number of decision points identified by human referees.

4.3 Clause Identification

It is necessary to convert the conditional decision sentences into a question to enable collection of contextual information from the user. The question is based on the condition and the condition is contained in the independent clauses. Dependent clauses point to the next set of actions depending on the answer received.

A BiLSTM-CRF model [19] was used for clause identification. It has been shown that such a model can efficiently handle input features thanks to a bidirectional LSTM component. The CRF layer helps leveraging sentence level tag information. We used a published pre-trained model trained on CoNLL-2001 Shared Task Data [20] which contains 7150 sentences and 211727 tokens. This model computes the score of a sentence $[x]_1^T$ along with a path of tags $[i]_1^T$ as the sum of transition scores and network scores.

$$s\left([x]_1^T, [i]_1^T, \tilde{\theta}\right) = \sum_{t=1}^{T} \left([A]_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t}\right) \tag{1}$$

This sequence tagging model identifies the clauses by maximizing the score $s\left([x]_1^T, [i]_1^T, \tilde{\theta}\right)$. It recognizes clause starts, clause ends and identifies complete, possibly

embedded, clauses. An elegant improvisation was a subsequent use of the Spacy parser to identify the "first" and the "innermost" clauses. For the conditional sentence: *"If you previously downloaded playlists, skip to part 2 to download new music and podcasts"*, the dependent and independent clauses can be separated as shown below.

(S1 if
(S2 you previously downloaded playlists, E2 I)
(skip to part 2 to download new music and podcasts E1 D))

Where D and I denote Dependent and Independent respectively.

4.4 Question Generation

This module generates questions the agent will pose to the user at a decision point. Independent clauses contain the condition to generate a question. We examined three approaches for question generation; viz. Pattern-based [21], Aspect-based [22] and Data Driven Models (QGNET) [23]; for their suitability in the present context. We performed the aspect and pattern mining on the identified decision points. "Aspect" is extracted from the positive answer which is the dependent clause. "Keywords" are extracted from the independent clause with respect to the answer. In pattern-based approach, "Keywords" are used to generate the "Patterns". Then a sequence-to-sequence model with BERT pretrained embeddings is trained for question generation. In aspect-based approach, "Aspect" and "Patterns" are encoded separately and used as inputs for training this sequence-to-sequence model. QGNET is a data-driven approach built on a sequence-to-sequence model with copy mechanism. This model was originally trained on SQuAD dataset with answer tokens and other linguistic features like PoS and Named-Entity-Recognition (NER) tags. The pre-trained model of QGNET generates a "wh" type of factoid question. In our work, we required binary type of questions as otherwise the user would have a potentially open-ended choice of

Table 2. Comparative study of quality of generated questions

Sample conditional statement	If you're using the latest version of Fitbit connect, click the Fitbit connect icon located near the date and time on your computer			
Ideal question	Does your computer use the latest version of Fitbit connect?			
	Pattern based	Aspect based	QG-Net (Pretrained)	QG-Net (Retrained)
Prediction	Pattern:- Does this device## Question:- Does this device update to latest version?	Aspect:- using latest version Question:- Can this device be used with latest version?	Question:- What is the version of Fitbit connect?	Question:- Are you using latest version of Fitbit connect?
Cos- similarity with ideal question	0.47	0.59	0.62	0.89

responses that could not be used. So, we further retrained the QGNET model with AQAD dataset [24]. We selected 200000 Boolean type of questions from AQAD; and 765 question and answer pairs from 300 procedural paragraphs in the FAQ dataset for training.

Table 2 above provides a snapshot of a qualitative comparison of the questions generated by all three methods. The questions generated by QGNET were more usable in the current context. This was probably owing to the flexibility available to retrain QGNET with specific type of question answer pairs as desired.

5 Implementation of Guided Response System

During implementation, we first train offline all the modules discussed in Sect. 4. These modules enable programmatic creation of the procedural trees from a given document. Each procedural tree is assigned a label. At the beginning of a conversation, an intent matching mechanism selects the correct tree. Each tree contains two types of nodes; viz. Information Seeking node and the Response node. The information seeking nodes may have multiple paths tagged with different possibilities which are prompted to the user for tree navigation. The response type nodes either provide next step or stop if a resolution is obtained. If a resolution is not attained till the leaf node in the tree, the system may involve a human agent for further conversation. For procedures described over multiple paragraphs, each of the paragraphs are identified as a sub-procedure and the task completion is possible only after all the trees are traversed. Usually a dialog manager will have to be integrated to manage the conversation and keep the conversation grounded to the end goal; however, this was not investigated in the current research. In a laboratory setup, more than 90% of the user queries, from approximately 700 procedures collected from the three datasets mentioned before were successfully responded; and led to a close to 50% of efficiency improvement in the business process.

6 Conclusion

We presented here an enabling mechanism for conversational agents to handhold the user through a step by step execution of tasks. Most of the automated agents today respond to the how-to questions in a passage form that often leaves the user to complete the task inefficiently or even fail to complete it altogether. The purpose of the work was to create a means to convert the available knowledge into a form that would aid the agent respond in a staggered, yet, contextual manner. We have seen that fully machine learned models can distinguish a procedural part of text from the rest and we can leverage the linguistic capabilities of the modern-day parsers and state of the art deep learning mechanisms to create a pre-scripted dialog that can be availed runtime. The strength of QGNET in generating questions was found useful. The method was found suitable for at least three different domains of applications.

References

1. Verhagen, T., Feldberg, F., Nes, J.: Virtual customer service agents: using social presence and personalization to shape online service encounters. J. Comput. Mediat. Commun. **19**(3), 529–545 (2014)
2. Delpech, E., Saint-Dizier, P.: Investigating the structure of procedural texts for answering how-to questions. In: Proceedings of the Language Resources and Evaluation Conference (2008)
3. Adam, C., Delpech, E., Saint-Dizier, P.: Identifying and expanding titles in web texts. In: Proceedings of the 8th ACM Symposium on Document Engineering, pp. 213–216 (2008)
4. Saint-Dizier, P.: Some challenges of advanced question-answering: an experiment with how-to questions. In: Proceedings of the 22nd Pacific Asia Conference on Language, Information and Computation, pp. 65–73 (2008)
5. Fontan, L., Saint-Dizier, P.: Analyzing argumentative structures in procedural texts. In: Nordström, B., Ranta, A. (eds.) GoTAL 2008. LNCS (LNAI), vol. 5221, pp. 366–370. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85287-2_35
6. Das, R., Munkhdalai, T., Yuan, X., Trischler, A., McCallum, A.: Building dynamic knowledge graphs from text using machine reading comprehension. In: Proceedings of the International Conference on Learning Representations (2018)
7. Ribeiro, D., Hinrichs, T., Crouse, M., Forbus, K.: Predicting state changes in procedural text using analogical question answering. In: 7th Annual Conference on Advances in Cognitive Systems (2019)
8. Weston, J., et al.: Towards ai-complete question answering: a set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698 (2015)
9. Long, R., Pasupat, P., Liang, P.: Simpler context-dependent logical forms via model projections. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pp. 1456–1465 (2016)
10. Du, X., et al.: Be consistent! Improving procedural text comprehension using label consistency. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis (2019)
11. FitBit. https://help.fitbit.com/?l=en_US&c=Topics%3AFAQs. Accessed 03 Feb 2020
12. DLink. https://eu.dlink.com/uk/en/support/faq. Accessed 03 Feb 2020
13. Bank of America. https://www.bankofamerica.com/deposits/resources/personal-schedule-fees.go. Accessed 03 Feb 2020
14. Jackson. http://manuals.jacksonmsc.com/ecolab%20manuals/ES-2000%2&%20ES-4000%20Rev%20O.pdf. Accessed 03 Feb 2020
15. Auladomour, F., Saint-Dizier, P.: Towards generating procedural texts: an exploration of their rhetorical and argumentative structure. In: Proceedings of the 10th European Workshop on Natural Language Generation (2005)
16. Breiman, L.: Random forest. Mach. Learn. **45**(1), 5–32 (2001)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
18. Spacy. https://spacy.io. Accessed 03 Feb 2020
19. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991v1 (2015)
20. Tjong Kim Sang, E.F., Déjean, H.: Introduction to the CoNLL-2001 shared task: Clause Identification. In: Proceedings of CoNLL-2001, Toulouse, France (2001)

21. Duan, N., Tang, D., Chen, P., Zhou, M.: Question generation for question answering. In: Proceedings of the EMNLP 2017, pp. 866–874 (2017)
22. Hu, W., Liu, B., Ma, J., Zhao, D., Yan, R.: Aspect-based question generation. In: Proceedings of the ICLR Workshop (2018)
23. Wang, Z., Lan, A., Nie, W., Waters, A.: QG-net: a data-driven question generation model for educational content. In: Proceedings of the 5th Annual ACM Conference on Learning at Scale (L@S), pp. 1–10. ACM (2018)
24. Amazon question/answer data. http://jmcauley.ucsd.edu/data/amazon/qa/. Accessed 03 Feb 2020