DOC2BOT: A Document Grounded Bot Framework

Kshitij Fadnis, Pankaj Dhoolia, Li Zhu, Q.Vera Liao, Steven Ross, Nathaniel Mills, Sachindra Joshi, Luis Lastras
IBM Research
{kpfadnis, zhul, steven_ross, wnm3, lastral}@us.ibm.com
vera.liao@ibm.com, {pdhoolia, jsachind}@in.ibm.com

Abstract

Conversational agents – or chatbots – are widely used to provide customer care and other informational support. Currently, the development of chatbots using standard frameworks requires a lot of manual crafting by subject matter experts (SMEs). On the other hand, while learning-based approaches to dialog have made significant advancements, they require training with a large volume of dialog data, which chatbot developers typically do not have access to. To tackle these challenges, we introduce DOC2BOT, a system that supports the automated construction of chatbots by digesting various forms of documents such as business manuals, HowTos, and customer support pages that organizations own. In addition to this, DOC2BOT provides a user-friendly experience to SMEs, and minimizes the effort expended by them by supporting intuitive interactions and streamlining their workflow.

1 Introduction

Automated conversational assistants – or chatbots – have become prevalent in customer care settings for tasks such as question answering, troubleshooting, transactional activities[1]. Even when information already exists on websites, customers may still prefer chatting with a bot[2] in lieu of navigating the dense complexity of the website. However, developing chatbots remains a painful process, especially as bot development frameworks move beyond simple intent and entity matching models. For example, popular frameworks like *IBM Watson Assistant*[3], *Google DialogFlow*[4], *Microsoft Bot Framework*[5], and *Amazon Lex*[6] provide richer Natural Language Processing (NLP) and dialog management capabilities. However, one big challenge still remains: content creation for chatbots still largely relies on either manual crafting, or on the manual conversion of existing information sources like product documents, manuals, FAQ webpages etc. into conversation models. This dependency leads to two major challenges. First, businesses need dedicated personnel or subject matter experts (SMEs) with the right skill-sets for such tasks; even with varying degrees of tooling help, significant hand-holding is still required. Second, businesses also need to commit significant time and effort into keeping the conversational models up-to-date with changing business contexts.

To tackle these challenges, we present DOC2BOT, a novel framework to build and update conversation assistants by automatically processing business documents. The key capabilities of DOC2BOT are:

1. Write Once. Use Everywhere: DOC2BOT consumes business content directly, and business content changes can easily trickle down to conversational models in an automated fashion.
2. Content Inspired Conversation Model: DOC2BOT utilizes the inherent structure in documents to automatically extract conversation models that support conversational experiences such as guided topic exploration and walk-down of procedures.
3. Conversation Model Enhancement and Customization Made Easy: When automatically extracted conversation models need to be improved or customized, DOC2BOT supports human-in-the-loop learning (details in Section 2.2).
4. Advanced Natural Language Understanding (NLU): NLU built with USE (Cer et al. 2018) provides the ability to detect multiple intents in a single user utterance (details in Section 3).
5. Declarative Dialog Manager: Orchestrates multi-turn conversations across multiple documents and supports digressions, fall backs, disambiguation, and confirmations.

In the following, we present the technical details of DOC2BOT in the phases of document ingestion, conversation model building, and deployment. An overview of the overall workflow is provided in Figure 1.

2 Ingestion Phase

Business documents like manuals and self-help webpages commonly have information laid out in a structured fashion in order to help the reader consume the content better. Examples of such structure include headers, bullet-point lists, how-to procedures, etc. Consuming such structural markers, DOC2BOT can produce a custom bot that walks a user

[1]https://www.revechat.com/blog/chatbots-use-cases/

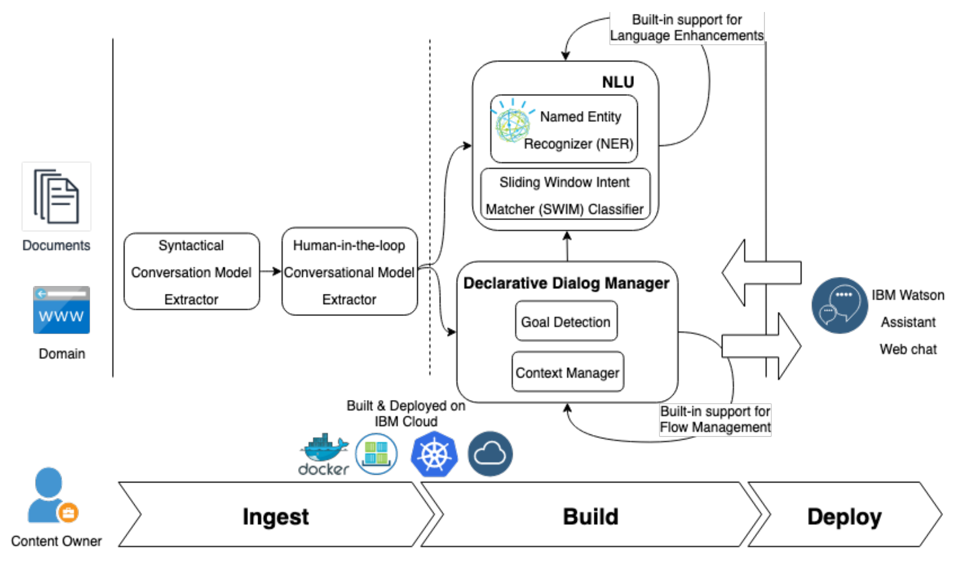
[2]https://www.cbinsights.com/research/report/most-successful-chatbots/

[3]https://www.ibm.com/cloud/watson-assistant

[4]https://cloud.google.com/dialogflow

[5]https://dev.botframework.com

[6]https://aws.amazon.com/lex



Built-in support for Language Enhancements
NLU
Named Entity Recognizer (NER)
Sliding Window Intent Matcher (SWIM) Classifier
Documents
Domain
Syntactical Conversation Model Extractor
Human-in-the-loop Conversational Model Extractor
Declarative Dialog Manager
Goal Detection
Context Manager
IBM Watson Assistant Web chat
Built & Deployed on IBM Cloud
Built-in support for Flow Management
Content Owner
Ingest
Build
Deploy

Figure 1: DOC2BOT Workflow; Video: ibm.biz/doc2bot

through procedures in a step-wise fashion. We highlight two major components of the ingestion phase.

2.1 Syntactical Conversational Model Extractor

This module extracts a hierarchical section structure from a document; with each section being associated with a name, content, location offsets, and possibly child sections (either to be explored as sequential steps, or as random choices on demand). Since our system works with HTML documents, we use HTML's in-built tags to identify section boundaries. Child sections are categorized as choice-points or 'choices' by default; the only exception is sections encapsulated within the ordered list (<ol>) tag, which are uniquely categorized as sequential 'steps' rather than choices.

2.2 Human-in-the-loop Conversational Model Extractor

The syntactic conversation model extractors outlined in Section 2.1 may fail from time to time. In such scenarios, DOC2BOT relies on a human-in-the-loop approach that allows guidance from SMEs to customize the extraction process to their specific domain. DOC2BOT's human-in-the-loop conversational model extractor starts by grouping documents that visually appear similar. While grouping documents, we ignore the content of the documents, and represent each document using a set of features extracted from its DOM tree structure[7]. We then use an agglomerative clustering implementation from `scikit-learn` (Pedregosa et al. 2011) to cluster the documents. Documents in the same cluster represent a unique set of section structures, which in turn maps to a unique set of conversation templates/patterns applicable to those documents. At this point, SMEs can work with an exemplary document from each cluster, and provide guidance to extract additional section structures via simple *drag & select* interactions if necessary. DOC2BOT learns from these interactions and, with the SME's confirmation, generalizes and applies enhanced extraction logic to other documents in the cluster. This module thus allows SMEs to identify and fill gaps in the conversation model at a rapid pace by working with a small number of exemplary documents.

[7]https://dom.spec.whatwg.org/

3 Build Phase

Once a conversation model is acquired, DOC2BOT's runtime retains this structure, where a section unit extracted from a document is referred to as an 'action', and a precondition for each action is indicated as a 'trigger'. Along with actions and triggers, DOC2BOT's runtime has built-in template-based conversational patterns such as *disambiguation* or *confirmation*.

Natural Language Understanding (NLU) In conversational systems, the natural language understanding (NLU) unit is used to decode user utterances in terms of intents and entities. State-of-the-art NLU approaches focus on associating a single intent to an entire user utterance. However, it is known that users often express multiple intents in a single conversation turn. DOC2BOT adopts a variable size (word lengths) sliding window approach over the user utterance; classifies each of those windows; and filters out candidate classes with scores below a configured threshold. It then computes a window-length weighted sum score for non-overlapping candidates and return those with the top-n weighted sum scores as multi-intent classification results.

Declarative Dialog Manager A basic conversation plan is built by evaluating the preconditions of an action, and responding with the action's content if that preconditions are triggered. If there are child actions, the first action is triggered (if the child-policy consists of steps); or alternately, the user is prompted to pick a choice from the children (if the child-policy consists of choices). We refer to top level sections as ‘goals’. At the beginning of a conversation, the user is expected to say something that matches the precondition of a goal. Similarly, when the conversation has proceeded to an action with a choice of child actions, the user utterance is matched with the precondition of one of the children. When the conversation is at an action that is a step, the user will have to say something to indicate the completion of that step, or the intention to go to the next step. If the user’s utterance triggers multiple actions, the runtime presents choices to disambiguate.

Oftentimes, knowledge from multiple documents is needed to hold a conversation with a user: a user may initiate a conversation related to a process goal; or, in the context of a specific step, a user may ask something that is detailed in another document and hence is a part of another goal. A key feature of our runtime is that it evaluates every user utterance for preconditions associated with all the goals to check if there is a goal-switch intended. Our runtime maintains the context of the conversation up to the current turn in the form of a goal-stack, allowing it to not only stitch conversations across goals dynamically but also to return to the appropriate point in the previous or main goal once the current goal on stack has completed.

4 Conclusion

We demonstrated DOC2BOT, a system that enables businesses to rapidly build and deploy bots built on their content with minimal human effort. In the future, we plan to incorporate linguistic clues along with structural clues to improve conversation model extraction.

References

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* .

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.