VISUALIZATION, CONFIGURATION AND AUTOMATED TESTING OF A NATURAL LANGUAGE PROCESSING APPLICATION FOR CONVERSATIONAL COMMERCE – A CASE STUDY

Ramkumar R.

Principle Architect, Bahwan Cybertek, Chennai, India

Iyer.r@bahwancybertek.com

Abstract- *Conversational commerce is the technology of interacting with financial services like order booking services, banks or brokerages. A plethora of Natural Language Processing based chatbots have evolved to solve this problem. We discuss the building components of such an application for a large financial institution especially with regards to the visualization, configuration and automated testing of the backend solution for the chatbot. This paper will be useful to system engineers who want to build conversational commerce applications that are highly scalable and extensible as well as architects who are interested in testability and manageability of such systems.*

Keywords—Artificial Intelligence, Natural Language Processing, Configuration, Visualization, Testing.

INTRODUCTION

Conversation commerce is a system where users can chat with messaging applications to pursue a wide variety of business needs like banking, order booking or shopping in an automated fashion without human assistance as much as possible. Past work has addressed such systems in M-Commerce [19, 21] or Recommender Systems [20] which has been use case specific. In this paper we look at the important dimension of the architecture and modules that are needed for such systems to be built and operated successfully based on a live use case of a major bank.

SYSTEM ARCHITECTURE

The conversational commerce system, targeted at banking was based on recognizing the user intent behind every conversation utterance. For example the intent behind an utterance: "What is my account balance?" would be "ACCOUNT_BALANCE" leading to triggering of

an appropriate dialog workflow. Multiple user intents behind each conversation utterance were not supported. A content management system and integration with external and internal services was used to provide appropriate responses. The system consisted of four important subsystems.

Subsystem1 dealt with the core intent detection logic based on a combination of machine, text based pattern matching, NLP based intent detection and ontology based detection.

Subsystem2 dealt with the workflow, content management, legacy/ new/ external service interaction and the adaptation of the interface to different endpoints like smart speakers or Mobile App or Web.

Subsystem3 dealt with a configurable simulator used by internal staff to simulate the conversations that will happen in real time and record them

Subsystem4 dealt with the visualization, configuration, and testing and release management of the entire system but especially subsystem 1**. In this paper we will focus mainly on Subsystem 4.** This is a key contribution of this paper as this area has not been addressed by past literature.

The interconnections between the systems are illustrated below:

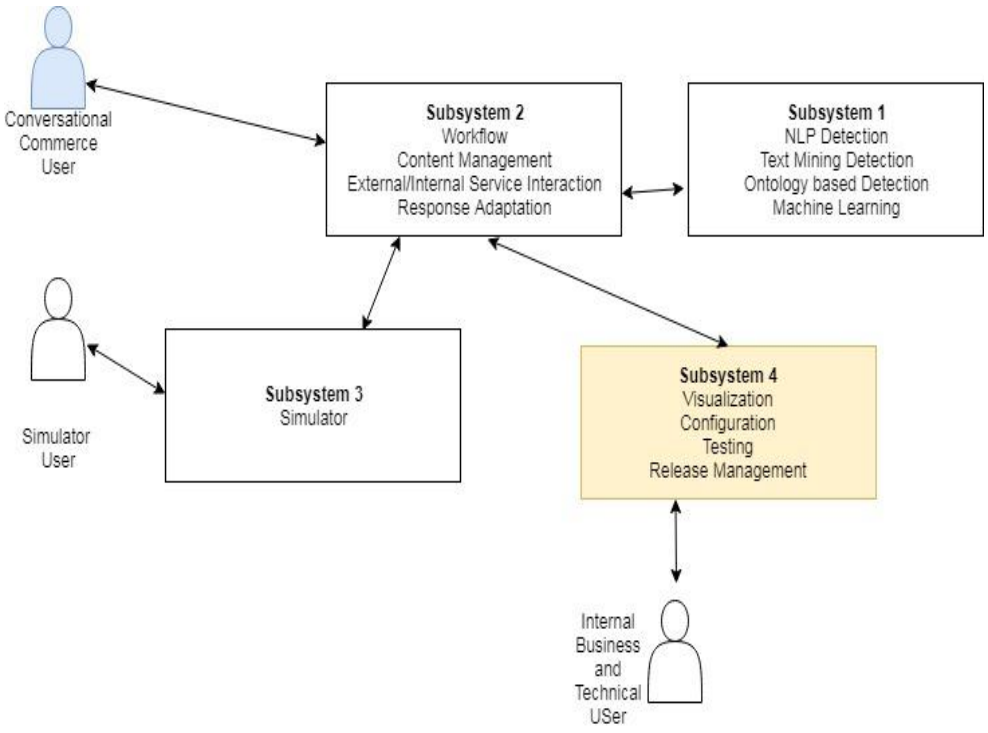

Fig 1: Subsystems and interconnections

The architecture was a 3-tiered architecture with different subsystems communicating via REST Services within each other and with each other. The frontend was a ReactJS [6] based Single Page Application, middleware was RESTful web services [7], application logic was using Spring MVC [8] and database was in Cassandra with Datastax [9, 17]. Major NLP software that was used is WordNet [10], Stanford Core NLP[11] and Solr [12]. Development languages were Es6 [13] and Java [14]. Development happened in Agile Methodology [15] and team size was approximately 30 people.

SUBSYSTEM 4 COMPONENTS

1. Ontology visualization
2. Entity detection
3. Workflow management
4. Abuse Management
5. System Diagnostics
6. Intent detection and Content Management
7. Job management
8. Release management, Authorization and Security
9. Automated Conversation testing and Regression
10. Supervised Feedback Tool
11. Dictionary Management
12. Bulk Data Management

A. Ontology Visualization

The domain concepts like Account, ATM, Credit Instrument and their internal relationships was specified using RDF (Resource Description format) and constantly queried and updated by business users. For this purpose, a tool called WebVoWL was integrated to the frontend [1]. This proved very useful for the business purposes. For display purposes, only 50 concepts were displayed at a single time.

B. ***Entity Detection***

One of the major tasks of NLP is to detect entities. The list of entities was made editable with provision to add new entries using table visualization.

C. ***Workflow management***

Internally, whenever a chat conversation response was detected, a workflow was initiated. The workflow consisted of calls to NLP steps like Named Entity Recognition; calls to content management system for uses like translation as well as calls to external and internal services for dialing a number or getting a bank balance respectively. The technical team used JSON to represent the workflow. This was visualized and made editable using the JointJS framework [2] which was freely available. Icons and colors were used to designate the various steps in the workflow as well an ability to see the list of workflows was provided. The workflow tool enabled us to save the layout of the workflow to the database after any edits.

D. ***Abuse management***

This was a simple user editable table where abusive words could be added to be recognized by NLP and generic responses were provided in the conversation.

E. ***System Diagnostics***

The NLP process used 4 major ways of detecting user intent and selected the matching user intent based on maximum confidence of a given chat utterance. They were machine learning, ontologies, Core NLP based logic and search based solutions using Solr. This is often called ensemble learning [4]. A tool was provided in the frontend to view the call stack and predictions of these components for debugging purposes.

F. ***Intent Detection and Content Management***

A simple table where a user could tag multiple preconfigured or new intents for each utterance for training the NLP was provided. Users could also provide the content identifier or create content that was used in the response.

G. ***Job Management***

A simple job management interface where the user could specify the training schedule for various NLP tasks and machine learning model. It was executed using the AutoSys scheduler. [5]

H. ***Release Management, Authorization, Security***

During various stages users could add intents, utterance or content to name a few which would cause changes in the system. The Release management provided an interface for the supervisor to approve or reject these changes along with a rationale to add a layer of protection. The authorization layer involved a user name and password over HTTPS which will give specific permissions mainly on Release Management but it could be extended to other modules.

I. ***Automated Conversation Testing and Regression***

Automated conversation testing allowed for the recording and rating of simulated conversation as compared to individual utterances. These conversations can then be run as a regression suite that can validate the system performance over time.

J. ***Supervised Feedback Tool***

This is a critical component which could display the real time user utterance from a live instance as well as the intent the system had detected. Then a supervisor could give a feedback on the accuracy or even suggest a better intent for the utterance. The key feature of this component was the auto-prioritization of an utterance to be analyzed over millions of other utterances through a flexible model that involved metrics like step time, recentness, frequency or conversation quality to name a few.

K. ***Dictionary Management***

This provided a simple tabular interface to search words, add words, and define their lemmatized versions. It also included facilities to provide syn-sets. There was also a coref resolution page where corefs could be identified for specific utterances to train the model.

L. ***Bulk Data Management***

Many of the modules like Content Management, Entity Management and Dictionary Management supported a mode for Bulk Upload using CSV files for ease of use. This was solved generically.

LEARNINGS

It is difficult to come up with a perfect solution to visualize, configure and test a Conversational Commerce System. Often the requirements are very dynamic and changing as

this is a newly emerging domain. Hence, it is better to create a base framework in terms of architecture, design and scaffolding of the system but only build modules on a need basis. This list of modules that were built gives an overview of probable requirements rather than exhaustive list. Often, a test module needs to be created and taken to the business users or core technical developers by the team to see if they benefit from the ensuing automation before building it out completely. It is also important to keep a consistent look and feel as well as design across these modules since the end users may access these modules in an integrated manner to achieve a purpose. For example a user may use the supervised feedback tool page to create a new intent in the intent page and further add content against the intent to train the model.

FUTURE WORK

One of the issues faced during development of the system was that the backend modules like Database need to be up and running to do our work which may be quite independent of the database components. Hence there was a need to build a simulated backend which is often called a Mock Object [18] to develop independently of backend.

Further, these modules were developed for activities to be done on a laptop/desktop and addressing the mobile requirements in future may be a key challenge as the adoption of the conversational commerce application increases.

REFERENCES

[1] Lohmann, S.; Link, V.; Marbach, E.; Negru, S.: WebVOWL: Web-Based Visualization of Ontologies. Proceedings of EKAW 2014 Satellite Events, LNAI 8982, pp. 154-158, Springer, 2015.

[2] https://www.jointjs.com/

[3] http://lucene.apache.org/solr/

[4] AMIA Annu Symp Proc. 2017 Feb 10;2016:1880-1889. eCollection 2016. Ensembles of NLP Tools for Data Element Extraction from Clinical Notes. Kuo TT(1), Rao P(2), Maehara C(3), Doan S(1), Chaparro JD(1), Day ME(1), Farcas C(1), Ohno-Machado L(1), Hsu CN(1).

[5] http://www.dbaref.com/autosys-jobs

[6] https://reactjs.org/

[7] Roy Thomas Fielding. 2000. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation. University of California, Irvine. AAI9980887

[8] http://cassandra.apache.org/

[9] https://spring.io/

[10] https://wordnet.princeton.edu/

[11] https://stanfordnlp.github.io/CoreNLP/

[12] http://lucene.apache.org/solr/

[13] http://es6-features.org/

[14] https://java.com/

[15] https://www.agilealliance.org/agile101/

[16] https://en.wikipedia.org/wiki/Single-page_ application

[17] https://www.datastax.com/

[18] "Interaction testing with mock objects et seq". The art of unit testing. Manning. ISBN 978-1-933988-27-6

[19] Siwar Rekik, Sid-Ahmed Selouanr, and Habib Hamam. 2009. A cooperative and conversational virtual agent for M-commerce applications. In Proceedings of the 6th international conference on Innovations in information technology (IIT'09). IEEE Press, Piscataway, NJ, USA, 221-225.

[20] D. H. Widyantoro and Z. K. A. Baizal, "A framework of conversational recommender system based on user functional requirements," *2014 2nd International Conference on Information and Communication Technology (ICoICT)*, Bandung, 2014, pp. 160-165.

[21] Mobile conversational commerce: messenger chatbots as the next interface between businesses and consumers. Author: Milan van Eeuwen. University of Twente