Conversational Analytics using Apache Spark: Acquisition of user engagement characteristics and providing recommendations of products and service

Aditya Wakade, Sujata Khedkar, Kinjal Chavda, Ishaan Malhi
Department of Computer Science,
VESIT, Chembur, Mumbai, India

Abstract

With the increasing popularity of development of chatbots to overcome restrictions of complex user interface design and inconvenience to the user in requesting data and services, we felt that there is a strong need for an analytics platform for chatbot developers through which they can implement analytics on the information collected by the chatbot to better understand the needs of the user. The paper tries to identify and find solutions for two types of analytics needed by the chatbot. One is to summarize the reviews and nature of a product or service from multiple sources on the internet. Other caters to capture and analyse user engagement characteristics to provide better recommendations.

Keywords: Conversational interface, chatter bots, spark, web crawling, text summarization, analytics, flume.

I. INTRODUCTION

A chatterbot is a computer program which conducts a conversation and often convincingly simulates how a human would behave as a conversational partner, thereby passing the Turing test. Chatterbots are being used for a variety of purposes such as customer service, feedback, analytics, product management, survey tools, financing, entertainment etc.

![Fig: Use cases from the botness survey][1]

Some chatterbots use sophisticated natural language processing systems, but many (such as those found on messaging platforms such as Slack, Facebook Messenger or HipChat) match the input string to a set of characters, i.e perform regular expression matching to execute a particular action. We initially explore the need for chatbot analytics, by providing a parallel to Fabric, Twitter's Mobile Analytics Suite[13] in order to bring a sense of urgency to the requirement of conversational analytics. We then explore the idea of context aware chatterbots, which are able to understand conversational context and take appropriate actions, based on user behaviour and preferences. This is followed by an overview of the current prevalent architecture of a general chatterbot, detailing the various parts of a conversational agent. Furthermore, modern chatterbots often rely on increasingly elaborate and innovative natural processing engines to parse user input and label parts of a sentence to extract meaningful information from it. The paper provides a summary of a number of current industry prevalent NLP training algorithms, their advantages and disadvantages and specific use cases. We then propose a system comprising of a conversational agent that responds to a user query for a product listed on Amazon and replies with a summarization of the reviews of said product. Over time, the system learns the user's preferences and engagement characteristics to understand user context. We describe the various modules of the system comprising of the chatterbot, the analytics module, the web crawler and the API.ai [11] external NLP engine. Our goal is to propose a system for real time (stream) as well as static chatbot analytics to provide an in-depth view of user engagement to a chatbot developer.

II. NEED FOR CONVERSATIONAL ANALYTICS

As of now, over 60,000 developers creating chatbots for fortune 500 companies. Bot building platforms like Api.ai, Botstack, Github's Hubot are being promoted as tools to easily create bots for non-technical personnel. Chatbot Analytics will allow these developers to review and understand user engagement patterns of their bots, in a manner similar to mobile app developers.

To understand the need for conversational analytics, we look at two industry relevant scenarios: the Botness survey, a survey buy a group of influential chat bot developers; and Twitter Fabric.

At present, Fabric is used by 250,000 developers, and installed on 1.325 out of 1.4 billion active Android devices. Twitter Fabric was released at a time when mobile development was about to reach it's peak, and as such the current scenario of chatbot development is the ideal time to release an analytics platform. A conversational analytics platform has the capability to have the same impact and utility as Twitter Fabric, if not more.
III. CONTEXT AWARE CHATTERBOTS

With the advent of Ubiquitous Computing, the need for context aware agents is growing. Agents that can understand and derive meaningful information from a user’s context such as weather information, spatial and temporal relationships, emotion state and character evaluation.

In large part this consists of the prior history of interactions by the agent with the user; coherent responses that may find the maximum relatability with a user will consists of the user’s real environment (temperature, wind speed, humidity) which can be captured by physical sensors and the virtual environment may include machine state, game state and application state.[8] This ideology is modeled after general social conversations in a group: a trusted friend (registered chabot) understands our preferences and propagates this understanding to others (new chatbot services) in the group.

Context aware chatbots will require a strong analytical backend, which may include the following among many: a user behavioural classification system, sentiment analysis system, recommendation system etc.

One of our supplementary goals is to develop a system for chatbots to understand context by referring to a global knowledge base that would contain user profiles and will therefore allow chatbots to use authorized information from each other. This will allow chatbot developers to provide a more personal user experience, and thus enable chatbots to be more relatable and useful, in effect proving them a viable alternative to mobile and desktop applications.

IV. ARCHITECTURE OF A CONVERSATIONAL AGENT

The general architecture of a chatterbot is as shown in the figure below. A user interacts with a chat interface, generally giving various commands through natural language (although, sometimes specialized commands have to be used). The knowledge engine steers the conversation into its relevant domain, which includes the various topics that a conversation may pertain to. Specific information about the topic is represented as context maps within the topic. The conversation engine is responsible for maintaining proper, meaningful conversations based on the current context.

V. NATURAL LANGUAGE PROCESSING TRAINING MODELS

A. Question and Answer Training with Subgraph Embeddings

Subgraph Embedding is a methodology to learn to answer questions on a broad range of topics from a knowledge base using few handcrafted features. The model learns low-dimensional embeddings of words and knowledge base constituents; these representations are used to score natural language questions against candidate answers.

The proposed system[7] uses an embedding model, which learns low-dimensional vector representations of words and symbols and can be trained with lesser supervision. This system has a more sophisticated inference procedure that is both efficient and can consider longer paths along with a richer representation of the answers which encodes the question-answer path and surrounding subgraph. However, the researchers only had access to the test predictions and used a combination method of 50% of their approach and 50% of an older approach. In summary, subgraph embeddings can be useful to train our NLP engine on Twitter data, although not altogether reliant, requiring us to look at other models.

B. Context-Sensitive Generation of Conversational Responses

The model consists of a response generation system that can be trained end to end on large quantities of unstructured Twitter conversations. A neural network architecture is used to address sparsity issues that arise when integrating contextual information into classic statistical models, allowing the system to take into account previous dialog utterances. The dynamic-context generative models show consistent gains over both context-sensitive and non-context-sensitive Machine Translation and Information Retrieval baselines.[8]

The model consists of continuous representations to estimate a probability function over natural language sentences. The researchers propose a set of conditional RLMs (Recurrent Language Models) where contextual information (i.e., past utterances) is encoded in a continuous context vector to help generate the response. The context vector is learned along with the conditional
RLM that generates the response. Additionally, the learned context encodings do not exclusively capture contentful words. Even “stop words” can carry discriminative power in this task; for example, all words in the utterance “how are you?” are commonly characterized as stop words, yet this is a contentful dialog utterance.

Probably the only disadvantage of this system is choosing which of three proposed models to use; though the scores are lower than those usually reported in classification tasks, the ranking of the three systems is ambiguous, and hence do not aid us in deciding our model.

C. Conversation engine for pragmatic semantics

This paper focuses on dealing with conversational situations in chatbots that usually revolve around a specific topic and go on longer than single sentences. The presented approach combines pragmatics and content semantics to generate conversations in the customer service domain.

The training conversations are first split into topics and types of queries[9]. Based on these terms, a query can be mapped with the correct answer to it, and a successful conversation can be generated. Various flow graphs enable the distinct ways to carry different types of conversations (informational, procedural, troubleshooting, dispute resolution etc.) The generated conversations are then evaluated by a panel of judges, and are evaluated by metrics based on Grice’s maxims.

The major drawbacks of this approach are that the evaluation conducted is manual and not automated. Also, incorporating richer knowledge representation and retrieval techniques will make the architecture work even with less situation specific contextual conversations. Richer modeling frameworks might be useful for modeling wider ranges of contexts.

D. Enable relations in chatbot responses

The authors of this paper have proposed an algorithm with which they aim to generate conversations which are able to switch between related conversation topics. They use a chatbot called the Virtual Diabetes physician (ViDi)[10] and a web application accessible to the developers in which every context can have a defined extension and prerequisite.

When a conversation proceeds through the chat interface, the context of the chat is checked against the stored database of contexts, extensions and prerequisites. When the conversations proceeds, at each step the context is checked and matched with the current one. If it doesn’t match, its extended contexts are checked for matches. If a match occurs, the chatbot switches to that context and the same process repeats till the end of the conversation.

The presented system has very limited scope, as it is limited only to one context of a chatbot. It can be extended to cover different types of conversations.

E. Extracting Dialog Models from conversation transcripts

Abstraction of tasks and subtasks from un-annotated man to man textual conversations and identifying patterns and clusters in them to infer facts about the nature, similarities, dissimilarities and priorities of different tasks is called extraction of dialog models. The procedure involves development of a small scale knowledge base using AIML (Artificial Intelligence based markup language). Using relations between simple AIML constructs, the paper suggests that we can structure a consistent knowledge base. This knowledge base is used as a reference by chatbots to analyse and reply to other conversational inputs.

The paper employs a rule-based named entity annotator that uses a set of dictionaries and rules to annotate various types of entities in the transcript data. The types of entities that the paper annotates are LOCATION, DATE, TIME, AMOUNT. It also handles a few domain specific named-entities such as CAR-MAKE (Chevrolet,Toyota), and VEHICLE-TYPE (car, van). All entities that are annotated in an utterance are replaced by their corresponding type.[6]

The algorithm used identifies non consecutive terms in text which regularly occur in patterns. The model used to identify and store such patterns in natural language expressions is based on inclusion of common words in a buffer and tracking their relation in graph with other common nodes. In case of multiple relations between nodes the longest sequence of common phrases is selected as the extracted pattern from the text. Moreover, identification of patterns and chronology of tasks is based on the subtasks and preconditions required.

Find $S_n$, the set of all l-item-sequences; $n = 1$
$S_n = \{\}
while $n \leq N$ do
$S_{n+1} = \{\}
for Each $s_n , s'_n \in S_n$ do
if $s_n$ and $s'_n$ have a subsequence of length $n - 1$ in common
then
Merge $s_n$ and $s'_n$ to obtain $s_{n+1}$
if $\sup(s_{n+1}) \geq \min\sup$ then
$S_{n+1} = S_{n+1} \cup \{s_{n+1}\}$
end if
end if
end for
for Each $s_n \in S_n$ do
if $\{s_{n+1} \in S_{n+1} \mid s_{n+1} \supset s_n\}$ then
$S_n = S_n \cup \{s_n\}$
end if
end for
$n = n + 1$
end while

Fig. Algorithm for identifying high priority common token sequences.

VI. SCALABLE ANALYTICS: AMAZON REVIEW SUMMARY CHATBOT

A. Proposed System

We propose the following system to tackle the problem statement:

Our main system consists of a web application accessible by the bot developer (here, the administrator) which provides basic metrics as well as user engagement patterns, in specific, the temporal patterns and preferences. Two main tasks are performed by this platform. One is to summarize the reviews and nature of a product or service from multiple sources on the internet. Other caters to capture and analyse user engagement characteristics to provide better recommendations accordingly.
The initial setup is to build an analytics platform with different functionalities, which are open for the public to use via an API interface. We also build a chatbot to give back a summarized version of the Amazon reviews for a particular product queried by the user.

B. Web Crawler

A web crawler scrapes a product page on Amazon.com based on the user query and feeds the unstructured data to a dynamic database. We use a streaming data service like Apache Flume or Spark to transfer this dynamic data to our static data storage.

C. Spark Analytics Algorithm

The resilient distributed dataset represents a read only collection of objects that can be reproduced over multi device platform in a partitioned manner, if one of the partition is lost. Spark architecture involves usage of two types of libraries spark.mllib and spark.ml. Both libraries provide functions for fast and convenient training of machine learning devices. Besides resilient distributed dataset, the spark model involves shared variables over multiple platforms and parallel processing of analytics operations.

NLP Training Module - api.ai

For training our Amazon Reviews Chatbot, we utilize the external NLP engine api.ai built by Speaktoit for its Assistant application. Api.ai provides a functionality of uploading a training corpus and an API to parse an input text and label it in the form of Entities (Objects of Interest, a strong indicator of the context of a conversation), Activities (actions) and Intents (mappings of user input to actions). Api.ai provides a quick way of bootstrapping a working chatbot using a contextual NLP model.

VII. CORPUS OF CONVERSATIONS

The datasets we require to build and train various parts of the application include:

A. Cornell Movie Dialogue Corpus[16]

This corpus contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts: - 220,579 conversational exchanges between 10,292 pairs of movie characters - involves 9,035 characters from 617 movies - in total 304,713 utterances. The movie metadata included:

- genres
- release year
- IMDB rating
- number of IMDB votes
- IMDB rating

Character metadata included:

- gender (for 3,774 characters)
- position on movie credits (3,321 characters)

B. Microsoft Research Social Media Conversation Corpus[15]

The dataset released by Microsoft Research consists of a collection of 12,696 Tweet Ids, each representing a three-step conversation extracted from Twitter. Each row in the dataset represents a single context-message-response triple that has been evaluated by crowd-sourced annotators as scoring an average of 4 or higher on a 5-point Likert scale[17] measuring quality of the response in the context. In order to access the underlying tweets and related metadata, we will need to call the Twitter API[18].

C. Amazon review Dataset

Product reviews from Amazon.com covering various product types (such as books, dvds, musical instruments). The data has been split into positive and negative reviews. There are more than 100,000 reviews in this dataset.
reviews come with corresponding rating stars. Itemized reviews in the dataset have been filtered and cleaned as positive, negative and unlabeled from the raw dataset.

ACKNOWLEDGMENT

We are thankful to our college Vivekanand Education Society’s Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project. It gives us immense pleasure to express our deep and sincere gratitude to Associate Professor Mrs. Mrs. Sujata Khedkar (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions. We are deeply indebted to Head of the Computer Department Dr. (Mrs.) Nupur Giri and our Principal Dr. (Mrs.) J.M. Nair , for giving us this valuable opportunity to do this project. We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully. We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering. We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement at several times.

REFERENCES


[18] Twitter Gnip tool for Academia : https://gnip.com/academic/