Chatbot Development
3rd Year Group Project
Prototype Documentation

Development Supervisor:
Dr. Matt Smith

Student Developers:
Daire Homan
B00029598

Brian Kelly
B00082716

Aaron Ward
B00079288

School of Informatics
Department of Informatics and Engineering
Institute of Technology,
Blanchardstown Dublin 15

Abstract
Chatbots are computer programs that mimic conversations with people using artificial intelligence. They can transform the way you interact with the internet from a series of self-initiated tasks to a quasi-conversation. Chatbots provide a conversational experience with the user whereby the user can request information or carry out tasks whilst texting a piece of software that feels like they are texting a human. Chatbots are being developed today in high volume to plugin to exisiting platforms such as Facebook Messenger, WhatsApp, Skype and Twitter. Facebook have released an API for developers to build their chatbots and deploy them to their Messenger platform. There are several frameworks and API's available to developers centered around building chatbots such as Facebook Messenger, Wit.ai, PandoraBots and several more. This documentation, along with the working prototype, aims to provide an overview of the potential features that can be implemented whilst building a chatbot and introduce areas for future development and research within this project.

1. Introduction

Through research and investigation the selection of a chatbot for development was enticing because of several factors. This paper aims to clarify the motivation in behind picking this particular piece of software for development. Additionaly, an overview of the working prototype will demonstrate some common use cases for a chatbot application alongside an explanation of how an app of this nature handles data. Finally, a robust outline as to how the chatbot will progress from a prototype to an application that will demonstrate high quality features that would be found in a real-world chatbot application. Features for implementation will be chosen based upon the necessity to demonstrate both individual capabilities and the teams capabilities to co-operate and develop software using tools to aid collaborative development.

2. Motivation

2.1 Chatbots

Chatbots have become increasingly popular since large companies have opened up their applications for chatbot developers to deploy to their platform. This opportunity embodies an exciting endeavor for developers because they're product can be made visible to the world at a significantly faster rate than if they develop their chatbot without the aid of an enormous platform such as Facebook Messenger. The Facebook Messenger application has approximately one billion active users today (*Messenger platform at F8 | Facebook newsroom*, 2015). Therefore by deploying a chatbot to Facebook Messenger it eliminates the need for users to download, install and sign-up to a separate service provided by another application. Additionally, by using Messenger, this provides users with a friendly and familiar user interface that gives a means of interaction without the need of acquainting themselves with a new application. The primary motivation, however, is that this implementation also nullifies the requirement of multiple developments across platforms as Facebook Messenger is available to the two most popular operating systems, Android and iOS (Zhang, 2015) and other operating systems such as Blackberry OS and the Windows Phone 8 (Fingas, 2014). Instead of learning the native code that is required to develop on each of these platforms, a chatbot can be developed in any programming language and deployed across all platforms non-discriminately.

2.1 Version Control

Collaborative development is a necessity for any project that is of large scale and a high level of complexity. Though this projects scale will not compare to that of a team with many developers, it will embody all the appropriate mechanisms for successful team development. Versioning source code for any computing project is an extremely resourceful skill to learn from both an academic and a professional standpoint. Post-prototype development of the chatbot application will be versioned entirely. This is to provide developers with a rich learning experience of the "do's and don'ts" when it comes to developing an application in a team that is being version controlled.

2.3 Testing and Deployment

Testing the modularized components of the application will also be a core motivation behind this project. In a professional production environment, code is rigorously tested both by testing units of the source code and doing extensive build tests. After all tests have been passed the application is deployed to the host server where users can access it and benefit from its features. There is an

array of effective tools to use when trying to carry out such tests and these tools can be configured to run automated tests once code is committed to the applications repository. Also, once code has successfully met all requirements to pass the tests it can then be auto-deployed to the server environment. Once again, exposure to using such a tool will be incredibly beneficial to the team members maturing towards being developers that can be relied upon in a professional production environment that requires critical testing of software.

2.4 RESTful API's

Implementation of features utilising REST API's will also be a primary focus. The language of the web today is becoming increasingly standardised and applications need to be able to talk to each other with a unified mechanism that allows the retrieval, updating and removal of data from applications across different domains. REST API's are currently the most efficient way of providing an always-on service that can be accessed by whatever application that requires its data. For the purpose of code modularity, experience and the demonstration of individual team member capabilities each feature that the chatbot will possess will be implemented as its own web-service. This strategy will lend itself to real-world applications and will mimic the operations of a service or feature created in a production environment. Once again, this exercise will be a proficiently positive way for team members to gain experience with real-world web service development methodologies.

3. Prototype

The prototype demonstrates some basic features that a chatbot application would generally encompass. Whilst the features are not currently implemented using a web service architecture or natural language processing, they exist purely to demonstrate the feasibility of the proposed features for future development.

3.1 Facebook Messenger

The prototype is deployed as a Facebook Messenger application and can be accessed by any authorized developers. In the event that development of the app reaches a point whereby it can be deemed releasable, the app can go live on the Facebook Messenger platform for users to interact with. For the purposes of clarity and direction for the project however, the app will only be developed with the initial goal of not going beyond the test/sandbox environment. In whole, this demonstrates that deployment to the Facebook Messenger platform is feasible. However, in the unlikely, yet possible, event that Facebook revoke developers access to their platform, a very basic UI will be designed as a fail-safe to pre-empt this type of situation. If this situation should arise and a custom UI has to be created for the chatbot, the team will still be able to achieve the core deliverables of the project. See the below screenshot for the chatbot welcome screen.

//******insert diagram

3.2 Conversational Interface

The prototype achieves a conversational feel. The goal of the responses from the chatbot will be to make them as "human-like" as possible so that users can interact with them naturally. There are a number of ways to achieve this with a piece of software and NLP will be used to provide the final applications "human" feel. There are API's that exist that aid development whereby the developer can use it to parse through a users input and extract context from the users message and give a rating of how likely The prototype demonstrates very basic comparisons of string data

types and returns a pre-prepared response if the comparison condition is met, otherwise it will throw a default message notifying the user that it didn't understand the last interaction.

//******insert diagram

3.3 Provides Information to the End-User

A core component of any useful chatbot is the provision of useful information through the conversational UI outlined above. The information chosen to demonstrate with the prototype is a bus timetable provided to the user upon request. This demonstrates thet the chatbot can be configured to provide an array of different information sets upon the users request and also that said information can be retrieved by querying one of the many web services available to developers to give accurate and timely updates i.e Dublin Bus API is a rich source of real-time data that can aid a user catch their next bus on time.

//******insert diagram

3.4 Drive the Information Pathway with UI Features

It is integral to the final build that the software can drive the conversation down the appropriate pathway. This is to rule out the chatbot getting confused with context i.e if the user requests a bus timetable in a way that the chatbot has not experienced before that it doesn't provide data about another context area such as library opening times. This can be achieved with UI features that direct the conversation pathway. For the prototype UI buttons have been created to initiate the user on a particular conversation path. See below for example of prototype UI features.

//*****add diagram

3.5 Launch In-App Browser

Facebook Messenger allows for an in-app browser to be launched within the chatbot application. This can allow the user to launch an external URL to retrieve a certain piece of information and return efficiently to the conversation, without having to launch an external browser or application, once their request has been processed. See below for prototype example.

4. Strategy For Future Work

4.1 Toolset for Development (subject to change slightly during implementation)

-NodeJs
A lightweight, server-side, JavaScript functional programming language that is ideal for web services because of its speed and performance capabilities.

-Git and GitHub
Industry standard version control tools for both versioning local code and pushing to a host repository for team members to collaborate.

-Travis CI
Industry standard tool for unit testing, build testing and automated deployment.

-Heroku
Secure host server that will allow multiple instances of the same application on the same domain, this is ideal for testing before promoting code to the 'master' build.
-Wit.ai

A NLP framework that allows user to build a DB of conversational inputs and categorise the inputs by context and intent.
-Swagger
A framework that makes creating a REST API very efficient and also generates documentation as you develop them.
-MongoDB
A service that allows you to store data in a NoSQL(JSON) format that is lightweight and always-on.

4.2 Timeline of development

//*******insert sprint thing fuck it

Conclusion

To achieve a satisfactory project the chatbot will be implemented in a way that will have a balance between rich features and still adhere to proper software development methodologies. The prototype demonstrates the potential features that this app will have with future work by generalizing some of the features. The toolset mentioned above will be utilized to as much of the 'heavy lifting' as possible and the timeline mentioned will be the developers guideline for execution and completion of the final build.

References

Messenger platform at F8 | Facebook newsroom (2015) Available at: https://newsroom.fb.com/news/2016/04/messenger-platform-at-f8/ (Accessed: 12 October 2016).

Messenger platform - documentation - Facebook for developers (2016) Available at: https://developers.facebook.com/docs/messenger-platform (Accessed: 12 October 2016).

Russell, J. (2016) Chatfuel lets publishers — and anyone — build bots for messaging apps. Available at: https://techcrunch.com/2016/03/18/chatfuel-lets-publishers-and-anyone-build-bots-for-messaging-apps/ (Accessed: 12 October 2016).

Messenger platform at F8 | Facebook newsroom (2015) Available at: https://newsroom.fb.com/news/2016/04/messenger-platform-at-f8/ (Accessed: 12 October 2016).

Zhang, L. (2015) A faster way to message on mobile | Facebook newsroom. Available at: https://newsroom.fb.com/news/2011/08/a-faster-way-to-message-on-mobile/ (Accessed: 12 October 2016).

Fingas, J. (2014) Facebook messenger arrives for windows phone sans voice features. Available at: https://www.engadget.com/2014/03/04/facebook-messenger-arrives-for-windows-phone/ (Accessed: 12 October 2016).

Wong, J.C. (2016) *What is a chat bot, and should I be using one?* Available at: https://www.theguardian.com/technology/2016/apr/06/what-is-chat-bot-kik-bot-shop-messaging-platform (Accessed: 12 October 2016).