Designing a scalable Conversational Interfaces

Cristian Gutierrez[1§], **Darshi Kasondra**[1§], **Archana Ravi**[1§], **Mounica Dingari**[1§]

[1]California State University of Northridge

Additional work related to this paper can be fournd at https://github.com/thehamop1/AdvSoftwareEngineeringProject;

This version dated May 9, 2021

Abstract The way we use interact with systems is contantly evolving with new advances in technology. As buisness continue to need flexible user interfaces many of them are implementing conversatinal interfaces in order to fufill customer needs without having to require users to learn how to use an entirely seperate application in order to make transactions. Modern conversatinal interfaces are fairly powerful with recent adavancements in natural language processing, integration with mobile asistants, and flexible frameworks. In this paper we provide background to some of the concepts of conversatinal interfaces, examing the current state of commercially available solutions, and provide a reference architecture for implementing a system centered around a conversatinal interface.

***For correspondence:**
cristian.gutierrez.56@my.csun.edu (CG)
Present address: [§]Department of Engineering and Computer Science, California State University of Northridge, United States of America

1 Introduction

1.1 Previous Work, Methods, Procedures

There is a growing need for approchable user interfaces as more interactions become digital. For example banking, transactions, and flights are all largely growing to be digital. In order to easily accessed and usable for all types of users conversatinal interfaces are the most ideal due to their low learning curve. If a user can use their native language in order to complete actions on any given system the need for 24/7 support or complex user interfaces become obsolete. This allows buisness to keep user satisfaction high while keeping costs low. Additionally it provides several benefits to users as they are able to access data from large databases easily, complete transactions even if their not so tech savy, have multilingual support, and have 24/7 support.

1.2 Previous Work, Methods, Procedures

The development and interest of conversatinal interfaces was has been a subject of interest since the 1970's. Early examples of these early chatbots include ELIZA, ALICE, and PARRY. Many of these early chatbots worked with the use of simple pattern matching. This simple regular expression based matching was combined with a tree design for controlling the flow of conversations. One of the major drawbacks of this naive form of design was the frequent matching of user utterances with conversation points that happened further up the tree. This would often lead to looping conversations. Researchers at the time had to develop markup based langugues such as AIML in order to develop expert systems. These large complex forms of nested databases had to be constantly maintined in order to add new features to these chatbots. With the modern development of machine learning algorithms in order to parse and extract meaning from user utterances. Many commercial solutions for developing conversatinal interfaces are now widely available and come with many integrations for various platforms.

1.3 Background

There are some basic concepts that are universal to most conversatinal interface platforms. The first is the user utterance which can either come in the form of a text entry or speech with the use of a microphone. Additional signal processing is required to transcribe the audio signals into text. This text is usually normalized where all text has its puntuation removed and all letters are moved into the same letter case. Next depending on the platform various machine learning algorithms are used in order to extract certain key tokens from the string. The most intent which is the objective of the user. For example the utterance "What is the weather in Los Angeles" would have the intenet of weather. This is a topic that the application would have to be designed to respond to. Next would be entities which are certain tokens the application will use to complete requests once an intenet has been deduced.

2 Methodology

Here (and in other sections if needed) you would describe how you compared the software in question, with subsections as needed. Perhaps you may also need to comment here on why you did not include certain pieces of software or how these might be included in the future.

2.1 Physical systems and properties

What did you calculate, on what systems? Why did you make that choice?

2.2 Methodology for comparison

How did you compute what you're comparing? What software versions did you use? How would the software be accessed?

2.2.1 Method selection

How did you select which protocols to use? How are you certain these are representative? Would a naive user get similar performance?

2.2.2 Supporting files

Are there run scripts or files others would need to reproduce your work? Are these provided in your GitHub repo, or how would they be accessed? How exactly would someone reproduce/extend this work?

2.2.3 Error analysis

You may wish to devote particular attention to error analysis.

3 Results

You would give results.

4 Discussion and conclusions

You may perhaps wish to revisit the issue of what performance a naive user might expect, here.

5 Author Contributions

(Explain the contributions of the different authors here)

For a more detailed description of author contributions, see the GitHub issue tracking and changelog at https://github.com/thehamop1/AdvSoftwareEngineeringProject.

6 Other Contributions