12-2022

Effective Detection of Local Languages for Tourists Based on Surrounding Features

Tobenna Eze

Effective Detection of Local Languages for Tourists Based on Surrounding Features

by

Tobenna Eze

A Starred Paper

Submitted to the Graduate Faculty of

Saint Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Computer Science

December, 2022

Abstract

The tourism industry is a trillion-dollar industry with many governments investing heavily in making their countries attractive enough to entice potential visitors. People engage in tourism due to different reasons which could range from business, education, leisure, medical or ancestral reasons. Communication between intending visitors and locals is essential, given the non-homogeneity that occurs across cultures and borders. In this paper, we focus on developing a cross-platform mobile application that listens to surrounding conversations, is able to pick certain keywords, automatically switch to the local language of its location and then offer translation capabilities to facilitate conversations. To implement this, we depend on the Google translate API for the translation capabilities of the application, starting with the English language as our base language. To provide the input (speech) for translation, we solely employ speech recognition software using the Speech-to-Text package available on Flutter. The output with the correct pronunciation (and local accent) of the translation is done with the Text-to-Speech package. If the application does not recognize any keywords, the local language can be determined using the geographical parameters of the user. Finally, we utilize the cross-platform competence of the Flutter software development kit and the Dart programming language to build the application.

Acknowledgement

I would like to sincerely thank my advisor Dr. Maninder Singh, for his guidance, mentorship, and whose expertise in the subject matter provided clarity for me and helped make this work a reality. I would like to express my immense gratitude to the members of the committee, Dr. Mark Petzold and Dr. Shakour Abuzneid, whose input and positive feedback contributed a great deal to achieving this work.

A special thanks goes to the Dean of the faculty of Computer Science and Information Technology (CSIT), Dr. Ramnath Sarnath, and every faculty member for their contributions to my academic development and competence. I would also like to extend my gratitude to Mr. Clifford Moran, for his continuous and invaluable help throughout the entirety of my program with registration of classes, scheduling meetings etc.

Finally, I also want to thank my parents, Mr. Nnaemeka and Dr. Nkechi Eze for their continuous prayers, support, and words of encouragement towards my success and goals. I also want to thank my friends for their help in completing this work timely.

Table of Contents

List of Figures

Figure Page

List of Tables

Chapter 1: Introduction

1.1 Overview

Natural Language Processing is a neoteric interdisciplinary field that examines the interaction between computers and human languages. It continues to expand and can be used in a multitude of applications such as email filters, predictive text, and language translation. Speech recognition is a subset of this field. Speech recognition, also known as Automatic Speech Recognition (ASR) can be defined as the capacity for a machine or computer program to analyze and process speech or a person's spoken words, understand it, and convert it into readable text. Nowadays, speech recognition finds its use in a variety of forms. One of which is in mobile phones through speech recognition software like the Google assistant in android devices, Siri in iOS devices and it is also used in the Amazon virtual assistant, Alexa.

As mentioned earlier, speech recognition being a subset of Natural Language Processing can be used in language translation. The ability of text to be translated automatically by a computer is known as Machine Translation. "Machine translation (MT) was first developed in the mid-20th century" (Tongpoon-Patanasorn & Griffith, 2020). The main aim of MT development was to replace human translation due to the hindrances of deliberate translating processes and possibly expensive translation costs (Brown et al., 1990). Statistical MT was first presented as a research project by IBM (Brown et al., 1990).

This could be used in tourism where we have travelers traversing different countries and continents, which provides a need for an effective electronic means of language translation facilitated by speech recognition software.

1.2 Problem Statement

The tourism industry is an evolving industry that continues to enjoy rapid expansion and growth. People embark on tourism for a myriad of reasons, these could be for business purposes, relaxation, medical, exploratory, and religious purposes. Due to cultural, linguistic, and social differences, visitors to host countries (or various tourist destinations) experience difficulty when communicating with locals. This communication gap greatly impacts the experience for the tourist because it limits the ability to interact with the host culture. Learning a new language is time-consuming and is a difficult task to expect from potential visitors, given the limited amount of time they usually spend in their tourist destination. In this project, we aim to develop a mobile application which utilizes speech recognition technology to translate phrases or sentences made by a tourist, to the local language of their surroundings. This is done with the ability to switch to the local language by listening out for high frequency key words (or salutation words).

1.3 Motivation

The potential of Machine Translation assisting with automatic recognition of a surrounding language has led to a lot of research being performed in this field. This has served as a motivation towards solving this problem in the tourism industry. We intend for this mobile application to be an efficient, reliable resource for tourists, and for it to also be easily accessible for those of them that are not very tech-savvy. Cohen & Cooper (1986) posit that, “Language barriers are, as everyone knows, an important obstacle to transcultural communication” (p. 534). Due to this barrier, Cohen & Cooper (1986) also assert that communication plays a considerable role in the choice of destination for tourists (p. 534). Studies show that tourists tend to pick destinations that are culturally similar to theirs when making decisions in relation to travel. Mobile translation applications that offer a smooth conversational experience for the tourist could bridge this gap. This makes the choice of where to vacation less daunting for the tourists due to issues arising from language or cultural differences.

1.4 Proposed Solution

Recent developments in Machine Translation have provided the possibility of more accurate translations for languages. We leverage on that technology, in particular, Neural Machine Translation systems which are more accurate than its predecessor, Statistical Machine Translation systems. We implement a cross-platform mobile application that listens to surrounding conversations, is able to pick certain keywords, automatically switch to the local language of its location and then offer translation capabilities to facilitate conversations.

Chapter 2: Background Research and Proposed Approach

2.1 Overview

This chapter provides a background on the approaches taken by extant translation applications. We also look at extant research done on tourism words as a subset of English for Special Purposes (ESP).

Machine Translation (MT) is the ability of text to be translated automatically by a computer.

> The main aim of MT is to create and enhance automatic translation from one language to another. The main approach of MT employs a corpus-based method in which words or text of the input language are translated by comparing them with samples of languages collected in the database, or parallel corpus. The translations are selected based on a statistical method in order to reduce variables in the translation process and to improve the accuracy of the translation. This approach is very effective in translating words with multiple meanings. MT adopts various models such as the reordering model, word translation model, and phrase translation model (Tongpoon-Patanasorn & Griffith, 2020, p. 135).

2.2 Google Translate

Google Translate (GT) is arguably the most popular machine translation software. As of 2022, it supports 133 languages with plans to expand to more languages. “Google Translate was launched in 2006 as a statistical machine translation service, and it uses United Nations and

European Parliament documents and transcripts to gather linguistic data” (Google Translate, 2022).

Google translate is the most popular machine translation program because it relies on a huge database, resulting in a higher rate of translation accuracy compared to other machine translation applications (Tongpoon-Patanasorn & Griffith, 2020; Anazawa et al., 2012; Groves & Mundt, 2015; Puangthong, 2015). Previously, GT used to operate as a phrase-based MT software which meant that it translated input text to the output language on a phrase-by-phrase basis. Currently, GT employs the use of a Neural Machine Translation engine called – Google Neural Machine Translation (GNMT) – which translates “whole sentences at a time, rather than just piece by piece. It uses this broader context to help it figure out the most relevant translation, which it then rearranges and adjusts to be more like a human speaking with proper grammar”(Turovsky, 2016).

2.2.1 Google Neural Machine Translation (GNMT)

The Google Neural Machine Translation is built off of the NMT framework. Neural Machine Translation (NMT) is a system for mapping input text to associated output text in an end-to-end fashion (Wu et al., 2016, p. 1). It translates words on a sentential level which is in contrast to the phrase-based translation model. Some drawbacks of NMT are that it has a slow training and inference speed, it is ineffective at dealing with rare words and occasionally fails to translate all the words in the source sentence. This often results in a lower accuracy value than phrase-based translation systems. The architecture of NMT consists of two recurrent neural networks (RNNs), one to consume the input text sequence and one to generate the output text

(Wu et al., 2016, p. 1). GNMT consists of 8 layers of Long Short-Term Memory (LSTM) RNNs which helps it address the shortcomings of the NMT. Wu et al. (2016) found that, to improve inference time, GNMT utilizes low-precision arithmetic, advanced with the Tensor Processing Unit (TPU) (p. 2). Secondly, it efficiently deals with rare words by implementing sub-word units which are also referred to as wordpieces. “Using wordpieces gives a good balance between the flexibility of single characters and the efficiency of full words for decoding, and also sidesteps the need for special treatment of unknown words” (Wu et al., 2016, p. 2). Finally, GNMT addresses the occasional failure of NMT to translate whole sentences by using a beam search technique that includes a normalization procedure for comparing hypotheses of different lengths and a coverage penalty to encourage coverage of the entire input (Wu et al., 2016, p. 2).

2.3 Microsoft Translator

Microsoft Translator was developed by the Microsoft Research team in the early 2000’s (Microsoft Translator, 2022). As of 2022, it supports 110 languages (Microsoft Translator, 2022). Microsoft Translator was developed based on the idea of semantic predicate-argument structures known as logical forms (LF) which was spun from Microsoft Word grammar correction feature (Translator Text API, 2019). Microsoft Translator implementation of Statistical Machine Translation (SMT) was built on more than a decade of natural-language research at Microsoft. “Rather than writing hand-crafted rules to translate between languages, modern translation systems approach translation as a problem of learning the transformation of text between languages from existing human translations and leveraging recent advances in applied statistics and machine learning” (Machine Translation, 2019).

Like the Google translate, Microsoft translator now makes use of the Neural Machine Translation. “Leveraging the scale and power of Microsoft’s AI supercomputer, specifically the Microsoft Cognitive Toolkit, Microsoft Translator now offers neural network (LSTM) based translation that enables a new decade of translation quality improvement” (Machine Translation, 2019).

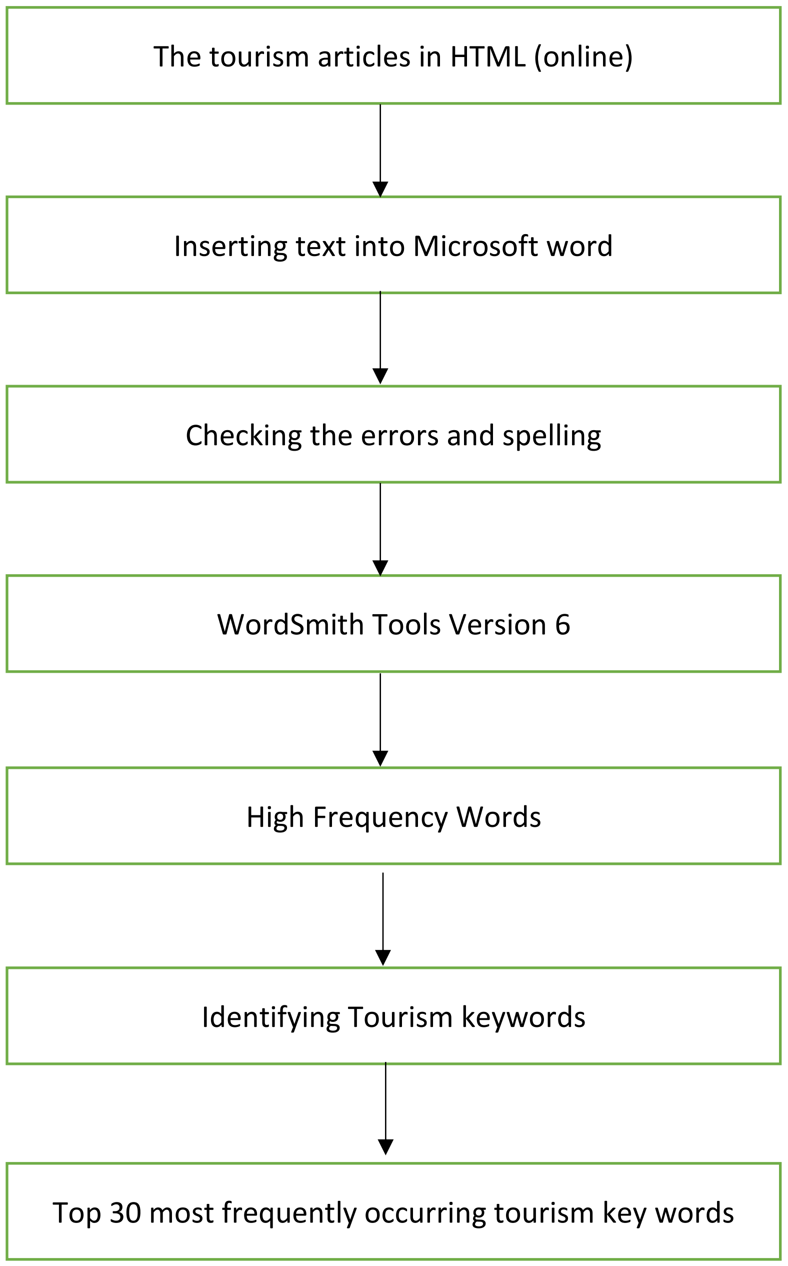
2.4 High Frequency Key Words in Tourism English in Newspapers: A Corpus-Based Approach

Tourism is a rapidly growing industry, and this has been the case since the 20th century. In Thailand, the tourism industry is one fundamental part of income for the country as it involved Thailand’s economy by promoting foreign exchange earnings, creating new jobs, broadening the distributions of income, increasing rural development. “In order to communicate with tourists around the world, language is used to represent people or things and to aid the creation of thoughts” (Holmes, 1998). Quirk et al. (1985) assert that English is the world’s most greatly used language.

Essentially, in this study, Junnak & Juajamsai (2017) analyze the most frequent and important vocabulary in the tourism field. They extracted data from a corpus of 246,601 words which were compiled from a program “Wordsmith Tool” and the tourism section of online Newspapers in Thailand.

Figure 1

Research Procedures Taken


The tourism articles in HTML (online)
Inserting text into Microsoft word
Checking the errors and spelling
WordSmith Tools Version 6
High Frequency Words
Identifying Tourism keywords
Top 30 most frequently occurring tourism key words

From figure 1, the first step in this research was extracting words from the tourism section of online articles, which were collected over a one-month period. These words were then inserted into Microsoft Word, and they surmounted to a total of 90 files. The next step was to check for spelling errors and mistakes, after which the processed news articles were inserted into the "WordSmith Tool Version 6". This tool was used in analyzing the most frequent words in the Tourism English Corpus (TEC). After receiving the most frequently occurring words of the Tourism Corpus, the frequency word list was checked by using the Longman Business English Dictionary manually to identify the technical vocabulary. Then, the top 30 most frequently occurring tourism key words were obtained (Junnak & Juajamsai, 2017).

Table 1

The Top 30 Most Frequently Occurring Tourism Key Words of the Tourism English Corpus (TEC)

Rank	Word	F	Pos	Rank	Word	F	Pos
1	tourism	189	n.	16	domestic	46	adj.
2	million	89	n.	17	industry	46	n.
3	travel	76	n./v	18	local	45	adj.
4	government	75	n.	19	place	43	n./v
5	hotel	75	n.	20	area	42	n.
6	business	71	n.	21	private	42	adj.
7	ASEAN	66	n.	22	sector	41	n.
8	economic	61	adj.	23	development	40	n.
9	market	57	n.	24	visa	39	n.
10	price	55	n.	25	high	38	n./v
11	growth	53	n.	26	president	38	n.
12	City	51	n.	27	money	37	n.
13	center	49	n.	28	resort	36	n.

14	time	47	n.	29	plan	35	n./v
15	billion	46	n.	30	area	34	n.

From Table 1 we see a list of the top 30 most frequently occurring words of the TEC. The total of the frequency in this table were calculated from the 1st rank up to 150th rank. “There were 1,659 occurrences of the whole corpus” (Junnak & Juajamsai, 2017, p.4). It is to be expected that high frequency words in the corpus were nouns and adjectives. From this, Junnak & Juajamsai (2017) are able to show that, the top five high frequency words were “tourism” which showed up 186 times, “million” 89 times, “travel” 76 times, “government” 75 times, and “hotel” 75 times. “Moreover, the above data indicated that the parts of speech of most key words in tourism were noun and the rest are adjectives. Yet, some words can be used both as noun and as verb (3rd rank), and as noun, adjective and as adverb (25th rank)” (Junnak & Juajamsai, 2017, p. 4).

This study had a few limitations which were noted by the researchers. One of which was a relatively low sample size consisting of only 90 online articles which meant that the findings were not applicable to hotel and rental services. Another limitation was that the study only focused on a limited number of linguistic features that are only content words. This meant that other linguistic features like tense, voice and clause modal verbs, imperatives, personal pronouns, and adjectival premodifiers were overlooked.

2.5 Corpus-based Creation of Tourism, Hotel, and Airline Business Word Lists

In creating a word list based on tourism, hotel and airline business words, the researchers first had to develop a corpus. Prior to creating this corpus, they surveyed a group of graduate students using a Google form. The form was shared to a private Facebook group comprising of more than 4500 members consisting of lecturers, students, and graduates of one of the hospitality programs in Thailand. The requirements to participate were that the respondents must have worked in tourism, hotel, or airline business for over a year. After this, a majority of responses from this survey showed that the respondents adjudged that the ability to understand English in websites related to tourism, hotel, or airline business was crucial since tourists always asked questions about the information in websites. Some other recommended sources for data collation included magazines, news, and work operation manuals.

Table 2

The Top 30 Most Frequently Occurring Tourism Key Words of the Tourism English Corpus (TEC)

HOSPITALITY BUSINESS CORPUS	SOURCES	NUMBER OF SOURCES	TOKENS
TOURISM BUSINESS CORPUS (TBC)	Official tourism websites Tourism magazines	152	31,701,430
HOTEL BUSINESS CORPUS (HBC)	Official hotel websites Hotel business news	124	4,835,926
AIRLINE BUSINESS CORPUS (ABC)	Official airline websites Airline work operation manuals	120	15,542,604

From Table 2, the Tourism Business Corpus comprises 31,701,430 running words from 152 different sources as follows:

1. 100 official tourism websites of the first 100 countries with the largest number of travelers ranked by the United Nations World Tourism Organization (2017) (Laosrirattanachai & Ruangjaroon, 2021).
2. 52 tourism magazines published from 2017 to 2018 (Laosrirattanachai & Ruangjaroon, 2021).

The Hotel Business Corpus was compiled by collecting data from 100 official hotel websites ranked as the best 100 hotels in 2017, and hotel news from 2017 to 2018. "As a result, the Hotel Business Corpus contained 4,835,926 running words" (Laosrirattanachai & Ruangjaroon, 2021, p. 58).

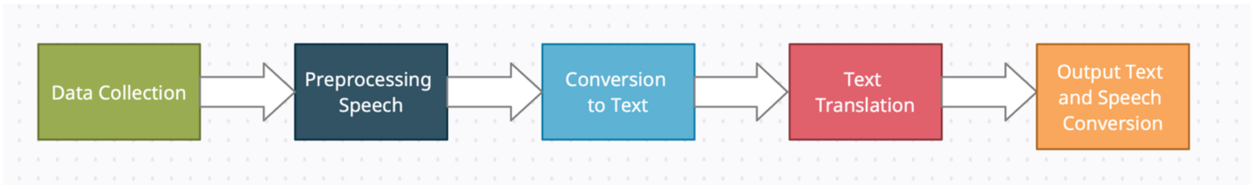
The Airline Business Corpus was compiled by collecting data from 100 official airline business websites which were rated as the best 100 airlines in 2017 and from airline work operation manuals (Laosrirattanachai & Ruangjaroon, 2021). This resulted in the corpus having 15,542,604 running words (Laosrirattanachai & Ruangjaroon, 2021).

2.6 Proposed Approach

Our approach to this project involves: data collection of speech; preprocessing of that speech; conversion of the speech to text with a Speech-to-Text converter; translation of that text to the local language; and finally, conversion of the translated text to speech via a Text-to-Speech converter.

Figure 2

Proposed Steps in Speech Translation



As an example, in collecting data, we take the English statement, "I am glad to be American", spoken into our application which is then translated to Spanish. Below are the next steps taken on the phrase, henceforth referred to as the "speech."

2.7 Speech Preprocessing

After the data (speech) has been collected, the system analyzes this data, and this is done on web servers being used by the system. In this project, we are using a Flutter package called Speech-to-Text (STT). The system breaks the sound (speech) into very little parts referred to as phonemes. In the case of the English language, there are 44 phonemes. The software takes into cognizance, context, usage of each word to derive the best match for the word spoken. The software then matches the word to the set of words in its database (in the STT package, a method called "recognizedwords" is used) and a confidence score is attached to each word. This is then returned to the system as the probable spoken word.

2.8 Conversion to Text

At the conclusion of the speech preprocessing phase, the analyzed words which were matched and returned (in this case: "I", "am", "glad", "to", "be", "American") are then converted to text to be translated. A function to check if the converted text contains a keyword is called. If a keyword is detected, a variable that is set as the language code of the keyword is passed into the function responsible for translation and also to the function responsible for text-to-speech output.

2.9 Text Translation

Here, the text translation is performed using the Flutter Google translate package. This package is powered by a Neural Machine Translation engine known as the Google Neural Machine Translation (GNMT). Essentially, the GNMT takes the text (which was converted from speech sounds) and translates it to the detected language.

2.10 Output Text and Speech Conversion to Local Language

In this final step, the translated text is displayed in the local language detected in the first listen function. The output is also passed through a text-to-speech function (also called "speech") which is achieved using the Flutter Text-to-Speech (TTS) package. The goal of the TTS function is to provide better understanding of the translated text and it achieves this by pronouncing the translated text with a local accent. The translated text results in a Spanish sentence that reads, "Me allegro de ser estadounidense".

2.11 WordList Creation

In creating the Tourist, Airline and Hotel business wordlists we study in this project, the researchers proposed a method they called the Six Filters (6F). They include:

1. Filter Lexical Frequency – This was the first filter used to create the word lists. Coxhead's frequency criterion was applied in this study (Coxhead, 2000). "In her study, Coxhead compiled a corpus of 3,500,000 tokens, and any word that

appeared at least 100 times was considered as passing the frequency criterion" (Laosrirattanachai & Ruangjaroon, 2021, p. 58).

2. Filter Lexical Range – This was the second filter, and the range criterion of Coxhead (2000) was applied. In the study, words that appeared in at least 50 per cent of the total sources passed this criterion (Laosrirattanachai & Ruangjaroon, 2021). "Therefore, words that appeared at least 76, 62, and 60 times in the TBC, HBC, and ABC, respectively, passed the criterion and tended to be included in the word lists" (Laosrirattanachai & Ruangjaroon, 2021, p. 59).
3. Filter Lexical Profiling – Laosrirattanachai & Ruangjaroon (2021) propose that the main concept of lexical profiling is that a word should be put in only a single word list . This helped the researchers eliminate irrelevant words from the created word lists.
4. Filter Lexical Keyness – "This was used to consider unusually high-frequency words appearing in the TBC, HBC, and ABC compared to the British National Corpus (BNC) used as the reference corpora in this study based on the log-likelihood applied in the Key-BNC program" (Graham, 2018).
5. Filter Expert Consultation – "The inputs and feedback from the experts and specialists in the field were gathered to ensure that the word lists were well designed and authentically used in the industry because they use ESP both receptively and productively on a daily basis" (Laosrirattanachai & Ruangjaroon, 2021, p. 60). Their inputs helped decide which words were appropriate to be

included in the word lists (Laosrirattanachai & Ruangjaroon, 2021; Chung & Nation, 2004; Martinez et al., 2009).

6. Filter Lexical Difficulty – Given that a long list of words might cause recognition difficulties for users, dividing such a list into shorter sub-word lists is one way to solve this problem (Laosrirattanachai & Ruangjaroon, 2021). In the current study, the researchers used the VocabProfile program (Cobb, 2018) to divide the three main word lists into sub-word lists based on the difficulty of the words (Laosrirattanachai & Ruangjaroon, 2021).

Table 3

Number of Words in the 3 Word Lists Using the 6Fs

	TBWL	**TBWL**	**HBWL**	**HBWL**	**ABWL**	**ABWL**
Research Procedure	Satisfying the filter itself	Satisfying itself and previous filter(s)	Satisfying the filter itself	Satisfying itself and previous filter(s)	Satisfying the filter itself	Satisfying itself and previous filter(s)
****Tokens to Types**	31,701,430	302,128	4,835,926	65,737	15,542,604	134,862
Filter Lexical Frequency	2,465	2,465	3,548	3,548	2,381	2,381
Filter Lexical Range	2,109	1.785	1,243	1,216	2,047	1,714
Filter Lexical Profiling	273	273	178	178	176	176
Filter Lexical Keyness	446	719	346	524	682	858
****Types to Word Families**	719	672	524	403	858	606
Filter Expert Consultation	378	378	274	274	245	245
Filter Lexical Difficulty	378 words separated into 13 sub-word lists	378 words separated into 13 sub-word lists	274 words separated into 9 sub-word lists	274 words separated into 9 sub-word lists	245 words separated into 8 sub-word lists	245 words separated into 8 sub-word lists

From table 3, we see that after using the Six filter (6Fs) procedure implemented by the researchers, for the Tourism Business Word List (TBWL), 378 words separated into 13 sub-word lists were developed. For the Hotel Business Word List (HBWL), we had 274 words separated into 9 sub-word lists. Finally, for the Airline Business Word List (ABWL), 245 words separated into 8 sub-word lists.

Chapter 3: Experiment Design

3.1 Overview

This chapter describes the steps involved in the design of a system for detecting speech and translating that speech into the local or destination language. As shown in Figure 2 from the previous chapter, first, the data (or speech) is collected or read. Second, this data then goes through a preprocessing phase. Third, this preprocessed data is then matched to the recognizable words found in the database with a confidence level attributed to each word. Fourth, these recognized words are then translated word-for-word into the local language. Finally, the translated words are converted from text to speech and output with the accent of the local language.

3.2 Research Questions

Building off of the technologies discussed in the background research chapter, there might be more optimal ways of accurately detecting the surrounding language. The following research questions will be explored along with the execution of this experiment design.

a) **RQ-1:** How effectively can the program detect the surrounding language and translate to that local language?

   Description of RQ-1: This RQ is aimed at evaluating how effective our proposed application is at detecting background language from the speech or conversation between individuals when it is activated. The performance of a speech recognizer is typically evaluated using metrics such as number of word-level insertions, deletions, and mismatches. This basically measures how accurate our system is at capturing what was said on a word level. How many word-level insertions, deletions, and mismatches did out system experience?

b) **RQ-2:** What are the important words used by tourists in emergency situations?

Description of RQ-2: This RQ is aimed at studying the tourism wordlists and identifying which words could be used in emergency situations by tourists. We define an emergency situation as a situation where a tourist could be in danger or need assistance. This help could come in form of medical aid, help with directions, assistance contacting law enforcement agents or help with a fire outbreak. What are the useful words used in such situations?

3.3 Data Collection

There are two functions built to listen to speech in this application. The first listen function (called "initialListen") is built to listen for the keywords and the second function (called "_listen") is built to listen to the conversation the user is trying to translate. When the application is launched, the first listen function is called. This function has a timer set to 40 seconds which means that after launch, the first function starts listening for 40 seconds trying to catch any keywords in the discussion before it stops listening. The data used in this research is from regular speech conversations. For the first listen function, the conversations should contain salutation words (keywords) from the two languages which are the focus of this experiment. The two languages are Spanish and French.

Some of the Spanish keywords are salutation words such as "buenos dias," "buenas tardes," "bienvenidos", "hola", "buenos noches" etc. Alternatively, some of the French keywords are “bonjour”, “bonsoir”, “comment ca va?” etc. The second listen function is called by clicking a button on the application.

In addressing RQ-2, we decided to study the wordlists developed by Laosrirattanachai & Ruangjaroon (2021) in their paper titled, “Corpus-Based Creation of Tourism, Hotel, and Airline Business Word Lists” and the wordlist developed by Junnak & Juajamsai (2017) in their paper titled, “High Frequency Key Words in Tourism English in Newspapers: A Corpus-Based Approach”. Our goal was to study the words in these wordlists that could be used in emergencies. We defined an emergency as a situation where a tourist could be in danger or require assistance.

3.4 Evaluation Metrics

The evaluation metric used for this research is the word error rate (WER). This consists of substitutions (S), insertions (I), deletions (D), and total number of words spoken (N).

$$WER = S+I+D/N \tag{1}$$

With this metric, we are able to measure and determine the accuracy in our translations. We compare the original text to what our system is able to pick up and measure the number of substitutions, insertions and deletions present in the total words spoken.

For example:

Original text : “I have a lively and bubbly personality”

System capture : “I have a lively personality”

In this example, we can see that the system is able to correctly match five (5) out of seven (7) words from the original text. This gives it an accuracy of 0.714 (5/7).

Chapter 4: Experiment Procedure

4.1 Overview

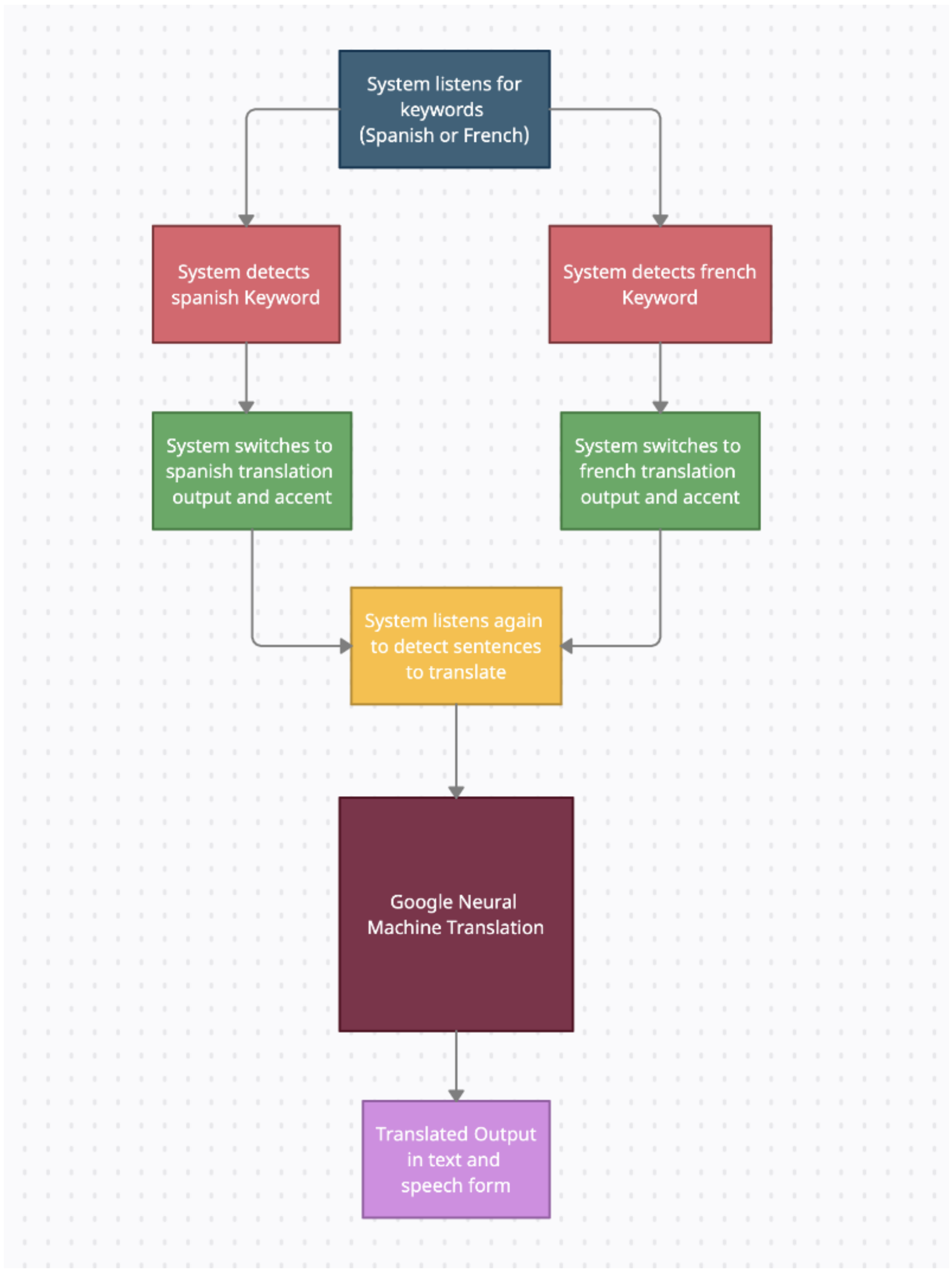
This chapter describes the procedures involved in building a system for detecting speech and translating that speech into the local language**.** As stated earlier in the experiment design chapter, first, the data (or speech) is collected or read. Second, this data then goes through a preprocessing phase. Third, this preprocessed data is then matched to the recognizable words found in the database with a confidence level attributed to each word. Fourth, these recognized words are then translated word-for-word into the local language. Finally, the translated words are converted from text to speech and output with the accent of the local language.

Here, we make use of a few technologies, libraries, and packages. We set up our program using the cross-platform Flutter Software Development Kit (SDK) and the Dart programming language. We make use the Android Studio as our preferred Integrated Development Environment (IDE). Some of the libraries and packages we utilize are the Speech-to-Text package, Google translate package, Geolocator package, Geocoding package, and the Text-to-Speech package.

4.2 Overall System Procedure

Figure 3

Overall System Procedure

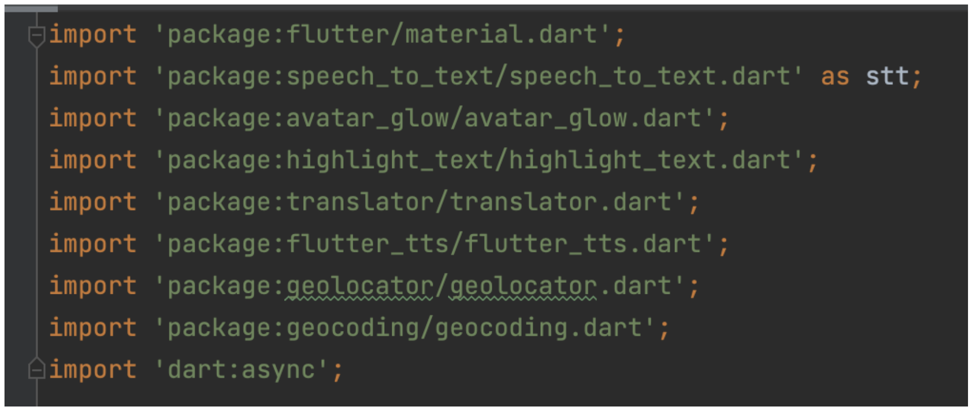

System listens for keywords (Spanish or French)
System detects spanish Keyword
System detects french Keyword
System switches to spanish translation output and accent
System switches to french translation output and accent
System listens again to detect sentences to translate
Google Neural Machine Translation
Translated Output in text and speech form

4.3 Install libraries

In the setup phase of our program, we install a few libraries and packages. We also add the corresponding dependencies of these packages to the pubspec.yaml file. The packages installed were: the avatarglow package which is responsible for the glow on the microphone button to indicate the program is currently listening to speech; the Speech-to-Text package, which is responsible for the conversion of speech to text; the translator package, which focuses on the translation of the text; the Text-to-Speech package, which handles the conversion from txt to speech; the geolocator and geocoding packages, primarily tasked with identifying the location of the user.

Figure 4

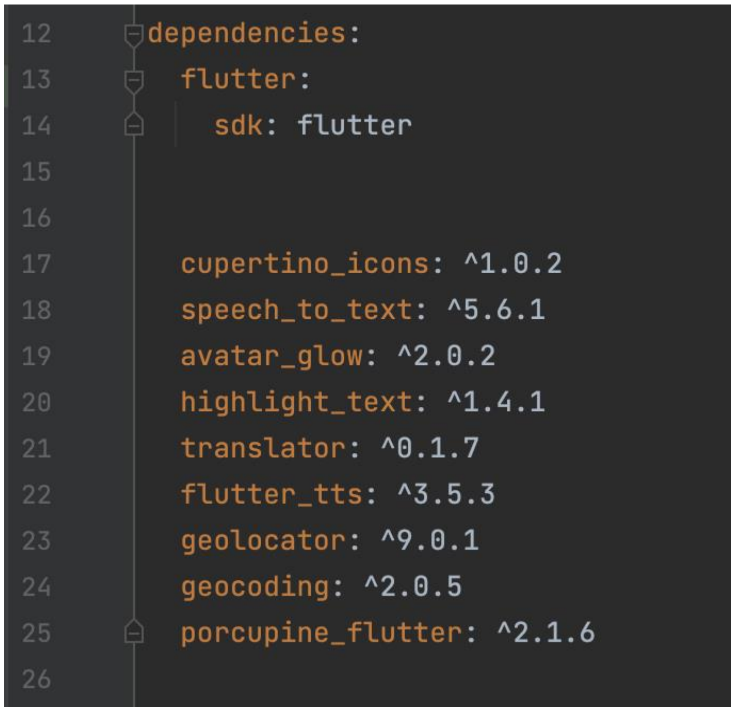
Packages Imported Into Program



```dart
import 'package:flutter/material.dart';
import 'package:speech_to_text/speech_to_text.dart' as stt;
import 'package:avatar_glow/avatar_glow.dart';
import 'package:highlight_text/highlight_text.dart';
import 'package:translator/translator.dart';
import 'package:flutter_tts/flutter_tts.dart';
import 'package:geolocator/geolocator.dart';
import 'package:geocoding/geocoding.dart';
import 'dart:async';
```

Figure 5

Dependencies Added to pubspec.yaml file



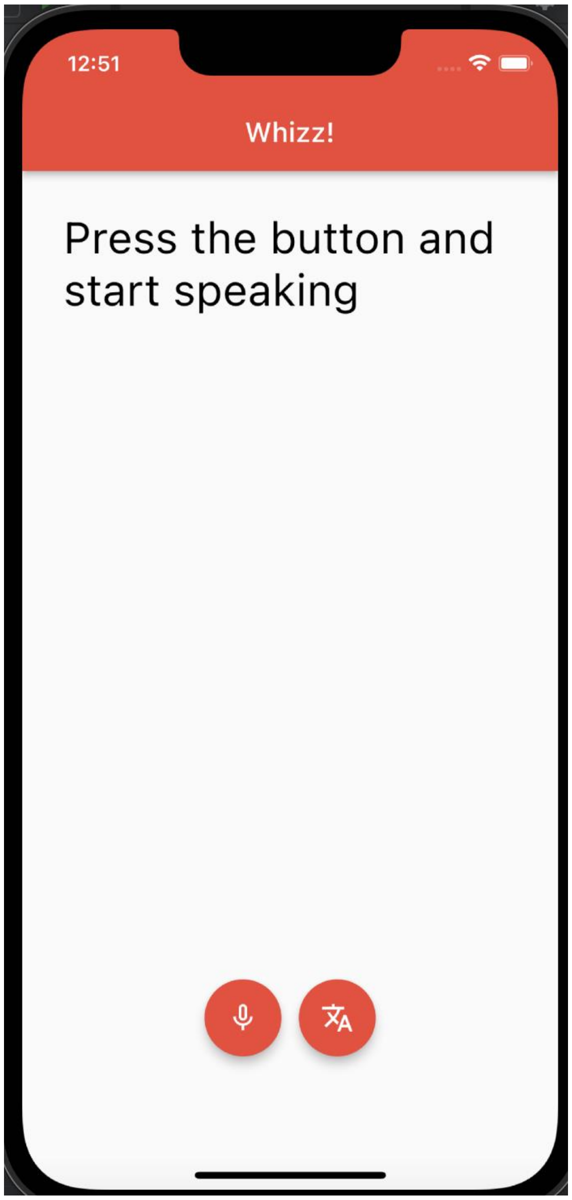
```yaml
dependencies:
  flutter:
    sdk: flutter


  cupertino_icons: ^1.0.2
  speech_to_text: ^5.6.1
  avatar_glow: ^2.0.2
  highlight_text: ^1.4.1
  translator: ^0.1.7
  flutter_tts: ^3.5.3
  geolocator: ^9.0.1
  geocoding: ^2.0.5
  porcupine_flutter: ^2.1.6

```

4.4 Build User Interface (UI) of the application

This step involves building the user interface of the application. In this case, we used a textbox to display the input speech converted into text and also to display the translated text. We also have two floating action buttons, one with a microphone icon for listening to speech and another with a translation icon for performing the translation.

Figure 6

Application User Interface (UI)



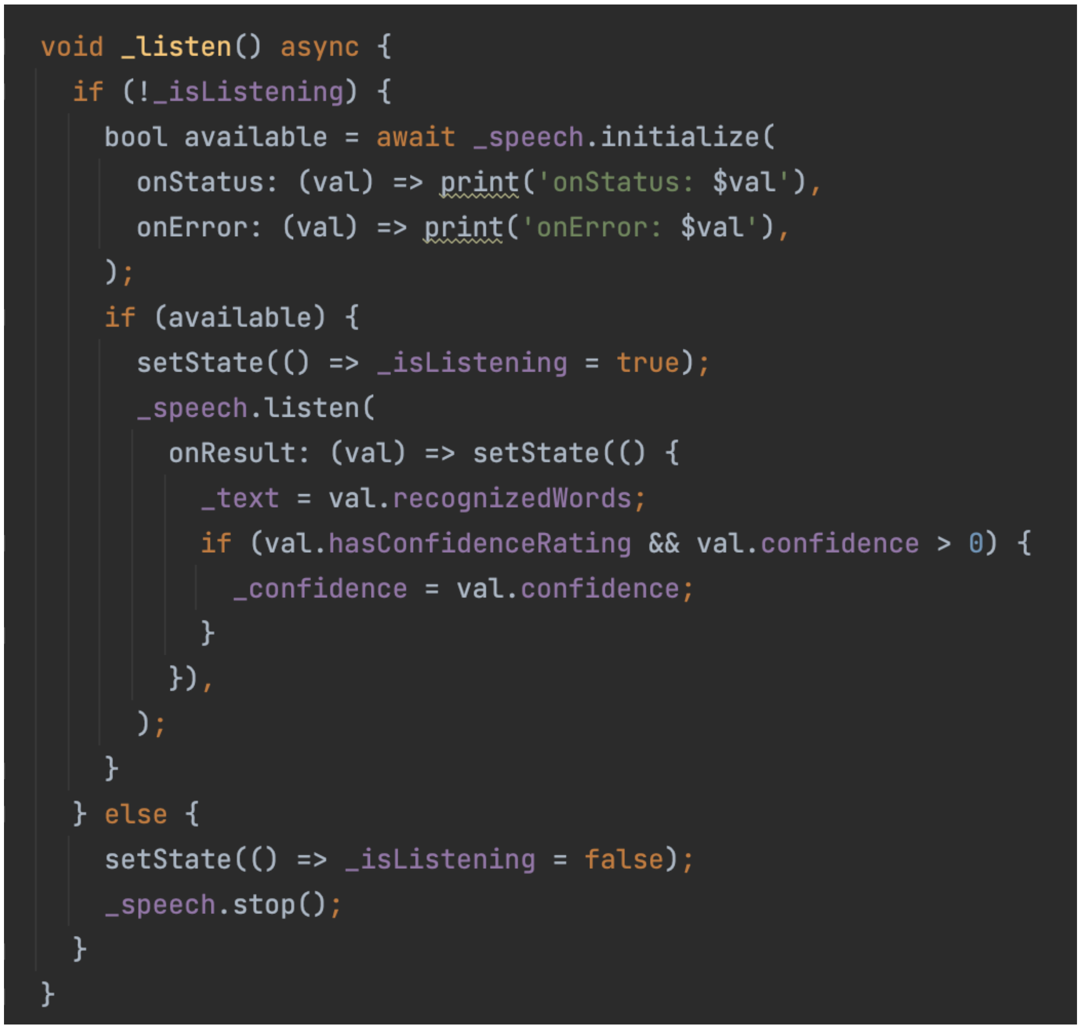
Press the button and start speaking

4.5 Build Speech-to-Text function

We have to build two asynchronous functions for listening to audio input, and they essentially operate the same way. The first function called the "initialListen" function is used to listen out for the keywords in order to facilitate the switching to the language code of the local language. This function is invoked in the “initState” function, and it is initialized for a 40-second period immediately the application is launched. The second listen function called “_listen” is triggered when the microphone button is pressed, and it listens to audio inputs or speech which is then converted to text.

Figure 7

Code Snippet of Asynchronous Function for Listening to Speech



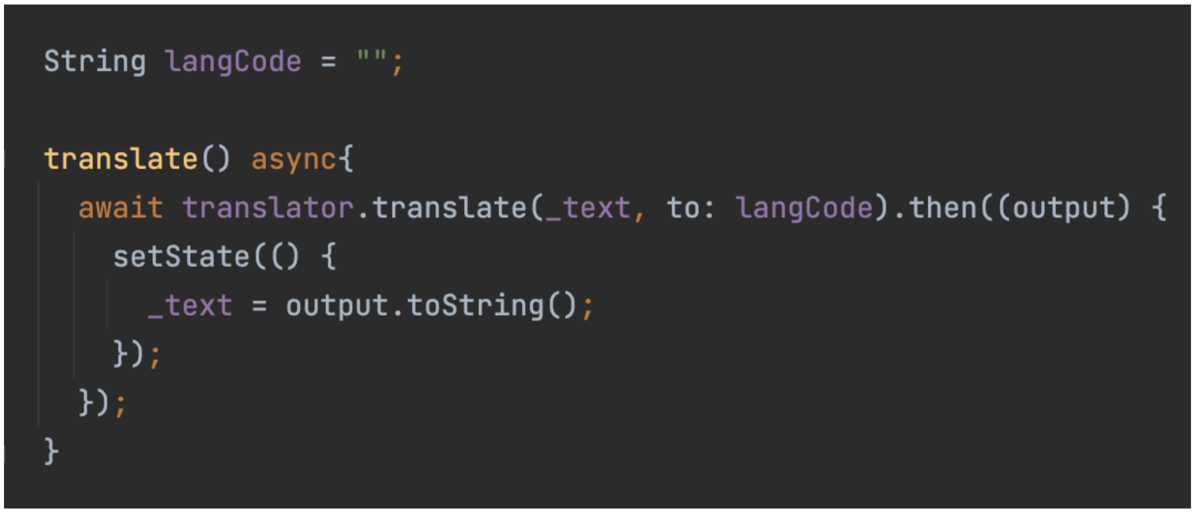
```
void _listen() async {
  if (!_isListening) {
    bool available = await _speech.initialize(
      onStatus: (val) => print('onStatus: $val'),
      onError: (val) => print('onError: $val'),
    );
    if (available) {
      setState(() => _isListening = true);
      _speech.listen(
        onResult: (val) => setState(() {
          _text = val.recognizedWords;
          if (val.hasConfidenceRating && val.confidence > 0) {
            _confidence = val.confidence;
          }
        }),
      );
    }
  } else {
    setState(() => _isListening = false);
    _speech.stop();
  }
}
```

4.6 Build Translate function

In this step we build an asynchronous function for translation which essentially collects the speech converted into text and translates it to the set local language. This is done using the Google Neural Machine Translation engine. It then returns the translated text to the system which is then displayed on the application.

Figure 8

Code Snippet of Asynchronous Function for Translation



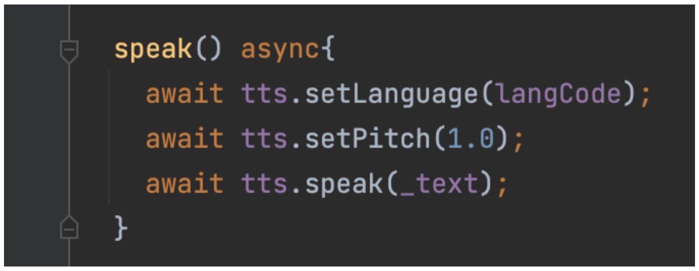
```
String langCode = "";

translate() async{
  await translator.translate(_text, to: langCode).then((output) {
    setState(() {
      _text = output.toString();
    });
  });
}
```

4.7 Build Text-to-Speech function

We build another asynchronous function to handle the Text-to-Speech capabilities of the application. This function takes the translated text and outputs it in the local language with a local accent.

Figure 9

Code Snippet of Asynchronous Function for Speech



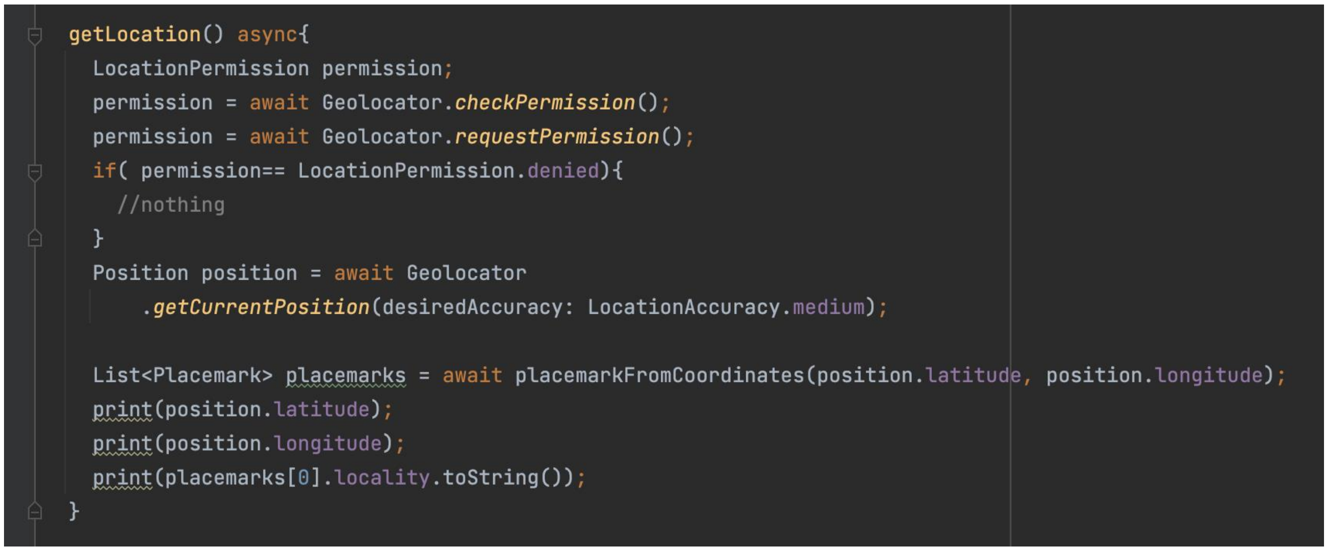
```
speak() async{
  await tts.setLanguage(langCode);
  await tts.setPitch(1.0);
  await tts.speak(_text);
}
```

4.8 Build Geolocator function

In this final step, we build a geolocator function that acts as a backup to our initialListen function. This function is used to determine the location of the user and if the keywords are not detected from the surrounding the language code is set to that generated from the location of the user. This function, like the initialListen function is called in the initState function.

Figure 10

Code Snippet of Asynchronous Function to Determine Location



```
getLocation() async{
  LocationPermission permission;
  permission = await Geolocator.checkPermission();
  permission = await Geolocator.requestPermission();
  if( permission== LocationPermission.denied){
    //nothing
  }
  Position position = await Geolocator
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.medium);

  List<Placemark> placemarks = await placemarkFromCoordinates(position.latitude, position.longitude);
  print(position.latitude);
  print(position.longitude);
  print(placemarks[0].locality.toString());
}
```

4.9 Algorithmic Steps for RQ-1

RQ-1: How effectively can the program detect the surrounding language and translate to that local language?

Description of RQ-1: This RQ is aimed at evaluating how effective our proposed application is at detecting background language from the speech or conversation between individuals when it is activated. The performance of a speech recognizer is typically evaluated using metrics such as number of word-level insertions, deletions, and mismatches. This basically measures how accurate our system is at capturing what was said on a word level. How many word-level insertions, deletions, and mismatches did out system experience?

a.) Build UI and Include a header with application name, text box for display of translated text and floating action buttons to initialize audio recording and translation.

b.) Import Speech-to-Text package and add corresponding dependency to pubspec.yaml file.

c.) Add microphone permissions to both the Info.plist for iOS devices and AndroidManifest.xml for Android devices.

d.) Add geolocator permissions to both the Info.plist for iOS devices and AndroidManifest.xml for Android devices.

e.) Initialize an object instance of the Speech-to-Text package and then call this in the initState() function. This is to initialize the Speech-to-Text functionality immediately the app is opened.

f.) Write two asynchronous functions for listening to audio input. In the first function, we initially check if the device can detect any keywords. In the second function we test if the device was set to listen to audio input (i.e. the microphone button is pressed) and if true, we check if the words are available in the recognized word list. If the device is not set to listen to audio input, we initialize the stop method.

g.) Import the Google Translator package and add corresponding dependency to pubspec.yaml file.

h.) Initialize an object instance of the Google translator and write a translate function that takes the words from the listen function and translates it to the desired local language.

i.) Import the Text-to-Speech package and add the corresponding dependency to the pubspec.yaml file.

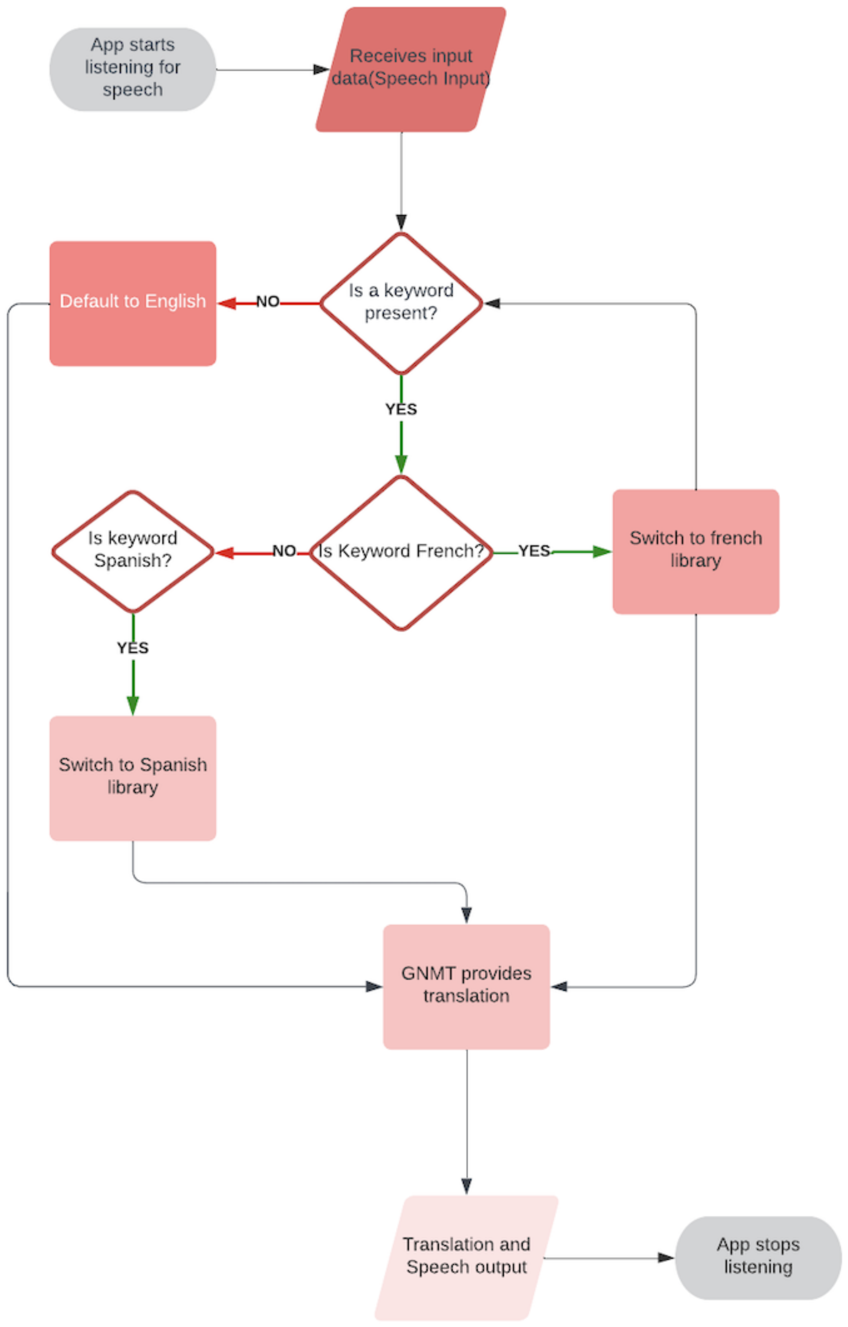
j.) Initialize an object instance of the Text-to-Speech package and write an asynchronous speak function. This function is to produce the translated text in a local accent for easy understanding.

k.) Import the Geolocator package and add the corresponding dependency to the pubspec.yaml file.

l.) Write an asynchronous “getLocation” function to determine the exact location of the user or device.

Figure 11

Flowchart for Algorithm


App starts listening for speech
Receives input data(Speech Input)
Is a keyword present?
NO
Default to English
YES
Is Keyword French?
YES
Switch to french library
NO
Is keyword Spanish?
YES
Switch to Spanish library
GNMT provides translation
Translation and Speech output
App stops listening

Chapter 5: Results and Discussions

5.1 Overview

In this chapter, we list results of the experiment procedure and then we further analyze these results and provide some relevant discussions.

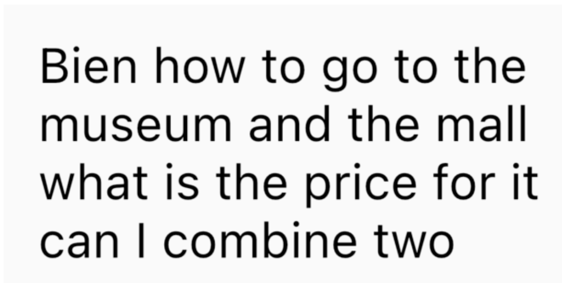
5.2 Results

In testing the efficiency of our application, we performed a usability test with 10 participants to capture how effective the application was at detecting the words spoken to it by the participants. These participants were of different nationalities (China, Philippines, USA, Nigeria, Nepal, India, Mexico) to cater for different accents and word pronunciations which in turn, tests the robustness of the application. We got each participant to come up with a sentence they would like to translate which was noted down. Then we had the participants speak into the application and these were the results:

Participant 1: "How to go to the museum and the mall. What is the price for it? Can I combine two?"

Figure 12

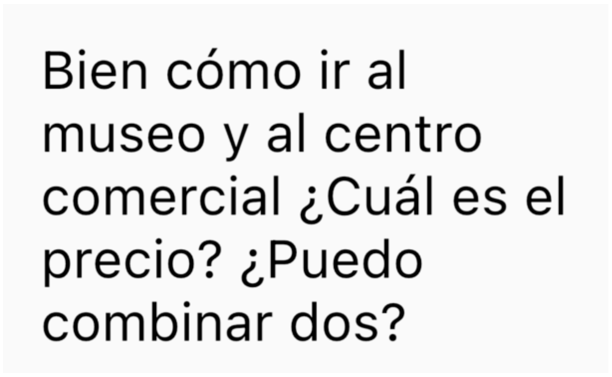
Participant 1's sentence captured by the system



Bien how to go to the museum and the mall what is the price for it can I combine two

Figure 13

Translation of Participant 1's Sentence


Bien cómo ir al museo y al centro comercial ¿Cuál es el precio? ¿Puedo combinar dos?

To calculate our Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 19)

Comparing our input with the result in figure 12, we can say that the WER for this speech is 0.

Participant 2: "It's very cold in Minnesota and I wish I knew this before, I would've gotten a much warmer jacket."

Figure 14

Participant 2's Sentence Captured by the System

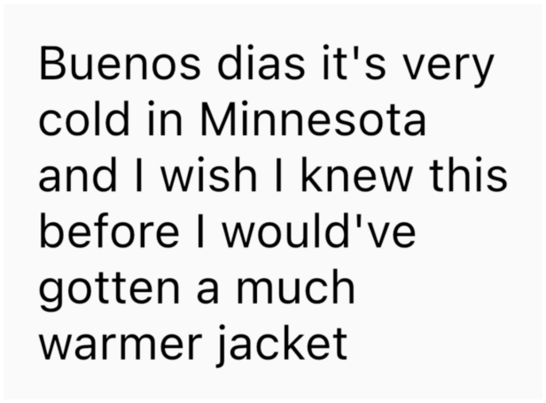
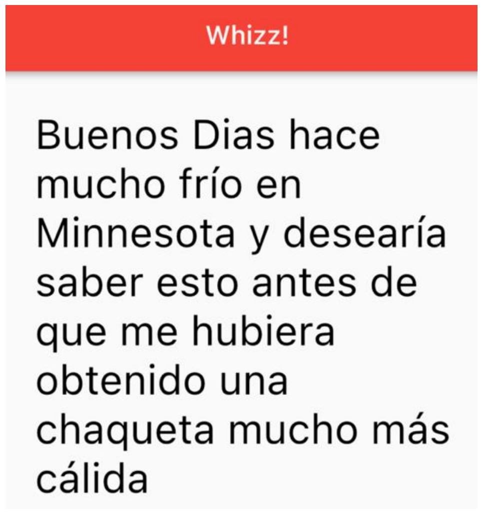

Buenos dias it's very cold in Minnesota and I wish I knew this before I would've gotten a much warmer jacket

Figure 15

Translation of Participant 2's Sentence



Buenos Dias hace mucho frío en Minnesota y desearía saber esto antes de que me hubiera obtenido una chaqueta mucho más cálida

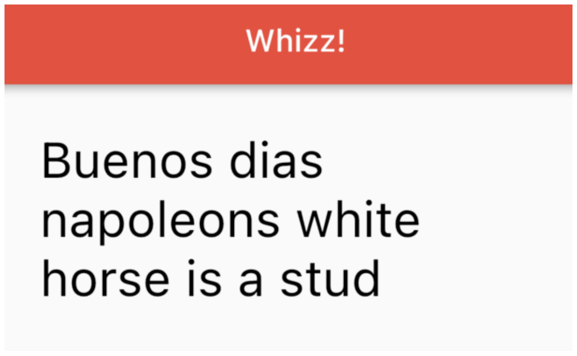
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 19)

Comparing our input with the result in figure 14, we can say that the WER for this speech is 0.

Participant 3: "Napoleon's white horse is a stud"

Figure 16

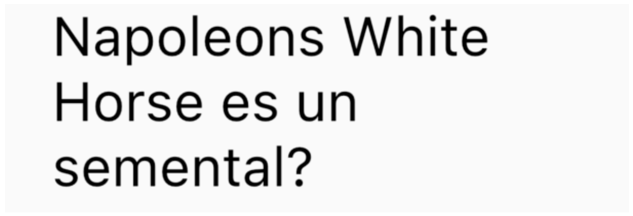
Participant 3's Sentence Captured by the System



Buenos dias napoleons white horse is a stud

Figure 17

Translation of Participant 3's sentence


Napoleons White Horse es un semental?

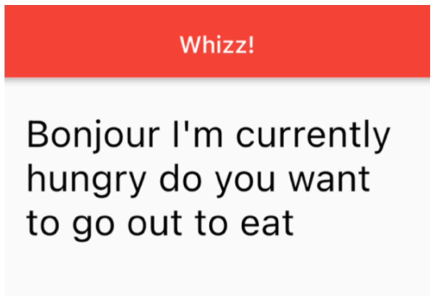
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 6)

Comparing our input with the result in figure 16, we can say that the WER for this speech is 0.

Participant 4: "I'm currently hungry, do you want to go out to eat?"

Figure 18

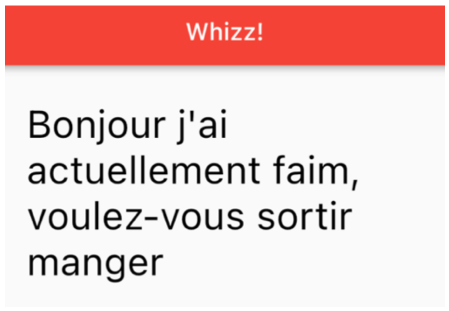
Participant 4's Sentence Captured by the System



Bonjour I'm currently hungry do you want to go out to eat

Figure 19

Translation of Participant 4's Sentence



Bonjour j'ai actuellement faim, voulez-vous sortir manger

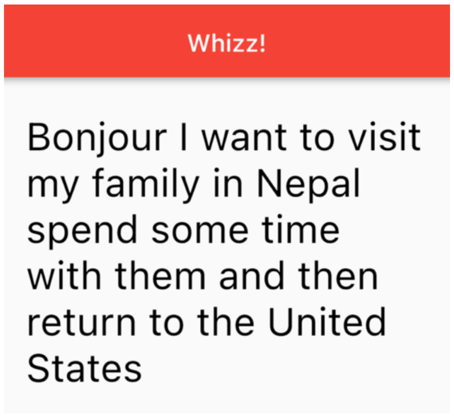
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 11)

Comparing our input with the result in figure 18, we can say that the WER for this speech is 0.

Participant 5: “I want to visit my family in Nepal, spend some time with them and then return to the United States”.

Figure 20

Participant 5's Sentence Captured by the System

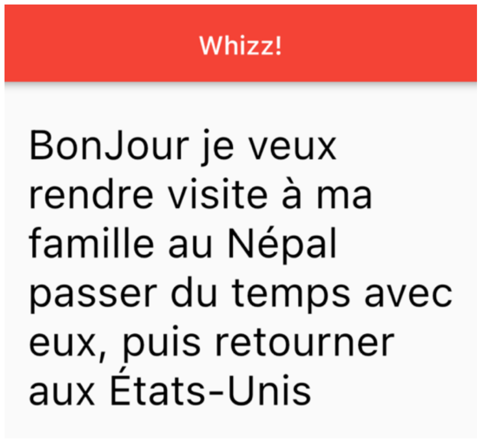


Whizz!

Bonjour I want to visit my family in Nepal spend some time with them and then return to the United States

Figure 21

Translation of Participant 5's Sentence



Whizz!

BonJour je veux rendre visite à ma famille au Népal passer du temps avec eux, puis retourner aux États-Unis

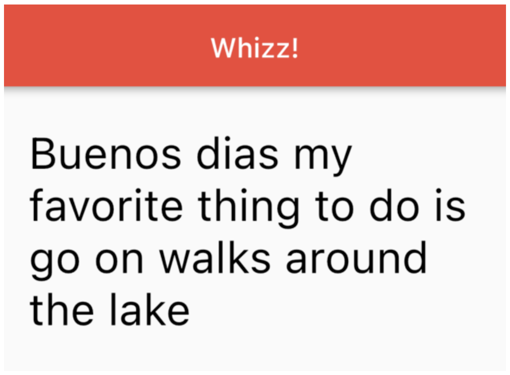
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 20)

Comparing our input with the result in figure 20, we can say that the WER for this speech is 0.

Participant 6: "My favorite thing to do is go on walks around the lake".

Figure 22

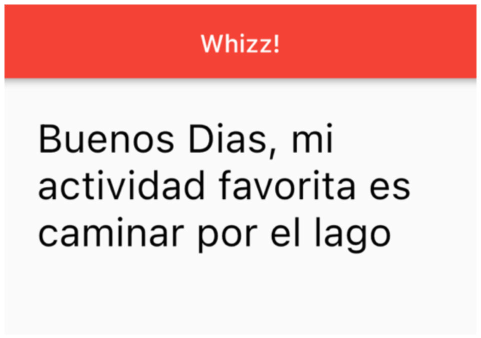
Participant 6's Sentence Captured by the System



Buenos dias my favorite thing to do is go on walks around the lake

Figure 23

Translation of Participant 6's Sentence



Buenos Dias, mi actividad favorita es caminar por el lago

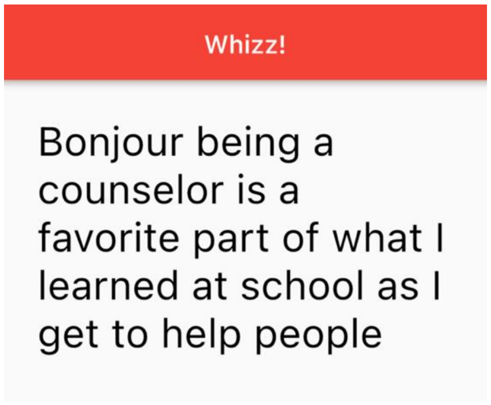
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 12)

Comparing our input with the result in figure 22, we can say that the WER for this speech is 0.

Participant 7: "Being a counselor is a favorite part of what I learned at school, as I get to help people."

Figure 24

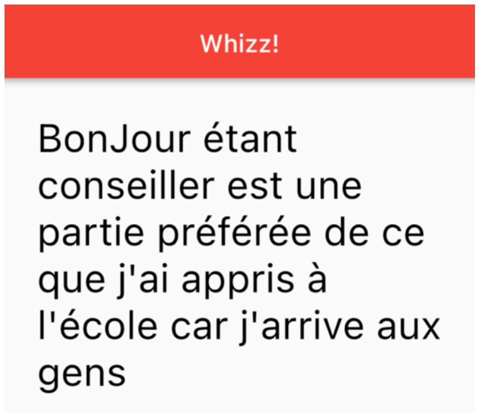
Participant 7's Sentence Captured by the System



Bonjour being a counselor is a favorite part of what I learned at school as I get to help people

Figure 25

Translation of Participant 7's Sentence



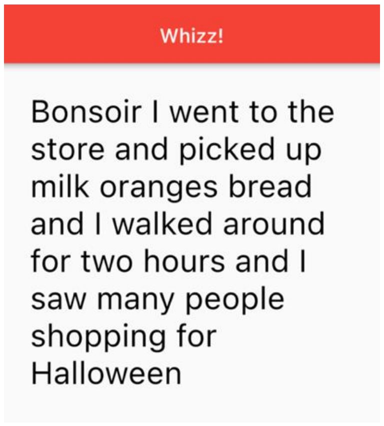
BonJour étant conseiller est une partie préférée de ce que j'ai appris à l'école car j'arrive aux gens

Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 19). Comparing our input from the participant with the result in figure 24, we can say that the WER for this speech is 0.

Participant 8: "I went to the store and picked up milk, oranges, bread and I walked around for two hours, and I saw many people shopping for Halloween".

Figure 26

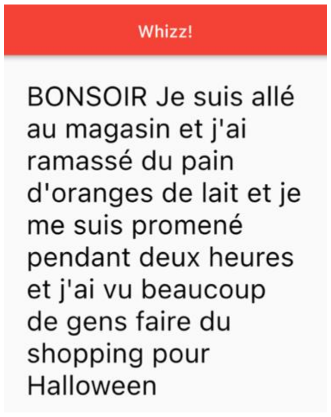
Participant 8's Sentence Captured by the System



Bonsoir I went to the store and picked up milk oranges bread and I walked around for two hours and I saw many people shopping for Halloween

Figure 27

Translation of Participant 8's Sentence



BONSOIR Je suis allé au magasin et j'ai ramassé du pain d'oranges de lait et je me suis promené pendant deux heures et j'ai vu beaucoup de gens faire du shopping pour Halloween

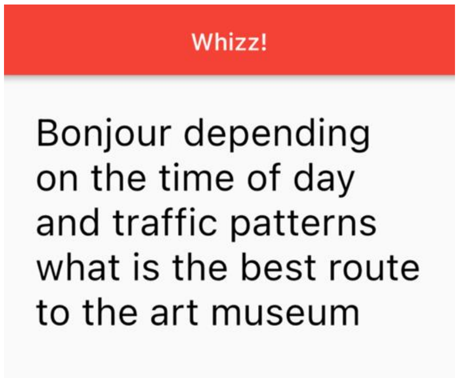
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 26)

Comparing our input with the result in figure 26, we can say that the WER for this speech is 0.

Participant 9: "Depending on the time of day and traffic patterns, what is the best route to the art museum?"

Figure 28

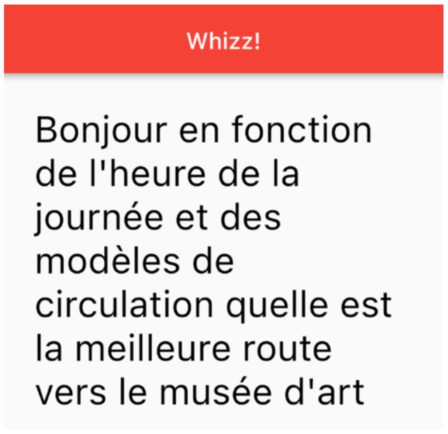
Participant 9's Sentence Captured by the System



Bonjour depending on the time of day and traffic patterns what is the best route to the art museum

Figure 29

Translation of Participant 9's Sentence



Bonjour en fonction de l'heure de la journée et des modèles de circulation quelle est la meilleure route vers le musée d'art

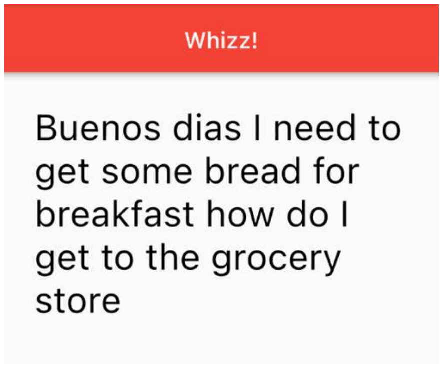
Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 18)

Comparing our input from the participant with the result in figure 28, we can say that the WER for this speech is 0.

Participant 10: "I need to get some bread for breakfast, how do I get to the grocery store?"

Figure 30

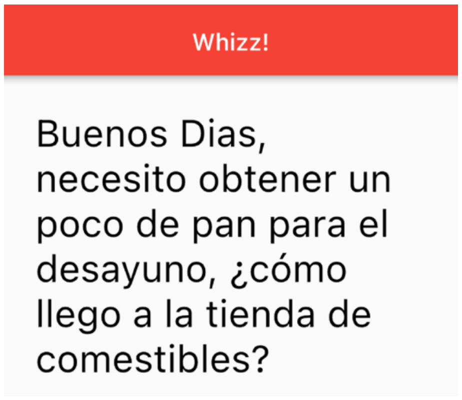
Participant 10's Sentence Captured by the System



Buenos dias I need to get some bread for breakfast how do I get to the grocery store

Figure 31

Translation of Participant 10's sentence



Buenos Dias, necesito obtener un poco de pan para el desayuno, ¿cómo llego a la tienda de comestibles?

Given Word Error Rate, WER = S+I+D/N (where S = 0, I = 0, D = 0, N = 16)

Comparing our input from the participant with the result in figure 30, we can say that the WER for this speech is 0.

5.3 Discussion

From figures 12-31, we notice a greeting at the beginning of each sentence spoken by each participant. This was used as a keyword to switch to the prospective language which the participant intended their sentence to be translated to. Taking our evaluation metrics of Word

Error Rate (WER) into cognizance and calculating for each participant, we can see that there are no Substitutions, Deletions, and Insertions for all sentences by each participant in this test. This gives our average accuracy a value of >95% for recognizing each participant’s words.

A.) RQ-1: How effectively can the program detect surrounding language and translate to that local language?

The results of our usability test demonstrate that with the use of key words such as “Bien” and “Buenos dias”, our application successfully identifies these keywords, switches to Spanish and then subsequently translates the sentence to Spanish. The case is the same for the French language where we employ the keywords “Bonjour” and “Bonsoir”. Our application is able to effectively detect these keywords and translate the sentence to the local language, which in this case is French.

B.) RQ-2: What are the important words used by tourists in emergency situations?

From a close study of the wordlists developed by Laosrirattanachai & Ruangjaroon (2021) and Junnak & Juajamsai’s (2017) paper on High frequency words in Tourism English, we can create a sub-wordlist of the emergency words that fit our earlier definition.

These words are: “danger”, “report”, “battery”, “secure”, “safe”, “assist”, “medical”, “emergency”, “evacuate”, “exit”.

5.4 Surveys and Feedback

At the end of our usability test, we sent out surveys to our participants to provide their thoughts and impressions on the application. We had a set of questions which we asked each participant, and their answers were collated. These are the questions below:

1. Were the instructions on how to use the app clear enough?
2. Would you use the application in real life?
3. What can be improved upon?
4. Were you satisfied with the functionality of the app?
5. What features would you like added to the app?

We got the following feedback for some of our questions

Figure 32

Bar Chart of Responses to First Survey Question

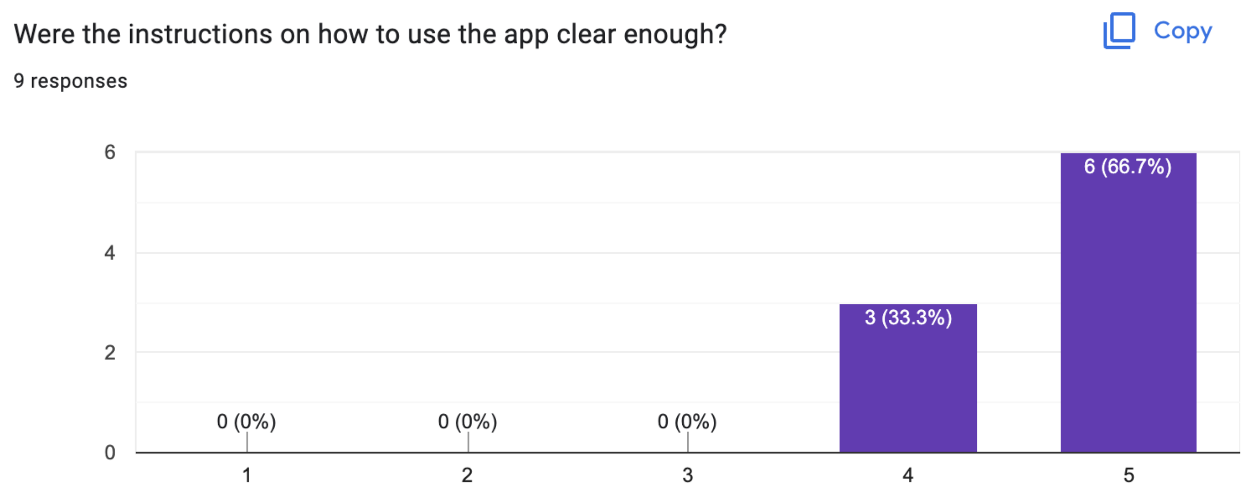
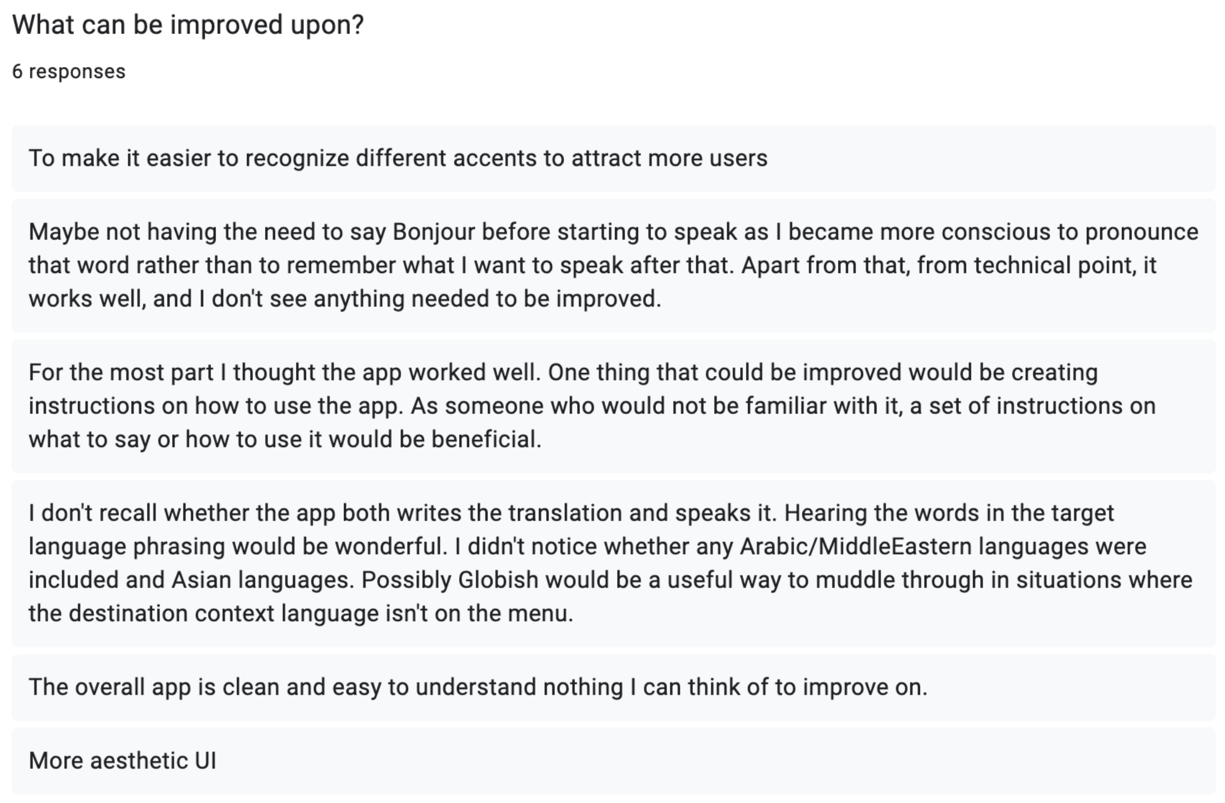


Figure 33

Responses to the Third Survey Question



What can be improved upon?

6 responses

To make it easier to recognize different accents to attract more users

Maybe not having the need to say Bonjour before starting to speak as I became more conscious to pronounce that word rather than to remember what I want to speak after that. Apart from that, from technical point, it works well, and I don't see anything needed to be improved.

For the most part I thought the app worked well. One thing that could be improved would be creating instructions on how to use the app. As someone who would not be familiar with it, a set of instructions on what to say or how to use it would be beneficial.

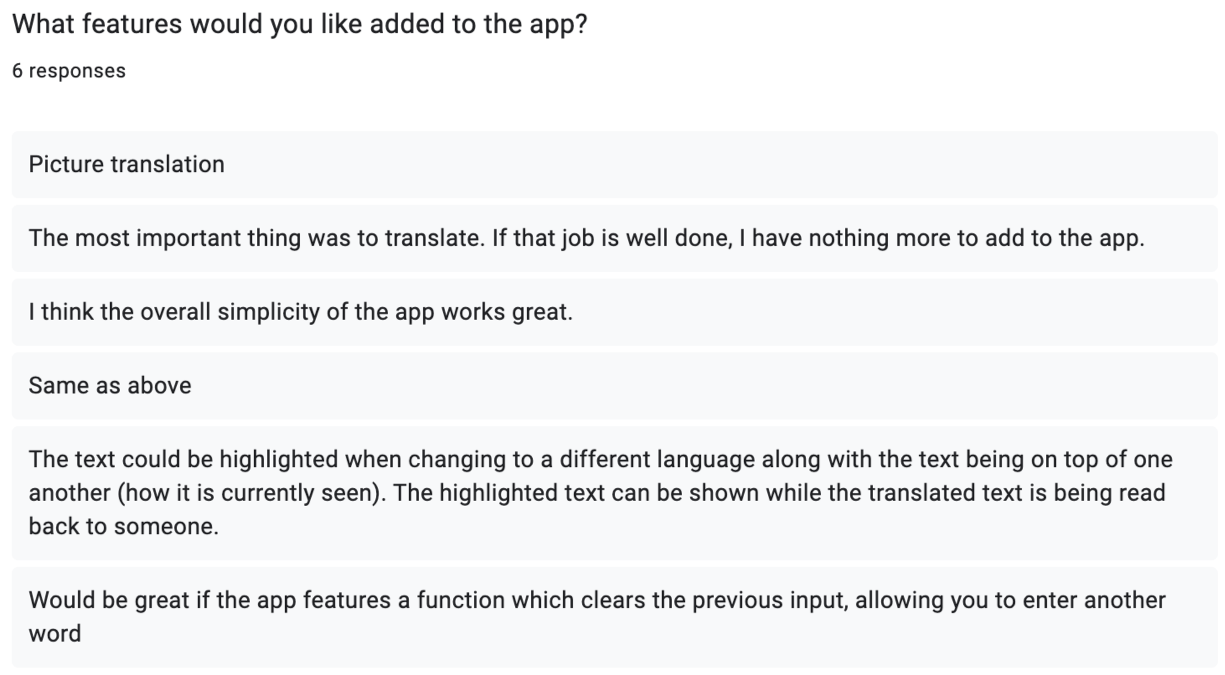
I don't recall whether the app both writes the translation and speaks it. Hearing the words in the target language phrasing would be wonderful. I didn't notice whether any Arabic/MiddleEastern languages were included and Asian languages. Possibly Globish would be a useful way to muddle through in situations where the destination context language isn't on the menu.

The overall app is clean and easy to understand nothing I can think of to improve on.

More aesthetic UI

Figure 34

Responses to the Fifth Survey Question


What features would you like added to the app?

6 responses

Picture translation

The most important thing was to translate. If that job is well done, I have nothing more to add to the app.

I think the overall simplicity of the app works great.

Same as above

The text could be highlighted when changing to a different language along with the text being on top of one another (how it is currently seen). The highlighted text can be shown while the translated text is being read back to someone.

Would be great if the app features a function which clears the previous input, allowing you to enter another word

5.5 Scalability and Accuracy

The results show that the accuracy of the application is >95%. This means that when it comes to detecting speech, the application is quite robust and effective. The application's scalability is also very high because we can extend this to a number of languages such as Portuguese, Arabic, Igbo, Yoruba, German, Swahili, Russian etc. Given the effectiveness the application has shown in detecting keywords, we anticipate that it can cater to more languages.

Chapter 6: Conclusions and Future Discussions

6.1 Conclusion

Speech recognition finds it usage in numerous instances and the technology behind it is continuing to improve. In this project, we built an application for detecting local languages, switching to that local language, and easing conversations by providing translation services. This was facilitated by the speech recognition software and machine translation. We made use of the Speech-to-Text package offered as a Flutter package to develop our speech recognition capability. We then utilized the Google translate package to perform our translation ability. This was done using the Google Neural Machine Translation engine which is based off of the Neural Machine Translation technology.

6.2 Future Discussions

In our project, we faced some limitations that were due to the lack of presently available technologies. We were limited to a number of keywords which we could use for language detection, and this was due to the unavailability of a multilingual speech recognizer. Our speech recognizer primarily recognized words that were present in the English dictionary. The keywords we were able to recognize were popular salutation words that overlapped into the English dictionary or were borrowed from other languages. We also observed a dearth in the amount of research done in Tourism English words. This was evidenced in our second research question where we were able to generate just a list of few words that could be used in emergencies. The functionality of our application can be further improved when we have a system capable of multilingual speech recognition.

References

Anazawa, R., Ishikawa, H., Park, M. J., & Kiuchi, T. (2012). Preliminary study of online machine translation use of nursing literature: Quality evaluation and perceived usability. *BMC Research Notes*, *5*(1), Article 1. https://doi.org/10.1186/1756-0500-5-635

Brown, P., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., & Rossin, P. (1990). A statistical approach to machine translation. *Computational Linguistics*, *16*(2), pp. 76–85.

Chung, T., & Nation, P. (2004). Identifying technical vocabulary. *System*, *32*(2), 251-263.

Cohen, E., & Cooper, R. L. (1986). Language and tourism. *Annals of Tourism Research*, *13*(4), 533-563.

Cobb, T. (2018, June 22). *Web Vocabprofile*. [Online program]. http://www.lextutor.ca/vp/

Coxhead, A. (2000). A new academic word list. *TESOL Quarterly*, *34*(2), 213-238.

Google Translate. (2022, September 19). In *Wikipedia.* https://en.wikipedia.org/w/index.php?title=Google_Translate&oldid=1111081295

Graham, D. (2018). *Key-BNC* [Software]. http://crs2.kmutt.ac.th/Key-BNC/.

Groves, M., & Mundt, K. (2015). Friend or foe? Google translate in language for academic purposes. *English for Specific Purposes*, *37*, 112-121.

Holmes, S.W. (1998). Our Addiction to Attributive Adjective. *Etc*, *55*(3), 299-302.

Junnak, C., & Juajamsai, L. (2017). High frequency key words in tourism English in newspapers: A corpus-based approach. *The Asian Conference on Arts & Humanities*.

Laosrirattanachai, P., & Ruangjaroon, S. (2021). Corpus-Based creation of tourism, hotel, and airline business word lists. *LEARN Journal: Language Education and Acquisition Research Network*, *14*(1), 50-86.

Machine Translation. (n.d.). Microsoft Translator for Business. Retrieved October 1, 2022, from https://web.archive.org/web/20190902184351/https://www.microsoft.com/en-us/translator/business/machine-translation/

Martinez, I., Beck, S., & Panza, C. (2009). Academic vocabulary in agriculture research articles: A corpus-based study. *English for Specific Purposes*, *28*(3), 183-198.

Microsoft translator. (2022, August 7). In *Wikipedia*. *https://en.wikipedia.org/w/index.php?title=Microsoft_Translator&oldid=1102879309*

Puangthong, J. (2015). *Effects of Google Translate on English-Thai Translation in Students majoring in International Communication (English Program), Faculty of Arts, Rajamangala University of Technology Suvarnabhumi*. [Unpublished dissertation], Rajamangala University of Technology Suvarnabhumi, Nonthaburi, Thailand.

Quirk, R., Greenbaum, S., Leech, G., & Svartvik, J. (1985). A comparative grammar of the English language. *New York: Longman Group*.

Tongpoon-Patanasorn, A., & Griffith, K. (2020). Google Translate and translation quality: A case of translating academic abstracts from Thai to English. *PASAA: Journal of Language Teaching and Learning in Thailand*, *60*, 134-163.

Translator Text API. (2019). Microsoft Translator for Business. Retrieved October 1, 2022, from https://web.archive.org/web/20190902184414/https://www.microsoft.com/en-us/translator/business/translator-api/

Turovsky, B. (2016). Found in translation: More accurate, fluent sentences in Google Translate. *Google*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv preprint.* https://doi.org/10.48550/arXiv.1609.08144

Appendix A: Application Code

```
import 'package:flutter/material.dart';
import 'package:speech_to_text/speech_to_text.dart' as stt;
import 'package:avatar_glow/avatar_glow.dart';
import 'package:highlight_text/highlight_text.dart';
import 'package:translator/translator.dart';
import 'package:flutter_tts/flutter_tts.dart';
import 'package:geolocator/geolocator.dart';
import 'package:geocoding/geocoding.dart';
import 'dart:async';


void main() {
  runApp(MyApp());
}


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Whizz',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.red,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: SpeechScreen(),
    );
  }
}


class SpeechScreen extends StatefulWidget {

  @override
  _SpeechScreenState createState() => _SpeechScreenState();
}

class _SpeechScreenState extends State<SpeechScreen> {
  GoogleTranslator translator = GoogleTranslator();
  FlutterTts tts = FlutterTts();


  String langCode = "";

  translate() async{
    await translator.translate(_text, to: langCode).then((output) {
      setState(() {
        _text = output.toString();
      });
    });
  }
  speak() async{
    await tts.setLanguage(langCode);
    await tts.setPitch(1.0);
    await tts.speak(_text);
  }

  final Map<String, HighlightedWord> _highlights = {
    'flutter': HighlightedWord(
      onTap: () => print('flutter'),
      textStyle: const TextStyle(
        color: Colors.blue,
        fontWeight: FontWeight.bold,
        fontSize: 32.0,
      ),
    ),
  };

  late stt.SpeechToText _speech;
  bool _isListening = false;
  String _text = 'Press the button and start speaking';
  double _confidence = 1.0;

  getLocation() async{
    LocationPermission permission;
    permission = await Geolocator.checkPermission();
    permission = await Geolocator.requestPermission();
    if( permission== LocationPermission.denied){
      //nothing
    }
    Position position = await Geolocator
        .getCurrentPosition(desiredAccuracy: LocationAccuracy.medium);

    List<Placemark> placemarks = await
placemarkFromCoordinates(position.latitude, position.longitude);
    print(position.latitude);
    print(position.longitude);
    print(placemarks[0].locality.toString());
  }

  @override
  void initState() {
    _speech = stt.SpeechToText();
    getLocation();
    initialListen();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Whizz!'),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
      floatingActionButton: AvatarGlow(
        endRadius: 75.0,
        animate: _isListening,
        glowColor: Theme.of(context).primaryColor,
        duration: const Duration(milliseconds: 200),
        repeatPauseDuration: const Duration(milliseconds: 100),
        repeat: true,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            FloatingActionButton(
              onPressed: _listen,
              child: Icon(_isListening ? Icons.mic: Icons.mic_none),
            ),
            FloatingActionButton(
              onPressed: (){
                translate();
                Future.delayed(Duration(minutes: 0, milliseconds: 800), (){
                  speak();
                });
              },
              child: Icon(Icons.translate),
            ),
          ],
        ),
      ),
      body: SingleChildScrollView(
        reverse: true,
        child: Container(
          padding: const EdgeInsets.fromLTRB(30.0, 30.0, 30.0, 150.0),
          child: TextHighlight(
            text: _text,
            words: _highlights,
            textStyle: const TextStyle(
              fontSize: 32.0,
              color: Colors.black,
              fontWeight: FontWeight.w400,
            ),
          ),
        ),
      ),
    );
  }



  void _listen() async {
    if (!_isListening) {
      bool available = await _speech.initialize(
        onStatus: (val) => print('onStatus: $val'),
        onError: (val) => print('onError: $val'),
      );
      if (available) {
        setState(() => _isListening = true);
        _speech.listen(
          onResult: (val) => setState(() {
            _text = val.recognizedWords;
```

```
              if (val.hasConfidenceRating && val.confidence > 0) {
                _confidence = val.confidence;
              }
            }),
          );
        }
      } else {
        setState(() => _isListening = false);
        _speech.stop();
      }
    }

    /// Determine the current position of the device.

    Future<Position> _determinePosition() async {
      bool serviceEnabled;
      LocationPermission permission;

      serviceEnabled = await Geolocator.isLocationServiceEnabled();
      if (!serviceEnabled) {

        return Future.error('Location services are disabled.');
      }

      permission = await Geolocator.checkPermission();
      if (permission == LocationPermission.denied) {
        permission = await Geolocator.requestPermission();
        if (permission == LocationPermission.denied) {

          return Future.error('Location permissions are denied');
        }
      }

      if (permission == LocationPermission.deniedForever) {
        return Future.error(
            'Location permissions are permanently denied, we cannot request
  permissions.');
      }


      return await Geolocator.getCurrentPosition();
    }

    bool listening1 = false;

    initialListen() async {
      if (!listening1) {
        bool available = await _speech.initialize(
          onStatus: (val) => print('onStatus: $val'),
          onError: (val) => print('onError: $val'),
        );
        if (available) {
          setState(() => listening1 = true);
          _speech.listen(
            listenFor: Duration(seconds: 40),
          onResult: (val) =>
              setState(() {
                _text = val.recognizedWords;
                if (_text.contains("Hola") || _text.contains("Buenos dias")
|| _text.contains("buenos dias")) {
                  setState((){langCode = "es";
                  print(langCode);});
                }
                else if (_text.contains("Bonjour")) {
                  setState((){langCode = "fr";
                  print(langCode);});
                } else {
                  setState((){langCode = "en";
                  print(langCode);});
                }
                if (val.hasConfidenceRating && val.confidence > 0) {
                  _confidence = val.confidence;
                }
              }),
        );
      }
      else {
        setState(() => listening1 = false);
        _speech.stop();
      }
    }
  }



}
```

Appendix B: Pubspec.yaml

```yaml
version: 1.0.0+1

environment:
  sdk: ">=2.17.6 <3.0.0"


dependencies:
  flutter:
    sdk: flutter


  cupertino_icons: ^1.0.2
  speech_to_text: ^5.6.1
  avatar_glow: ^2.0.2
  highlight_text: ^1.4.1
  translator: ^0.1.7
  flutter_tts: ^3.5.3
  geolocator: ^9.0.1
  geocoding: ^2.0.5



dev_dependencies:
  flutter_test:
    sdk: flutter


  flutter_lints: ^2.0.0



flutter:


  uses-material-design: true
```

Appendix C: Info.plist

```
    <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>CFBundleDevelopmentRegion</key>
   <string>$(DEVELOPMENT_LANGUAGE)</string>
   <key>CFBundleDisplayName</key>
   <string>Whizz</string>
   <key>CFBundleExecutable</key>
   <string>$(EXECUTABLE_NAME)</string>
   <key>CFBundleIdentifier</key>
   <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
   <key>CFBundleInfoDictionaryVersion</key>
   <string>6.0</string>
   <key>CFBundleName</key>
   <string>whizz</string>
   <key>CFBundlePackageType</key>
   <string>APPL</string>
   <key>CFBundleShortVersionString</key>
   <string>$(FLUTTER_BUILD_NAME)</string>
   <key>CFBundleSignature</key>
   <string>????</string>
   <key>CFBundleVersion</key>
   <string>$(FLUTTER_BUILD_NUMBER)</string>
   <key>LSRequiresIPhoneOS</key>
   <true/>
   <key>UILaunchStoryboardName</key>
   <string>LaunchScreen</string>
   <key>UIMainStoryboardFile</key>
   <string>Main</string>
   <key>UISupportedInterfaceOrientations</key>
   <array>
      <string>UIInterfaceOrientationPortrait</string>
      <string>UIInterfaceOrientationLandscapeLeft</string>
      <string>UIInterfaceOrientationLandscapeRight</string>
   </array>
   <key>UISupportedInterfaceOrientations~ipad</key>
   <array>
      <string>UIInterfaceOrientationPortrait</string>
      <string>UIInterfaceOrientationPortraitUpsideDown</string>
      <string>UIInterfaceOrientationLandscapeLeft</string>
      <string>UIInterfaceOrientationLandscapeRight</string>
   </array>
   <key>NSMicrophoneUsageDescription</key>
   <string>This application needs to access your microphone</string>
   <key>NSSpeechRecognitionUsageDescription</key>
   <string>This application needs the speech recognition </string>
   <key>UIViewControllerBasedStatusBarAppearance</key>
   <key>CADisableMinimumFrameDurationOnPhone</key>

   <key>NSLocationWhenInUseUsageDescription</key>
    <string>This app needs access to location when open.</string>
    <key>NSLocationAlwaysUsageDescription</key>
    <string>This app needs access to location when in the
background.</string>
</dict>
</plist>
```

Appendix D: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.whizz">

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
   <application
        android:label="whizz"
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smalles
tScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- Specifies an Android theme to apply to this Activity as soon
as
                 the Android process has started. This theme is visible to
the user
                 while the Flutter UI initializes. After that, this theme
continues
                 to determine the Window background behind the Flutter UI. --
>
            <meta-data
              android:name="io.flutter.embedding.android.NormalTheme"
              android:resource="@style/NormalTheme"
              />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

Appendix E: WordLists

The 1st sub-TBWL

accommodate	culture	lounge	tour
airport	depart	luxury	transport
arrive	executive	offer	travel
atmosphere	express	private	trip
available	guest	region	visit
bay	holiday	relax	welcome
book	international	request	
capital	ideal	reserve	
convenience	journey	serve	
cuisine	leisure	service	

The 2nd sub-TBWL

amaze	climb	forest
animal	country	found
art	delicious	garden
autumn	discover	green
bar	double	hall
bath	east	hill
beach	experience	history
camp	fair	hour
church	farm	huge
city	fly	internet

The 3rd sub-TBWL

island	plenty	shop
lake	reach	spacious
local	rent	spot
mileage	rich	spring
mountain	river	square
nature	rock	station
outdoor	safe	stay
park	sail	stone
place	sea	store
plan	secure	street

The 4th sub-TBWL

town	advance	classic
track	adventure	cliff

train	attend	coast
view	attract	contact
weather	beer	decorate
wide	brick	delight
wild	bridge	desert
woods	cable	distance
access	castle	district
admire	century	dive

The 5th sub-TBWL

environment	gather	incredible
event	giant	invite
expense	gift	modern
famous	golf	mount
fantastic	gorgeous	mud
fascinate	guide	official
feature	hire	organize
flag	hotel	path
folk	impress	period
gate	improve	pine

The 6th sub-TBWL

planet	scenic	theatre
policy	schedule	ticket
pool	season	tip
popular	shelter	tower
port	shore	tradition
rare	site	traffic
royal	surroundings	typical
restaurant	ski	valley
recommend	solid	various
saint	tent	vehicle

The 7th sub-TBWL

village	budget	currency
wander	capacity	dedicate
abroad	carve	display
agriculture	celebrate	diverse
archaeological	ceremony	domestic
airline	concert	era
ancient	conservation	enhance

annual	cruise	emergency
architect	craft	exhibit
border	contemporary	explore

The 8th sub-TBWL

extraordinary	host	numerous
facilitate	ingredient	museum
fee	inhabitant	ocean
festival	insight	overlook
gallery	inspire	pace
goods	interior	palace
harbor	landscape	palm
heritage	legend	participate
highlight	lodge	passion
holy	marine	peak

The 9th sub-TBWL

platform	rural	unique
preserve	sculpture	urban
prior	stun	vast
province	summit	arch
remote	symbol	array
republic	territory	authentic
resort	theme	avenue
resource	trail	bathe
retreat	treasure	bronze
route	ultimate	café

The 10th sub-TBWL

Calendar	harvest	recreation
canal	infrastructure	refresh
cathedral	indigenous	romance
cave	inn	sacred
dynamic	marble	spectacular
escort	magnificent	soak
exotic	monument	statue
ferry	mineral	steep
fort	overnight	stroll
habitat	pearl	superb

The 11th sub-TBWL

temple	hike	renown

terrace	iconic	sanctuary
tropical	inland	spa
venue	jewel	surf
villa	landmark	terrain
volcanic	memorable	transit
altitude	paradise	vacation
courtyard	passport	visa
feast	peninsula	wilderness
globe	renovate	dune

The 12th sub-TBWL

ecosystem	vibrant	panorama
elegance	boutique	gourmet
excursion	canyon	itinerary
flora	culinary	kayak
hospitality	fauna	scuba
hub	lagoon	backpack
lush	majestic	campsite
plateau	motel	breathtaking
refund	pristine	coastline
trek	picturesque	countryside

The 13th sub-TBWL

downtown	wildlife
limestone	wheelchair
nightlife	underwater
underground	waterfall
sightseeing	wellness
sunset	oneway

The 1st sub-HBWL

accommodate	culture	lounge	tour
airport	depart	luxury	transport
arrive	executive	offer	travel
atmosphere	express	private	trip
available	guest	region	visit
bay	holiday	relax	welcome
book	international	request	
capital	ideal	reserve	
convenience	journey	serve	
cuisine	leisure	service	

The 2nd sub-HBWL

amaze	business	dinner
art	centre	discover
bar	check	done
base	club	double
bath	coffee	drink
beach	collect	excite
beauty	comfort	experience
bed	cook	floor
breakfast	couple	garden
bring	delicious	heart

The 3rd sub-HBWL

history	park	stay
indoor	rate	table
inform	rich	treat
island	river	view
lake	rock	wedding
local	room	access
lunch	sign	attract
mountain	spacious	benefit
nature	special	casual
outdoor	square	ceiling

The 4th sub-HBWL

cheese	dish	impress
classic	due	improve
coast	entertain	include
complain	event	incredible
contact	extend	invite
create	favour	item
deck	feature	locate
decorate	golf	modern
design	guide	partner
desk	hotel	pool

The 5th sub-HBWL

property	tradition	corporate
provide	twin	custom
rare	valley	craft
register	valuables	contemporary

restaurant	various	dedicate
scenic	approximate	distinct
shower	architect	diverse
site	award	expand
standard	blend	explore
surround	celebrate	extraordinary

The 6th sub-HBWL

facility	intimate	resort
fee	landscape	retreat
gallery	legend	stun
heritage	menu	sophisticate
ingredient	ocean	trail
host	occupancy	ultimate
innovate	overlook	unique
inquire	passion	vast
inspire	reception	adjacent
interior	reside	array

The 7th sub-HBWL

authentic	marble	suite
boast	premier	terrace
chef	premium	venue
champagne	refine	villa
destination	refresh	balcony
exotic	romance	butcher
fare	signature	cocktail
furnish	sparkling	complimentary
magnificent	stroll	exquisite
laundry	spectacular	gym

The 8th sub-HBWL

sumptuous	bellhop	housekeeper
yoga	appetizer	getaway
unwind	bathroom	lifestyle
utensil	ballroom	medium
cutlery	babysit	sunset
deluxe	bartender	walk-in
valet	breathtaking	wellness
toiletries	busser	wildlife
brunch	doorman	

concierge	fireplace	

The 1st sub-ABWL

accommodate	culture	lounge	tour
airport	depart	luxury	transport
arrive	executive	offer	travel
atmosphere	express	private	trip
available	guest	region	visit
bay	holiday	relax	welcome
book	international	request	
capital	ideal	reserve	
convenience	journey	serve	
cuisine	leisure	service	

The 2nd sub-ABWL

allow	charge	hour
bag	class	inform
baggage	clearance	land
base	comfort	load
board	country	middle
business	danger	mile
call	engine	pass
card	first	prepare
carrier	fly	rate
carry	flyer	report

The 3rd sub-ABWL

responsible	tax	battery
return	weather	bound
rule	window	captain
safe	adult	cart
seat	advance	chief
secure	advice	claim
sign	agent	commit
space	alcohol	deck
special	assist	deliver
store	attendant	direct

The 4th sub-ABWL

distance	log	port
duty	loss	provide
economy	mask	refuse
gate	medical	recommend
identify	minor	remain
individual	operate	require
instruct	organize	respect
instrument	pat	row
item	plane	schedule
length	pocket	seal

The 5th sub-ABWL

senior	annual	corporate
spare	approve	consult
ticket	assign	crew
tower	capacity	cruise
traffic	charter	currency
transfer	climate	custom
tray	companion	delay
accompany	code	device
aircraft	consent	disabled
airline	compensate	domestic

The 6th sub-ABWL

electronic	pilot	cabin
infant	proceed	cargo
excess	prohibit	carriage
facilitate	republic	comply
emergency	restrict	destination
jet	route	duration
liquid	specify	escort
numerous	update	evacuate
passenger	weigh	exit
permit	zone	fare

The 7th sub-ABWL

fleet	aisle	notify
haul	altitude	passport
immigrate	automate	ramp
overhead	aviation	runway
premium	brace	transit

strap	compartment	turbulent
tag	congestion	vent
terminal	fasten	visa
upgrade	disarm	cockpit
airways	complimentary	hub

The 8th sub-ABWL

luggage	purser	layover
portable	decompress	legroom
refund	airbus	onboard
galley	airside	pregnant
aft	armrest	seatback
itinerary	copilot	seatbelt
lavatory	headset	takeoff
recline	inflight	taxiway
disembark	jumpseat	wheelchair
deplane	landside	