Acknowledgements

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project work. I would like to take this opportunity to thank them all.

First and foremost I would like to thank Dr. N. V. R. Naidu, Principal, Ramaiah Institute of Technology, Bengaluru, for his moral support towards completing my project work.

I would like to thank Dr. Vijaya Kumar B P, Head of Department, Information Science of Engineering, Ramaiah Institute of Technology, Bengaluru, for his valuable suggestions and expert advice.

I deeply express my sincere gratitude to my guide Ms. Rajeshwari S. B, Assistant Professor, Department of Information Science and Engineering, Ramaiah Institute of Technology, Bengaluru, for her able guidance, regular source of encouragement and assistance throughout this project.

My thanks to all the Faculty members of the Department of Information Science and Engineering for their constant support and encouragement.

My sincere gratitude to my family for their continuous support in all aspects of my life.

Last, but not the least, I would like to thank my peers and friends who provided me with valuable suggestions to improve my project.

Abstract

Technology has reached a point where conversational agents, in the form of Home Assistants and Intelligent Chatbots, are capable of understanding and responding to human speech. Due to their widespread use around the globe, especially in a multilingual country like India, such assistants are programmed to be capable of understanding speech and conversing for an array of languages. The ideal objective of the assistant is to understand user intent and respond with human expression, achieving a great degree of accuracy and efficiency. However, a multilingual society such as India has grown quite comfortable using a combination of two or more languages while communicating, commonly referenced as code switching. Code switching is not inherently taken care of by today's intelligent assistants. Hence, users that replace certain subsets of a language's vocabulary with another language when conversing end up being misinterpreted and hence fail to achieve desired response. In lieu of this, it is identified that the underlying problem is in the inability of the intelligent assistant to understand code mixed speech input.

This project, therefore, specifically deals with vernacular code switching in Hindi-English switches. Due to the lack of standardized dataset, the project includes corpus generation of code-switched frequently asked queries was generated. The project then includes a study of intent classification of code-switched queries with various vectorizers and classifier combinations. The project further comprises a pipeline to identify the user intent and provide desired response using keyword extraction and named entity recognition models on code switched queries.

Standardized code-switching metrics applied on the generated corpus to estimate the strength of code-switching yielded high intensity. The results of the intent classification study on code-switched queries obtained by evaluating the models on standardized classification metrics have yielded high accuracy. Named entity recognition in code-switched queries also have yielded high precision and recall scores. Through this

dissertation work a feasible scope is shown for understanding code mixed input in Hindi-English and Kannada-English switches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this epoch of digital technology, the rise of intelligent assistants is eminent as they are powered by advancements in domains of artificial intelligence, machine language and natural language understanding. Individuals find using assistants convenient as all they have to do is make a query while the assistant undergoes the process of understanding the user intention and providing desired responses to the user. Currently, intelligent assistants are capable of handling varied languages and are very effective in providing services to users.

Considering linguistically diverse countries like India, Russia, Switzerland and many others that have multilingual speakers that converse in second language or third language too as a part of everyday life, it is natural for such speakers to have a tendency to switch languages or use multiple elements from different languages to convey their thoughts in a single context of conversation, which is called code switching.

Therefore though intelligent assistants are programmed to be capable of understanding human speech and conversing in an array of languages, code switching is not inherently taken care of by today's intelligent assistants. Hence, speakers that replace certain subsets of a language's vocabulary with another language when conversing end up being misinterpreted by the assistant and hence fail to achieve desired response. Therefore this project aims at providing facilitation to Indian multilingual speakers that query in code switched requests of Hinglish (Hindi-English) and achieving desired response from the intelligent assistant.

1.2 Constraints and Requirements

- **Identifying the code-switched regions**

There are no fixed regions where code-switching tends to occur. Certain speakers have a tendency to code mix while conversing where they replace certain subsets of a language's vocabulary with another language in a single sentence while other speakers code switch languages after a few sentences or a paragraph. The speakers code-switch according to their comfort, consciously or unconsciously, or to express particular ideas clearly and hence the switch can happen any time during the conversation.

- **Ambiguity in dialects**

Different dialects within the same language can lead to ambiguities in identifying regions of code switching.

our

- **Unavailability of Code-switched FAQ dataset**

Non-availability of open-sourced code-switched FAQ datasets with labeled intents. Code-switched dataset for the purpose of intent classification and querying is not generalized, hence it is not available for this application.

- **Utterance and pronunciation of Non-English words in code-switched text**

The words in English are defined and are strictly followed in case of text and speech, while the same is not expected in regional languages which varies from dialect and utterance. These ambiguities also contribute to spelling errors when considered in the form of text.

1.3 Problem Statement

This project aims at providing facilitation to Indian multilingual speakers by handling vernacular code switched requests made by users while conversing with intelligent assistants and providing desired responses to the users. This project additionally throws insights on performance studies of cross-lingual models used in the pipeline. This project also includes generation of corpus of code-switched Hinglish frequently asked questions (FAQs) and its statistics.

1.4 Scope and Objectives

1.4.1 Scope

- **Home Assistants and Chatbots:**

Build a robust intelligent assistant that can interact with a larger user base consisting of people who interact primarily in code mixed speech.

- **Preprocessing aid:**

Handling the issue of code mixed language data that are difficult to be processed and analysed, by providing a preprocessing solution to make it comprehensible to any generic algorithms of intelligent systems performing various NLP tasks.

- **Scaling:**

The project can set-up experiments for similar tasks with a slight variation in the languages dealt with, namely Kannada-English code switching, and Hindi-English code switching.

1.4.2 Objectives

The core objective of the project focuses on the task of building a robust vernacular voice assistant that can handle code switched queries made by Indian multilingual users. It involves sub-tasks such as -

- Building a Hinglish FAQs corpus.
- Intent classification of the code switch queries.
- Named entity recognition of Hinglish code switched queries.
- Identifying the intent of the user and providing desired response.

1.5 Proposed Model

The user provides code-switched query as input either in speech or text format. This query is then passed to a supervised classifier to identify the type of intent. If the intent requires an action to be taken, then the query is passed to the code-switched named entity recognition model to identify entities such as Date, Location, Restaurant names, Cuisines, Contact details etc whereas other queries for getting information are passed to the keyword extractor and reordering process to identity the words contributing to interpreting the intent of the user. Once the intent is identified from the code switched query, the desired response is provided to the user.

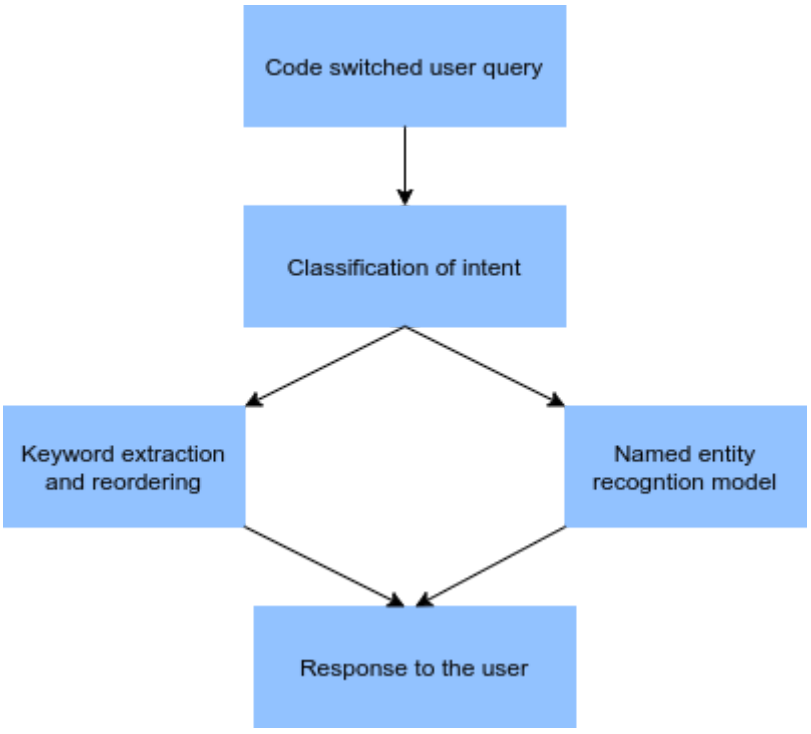

Figure 1 - Overview of proposed models

1.6 Organisation of Report

In order to explain the developed system, the following sections are covered:

- **Literature Review** describes the study of the existing systems and techniques taken into account prior to development of the proposed system.
- **System Analysis and Design** provides a detailed walk through of the software engineering methodology adopted to implement the model, an overview of the system and the modules incorporated into the system.

- **Modelling and Implementation** provides a deeper insight into the working of the model. The various modules and their interactions are depicted using relevant descriptive diagrams.
- **Testing** the model to ensure bug/error free model along with the **Results** obtained. **Discussion** then provides detailed analysis on quality assurance measures.
- **Conclusion** about the Results obtained after successfully running the model and **Future Scope** of the model is highlighted.

Chapter 2

Literature Review

Table 1 - Literature review

Papers referred	Authors	Inferences
GLUECoS : An Evaluation Benchmark for Code-Switched NLP	Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram,	• Presents an evaluation benchmark, GLUECoS, for code-switched languages, that spans several NLP tasks in English-Hindi.

	Monojit Choudhury	
Speech Recognition on Code-Switching Among the Chinese Dialects.	Dau-cheng Lyu, Ren-yuan Lyu, Yuang-chin Chiang, & Chun-nan Hsu.	• Proposes an integrated approach to do automatic speech recognition on code-switching utterances, where speakers switch back and forth between at least two languages.
Defining Code Switching. In: The Syntax of Arabic and French Code Switching in Morocco.	Aabi M	• This paper provides a general introduction to code switching, explains the need and scope of the topic
Analyzing Code-Switching Rules for English–Hindi Code-Mixed Text.	Mahata S.K., Makhija S., Agnihotri A., Das D.	• Proposes an efficient and less resource intensive strategy for parsing and analyzing switching points in code-mixed data. • It explores the rules of code-switching in Hindi–English code-mixed data. • The work involves code-mixed text extraction, translation of the extracted texts to its pure form, forming word pairs, annotation of these using Parts-of-Speech tags.
Transfer Learning for Detecting Hateful Sentiments in Code Switched Language.	Rajput K., Kapoor R., Mathur P., Hitkul, Kumaraguru P., Shah R.R.	• Machine transliteration can be employed for converting the code-switched text into a singular script but poses the challenge of the semantical breakdown

		of the text. ● To overcome this drawback, this chapter investigates the application of transfer learning.
Automatic speech recognition of Cantonese-English code-mixing utterances	Chan, Joyce Y. C. / Ching, P. C. / Lee, Tan / Cao, Houwei.	● Describes recent work on the development of a large vocabulary, speaker-independent, continuous speech recognition system for Cantonese-English code-mixing utterances.
Multi-Task Learning in Deep Neural Networks for Mandarin-English Code-Mixing Speech Recognition.	Chen, M., Pan, J., Zhao, Q., & Yan, Y.	● Proposes a multi-task learning deep neural networks approach that has been proven to be effective for acoustic modeling in speech recognition which is applied to Mandarin-English code-mixing speech recognition.
Acoustic data augmentation for Mandarin-English code-switching speech recognition.	Long, Y., Li, Y., Zhang, Q., Wei, S., Ye, H., & Yang, J.	● Focuses on acoustic data augmentation for the Mandarin-English code switched speech recognition task. ● Proposes a code switched acoustic event detection system based on the deep neural network to extract real code-switching speech segments automatically.
Part-of-Speech Tagging for Code-Mixed	Anupam Jamatia, Björn Gambäck, Amitava Das.	● The paper reports work on collecting and annotating code-mixed English-Hindi

English-Hindi Twitter and Facebook Chat Messages		social media text (Twitter and Facebook messages), and experiments on automatic tagging of these corpora, using both a coarse-grained and a fine-grained part-of-speech tag set.
Named Entity Recognition for Code Mixing in Indian Languages using Hybrid Approach	Rupal Bhargava, Bapiraju Vamsi Tadikonda, Yashvardhan Sharma	● It proposes the optimistic exponential type of ordered weighted averaging (OWA) operator as a hybrid recommender system.
Challenges of Computational Processing of Code-Switching	Özlem Çetinoğlu, Sarah Schulz, Ngoc Thang Vu.	● Addresses the challenges of Natural Language Processing tasks such as normalisation, language identification, language modelling, part-of-speech tagging and dependency parsing ● It also reports more downstream ones such as machine translation and automatic speech recognition.
Text normalization in code-mixed social media text.	S. Dutta, T. Saha, S. Banerjee and S. K. Naskar.	● Addresses the problem of text normalization, an often overlooked problem in natural language processing, in code-mixed social media text. ● The objective of the work presented here is to correct English spelling errors in code-mixed social media text. ● It employs a CRF based machine

		learning approach followed by post-processing heuristics for the word-level language identification task.
A language identification system for code-mixed English-Manipuri Social Media text.	P. Lamabam and K. Chakma.	• Deals with issues of language identification in code mixed texts. Automatic language identification for the code-mixed social media texts is a challenging task. • A word-level language identification experiment is performed using a Trigram-based and Conditional Random Field (CRF)-based model.
Intent Detection for code-mix utterances in task oriented dialogue systems	Pratik Jayarao, Aman Srivastava	• Proposes an intent detection approach using supervised classifier models for code-switched data. • Provides an extensive comparison between various vector models, encodings and classifiers based on their performance on code-switched data.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova	● Proposes an approach of a trained model on 104 languages from Wikipedia. It uses a shared 110k WordPiece vocabulary. ● It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models
Speech and Language Processing (3rd ed. draft)	Dan Jurafsky and James H. Martin	● Provides an insight into basics of POS tagging,Context free grammar, and sentence construction rules for imperative and interrogative sentences

Chapter 3

System Analysis and Design

3.1 Corpus generation

Due to the lack of standardized code-switched frequently asked question datasets involving Hindi-English switches, the project involves creation of such a corpus.

3.1.1 Corpus statistics

A corpus of about 1000 Hinglish and 750 Kanglish code-switched frequently asked questions was created that included code-switched queries on various domains such as :

- Delivery queries with queries on order delivery details, payment details, refund details.
- Insurance queries with queries related to health, travel and motor insurance.
- Aadhaar queries with queries related to E-aadhaar, updation of aadhaar card and other general queries.
- Medical queries with queries on symptoms, treatment and prevention of diseases.
- Find nearest queries with queries to find the nearest location.
- Booking queries which includes queries for booking a hotel, flight, bus, train or a table at a restaurant and enquiry queries on vacation details.
- Reminder queries to note, set an alarm or reserve calendar slots.

Table 2 : Details of Hinglish corpus collection

Class	**Number of samples**	**Number of unique tokens in Hindi**	**Number of unique tokens in English**
Delivery queries	170	101	118
Insurance queries	155	74	181

Aadhaar queries	155	109	161
Medical queries	175	111	156
Find nearest queries	100	55	107
Reminder queries	100	84	138
Booking queries	150	160	199

Table 2 : Details of Kanglish corpus collection

Class	**Number of samples**	**Number of unique tokens in Kannada**	**Number of unique tokens in English**
Delivery queries	160	105	110
Aadhaar queries	152	111	149
Medical queries	175	114	153
Find nearest queries	46	27	46
Reminder queries	64	58	92
Booking queries	150	172	187

3.1.2 Code-switching Statistics

Since a new corpus was generated, code-switching statistics of the data in terms of standardized metrics for code-switching are used to validate code switching in the corpus. Some of the standardized code-switched metrics used are -

- **Multilingual Index (M-index)** : A word-count based measure quantifying the inequality of distribution of language tags in a corpus of at least two languages. The M-index is calculated as follows, where $k > 1$ is the total number of languages represented in the corpus, p_j is the total number of words in the language j over the total number of words in the corpus, and j ranges over the languages present in the corpus. The index is bounded between 0 (monolingual corpus) and 1 (each language in the corpus is represented by an equal number of tokens).

$$\textbf{M-Index} = 1 - \Sigma\ p_j^{\,2} / (k-1) \cdot\ p_j^{\,2}$$

- **Language Entropy (LE)** : The bits of information needed to describe the distribution of language tags. language entropy is calculated as

$$\textbf{LE} = -\ \Sigma\ p_j \log^2 (p_j)$$

and is bounded from below by 0 (representing a completely monolingual text) and bounded from above by $\log^2(k)$ which is the maximum entropy for a corpus with k languages (and, in such a case, each language is represented equally). In the case of two languages, the M-index and LE can be derived from one another.

Table 3 : Corpus statistics on code-switching metrics on Hinglish dataset

Intent class	**Multilingual Index (M-index)**	**Language Entropy (LE)**
Delivery queries	0.926	0.972
Insurance queries	0.680	0.858
Aadhaar queries	0.988	0.995
Medical queries	0.979	0.992
Find nearest queries	0.984	0.994
Reminder queries	0.975	0.991
Booking queries	0.873	0.950

Table 4 : Corpus statistics on code-switching metrics on Kanglish dataset

Intent class	**Multilingual Index (M-index)**	**Language Entropy (LE)**
Delivery queries	0.93893	0.9771

Aadhaar queries	0.99979	0.9999
Medical queries	0.97777	0.9918
Find nearest queries	0.7837	0.910
Reminder queries	0.98194	0.9934
Booking queries	0.91118	0.9666

3.2 Project workflow

The user provides a code-switched input either in the form of speech or text. If the query is in speech form, it is converted to text. If the query is in text form, the script of the text is checked. The entire text is to be transliterated to Roman script. This preprocessed text query is then passed to a supervised classifier to identify the type of intent. If the intent requires an action to be taken, then the query is passed to the code-switched named entity recognition model to identify entities such as Date, Location, Restaurant names, Cuisines, Contact details etc and then accordingly an action is performed by the assistant, whereas if an intent is one that requires information to be retrieved based on an user text input, it is passed through a sequence of steps that includes stopword removal, keyword extraction based on POS tags, language identification and translation to target language (English) and the reordering process to identity the words that contribute towards interpreting the intent of the user. Once the intent is identified from the search based code switched query, a query is passed to Google and the relevant results are web scraped and provided to the user.

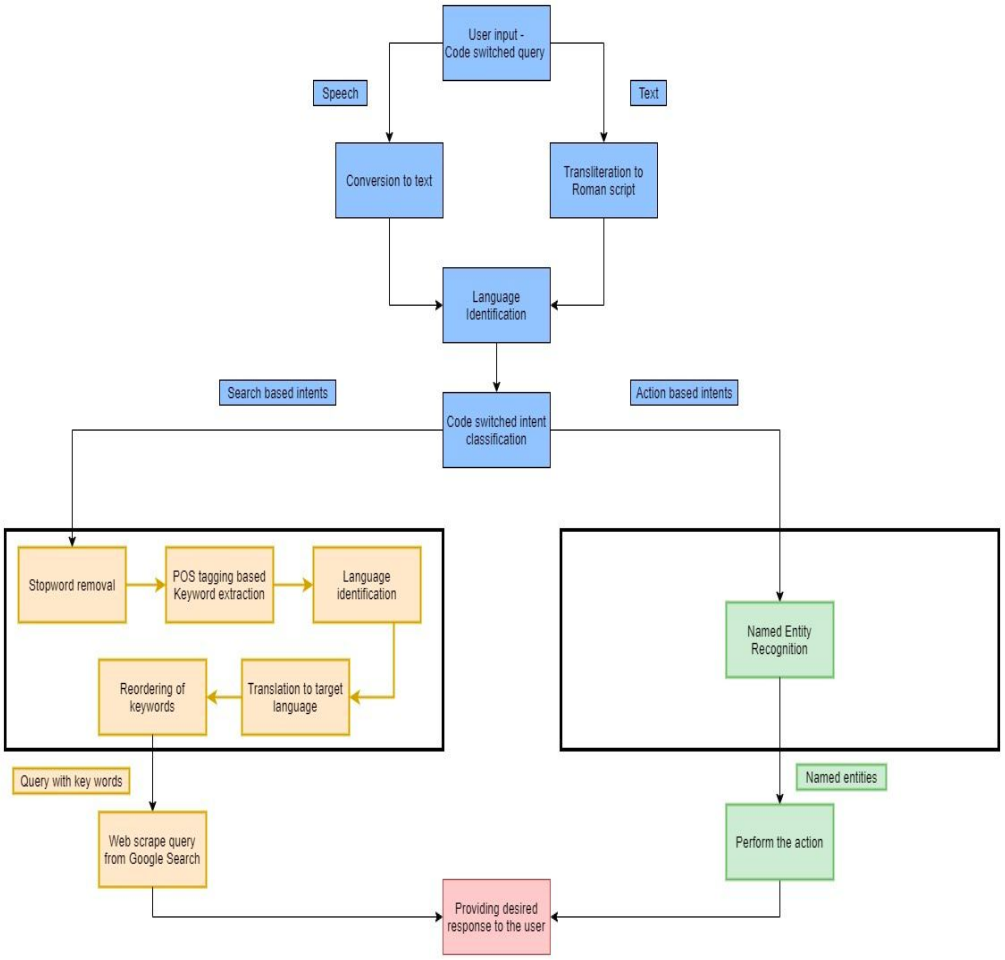

Figure 2 - Project Workflow and Architecture

3.3 Tasks

3.3.1 Intent classification

An user can interact with a virtual assistant to perform a variety of different tasks, like querying for information, querying for local details, playing media, planning the day and many more. The act of correctly identifying what it is an user wishes to accomplish with the help of a virtual assistant is defined as intent classification, with the action that the user wants the virtual assistant to perform being the intent.[1]

It is essential to identify the type of the intent, so that the required action can be performed by the virtual assistant. A study was made on code-switched intent classification by using various supervised classification models with various vectorizing techniques to identify the efficient classification model.

The different vectorisers used were the following:

- **Count vectorizer** - The most straightforward vectorization method counts the number of times a token shows up in the document and uses this value as its weight. Since only the occurrence of the token matters, the language of the word and semantic meanings does not hold weightage in this technique.

- **TF-IDF** - TF-IDF stands for "term frequency-inverse document frequency", meaning the weight assigned to each token not only depends on its frequency in a document but also how recurrent that term is in the entire corpora. Again, neither the language of the word nor semantic meanings hold any weightage.

- **Word2Vec** - Word2Vec is based on a distributional hypothesis where the context for each word is in its nearby words. Hence, by looking at its neighbouring words, it can attempt to predict the target word. The Skip-gram architecture to generate Word2Vec was implemented where Skip-gram learns to predict the context words from a given word, in case where two words (one appearing infrequently and the other more frequently) are placed side-by-side, both will have the same treatment when it comes to minimising loss since each word will be treated as both

the target word and context word.

The Skip-gram architecture includes the following:

- **Data Preparation** - Define the corpus, clean, normalise and tokenize words
- **Hyperparameters** - Set earning rate, epochs, window size, embedding size
- **Generate Training Data** - Build vocabulary, one-hot encoding for words, build dictionaries that map an id to every word and vice versa
- **Model Training** - Pass encoded words through forward pass, calculate error rate, adjust weights using backpropagation and compute loss
- **Inference** - Get word vector and find similar words

The various supervised classifier models implemented include Naive Bayes classification model, K-nearest neighbour classification model, Random forest classifier, Linear Support Vector Classifier model, Logistic classifier model and Decision tree classifier model. A statistical approach was preferred according to the availability of dataset size, since a smaller size of data could lead to overfitting during training neural networks like RNN, the results of BiLSTM was also compared with statistical models which provided proof to the assumption. Among various statistical models tested, the one with prominent results was given by the 'Linear Support Vector classifier'.

3.3.1.1 Linear Support Vector Classifier

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples into various classes. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.[2,5]

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

The support vector machine searches for the closest points as shown in figure 3, which are considered as the 'support vectors'.

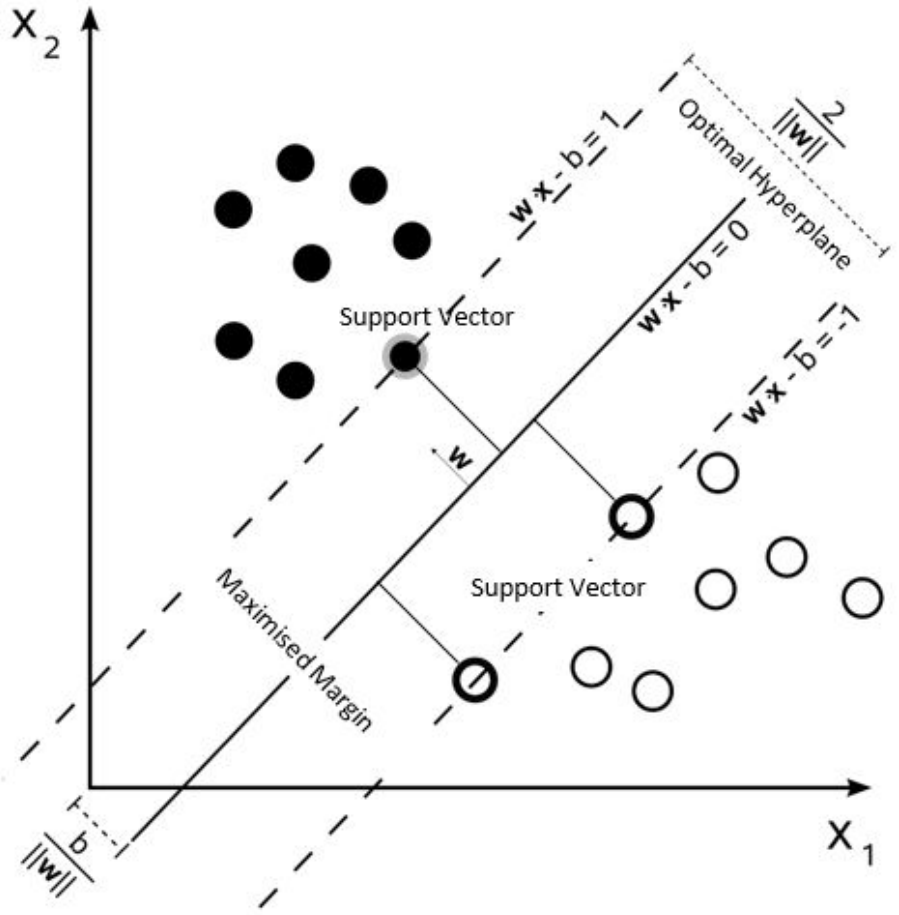

Figure 3: Support Vector Classifier - Hyperplane and Support Vectors

According to the SVM algorithm, it finds the points closest to the line from both the classes. These points are called support vectors. Now, it computes the distance between the line and the support vectors. This distance is called the

margin. The goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

3.3.2 Stopword removal

Stopwords are words in any language's vocabulary that are commonly used, and provide little to no value to the meaning of a sentence.

The removal of stopwords is preferred as it allows models to be trained on words that are truly contributive to the meaning of the sentence, and also reduce dimensional space of the feature vectors generated from the CountVectorizer.

A stopword list is generated by sorting the frequency of words occurring in a given corpus in descending order, and deciding upon a cutoff below which words are not considered to be frequent.

Different applications may or may not require the use of a stop list, and this dependency stems from a chance that a word or words in the list may provide some meaning to the target sentence or corpora.

This project implements a stop word list consisting of hindi, kannada, and english words all transliterated to the Roman script.

3.3.3 Parts of Speech tagging based keyword extraction

Parts of Speech (POS) tagging is the process of assigning a word class, such as a noun,verb, pronoun, etc to a target word in a given sentence.

These word classes, also called POS tags, give insight into the expected neighbors of a target word, and also relationships between words in the same sentence. They also act as features for a task called Named entity extraction, which will be extrapolated upon in the coming sections.

There are 3 ways to train a POS-tagger:

1. Using a HMM
2. Using a Maximum Entropy Markov Model
3. Using a CRF

The task in this project for Code Mixed POS tagging, the paper by Singh et al is considered, who concluded that the CRF model provided the best results for POS tagging of short code mixed Hindi-English social media tweets. The advantage of bidirectionality is present in the CRF while absent in the rest of the proposed models. Thus the same is implemented for POS tagging code mixed queries.

In the task of intent classification, nouns, verbs, adjectives, and pronouns contribute the maximum meaning to a given sentence. Thus words are extracted with these specific POS tags, and discard the rest.

3.3.4 Language identification:

Language identification at the word level involves the tagging of each word in the input code mixed sentence with the language it belongs to.

This is important as it helps to identify if the code mixed query is code mixed in the first place, and also acts as a feature in assigning POS tags to each of the words in the input query. During the reordering of keywords obtained, it is required that every word in the code mixed query be in the same language. Thus, it is imperative that non english words be translated to english, where in language identification identifies each word as an english, hindi or kannada word.

This subtask also gives insight into the languages that are involved in the code mixed query, usually a combination english and hindi, but can also be a combination of english and kannada.

Bhat et[] al as a part of FIRE 2014 developed a language identification tool called lit_cm, that could identify one of 7 different languages a word could belong to. The system made use of letter based language models to classify a series of letters into one of 7 different labels, effectively becoming a sequence classification task.

Google's language detect API, while patented has proved to have shown great accuracy in detecting the language of each word, and also has a well maintained open source community.

After comparing the pros and cons of the above 2 solutions, Google’s language detect API was decided upon.

3.3.5 Translating keywords to target language:

For external search engines like Google, Bing, Microsoft etc, it is observed that monolingual queries, english in particular, tend to give better search results, in terms of relevance to the query, as compared to code switched queries.

Along the same lines, there exist various monolingual task oriented dialogue systems that understand monolingual queries much more accurately than code switched ones.

Word level translation was performed using the Google Translate API that has support for almost all common vernacular languages. It was decided against training an in-house translator due to the lack of gold standard parallel data.

The target language was chosen as english, due to its widespread understanding and data abundancy for tackling further downstream tasks if needed, like sentence level machine translation, next word prediction etc.

3.3.6 Reordering the keywords:

The extracted Monolingual keywords, while capturing a majority of the information conveyed by the original code mixed query, does not retain order and consequently the relationships between each of the keywords.

In the chapter “Constituency Grammars” of “Speech and Language Processing” by Jurafsky et al[], the author proceeds to state that most virtual assistants, like chatbots for instance, receive queries that are of the imperative structure, or the interrogative structure.

An imperative sentence is a sentence that resembles a request, or an order, or an intent to request something from someone or something.

Eg: Show me last week's homework

An interrogative sentence is a sentence that resembles a question to something or someone.

Eg: What is the weather in Bangalore?

Based on the POS tags obtained for each of the keywords, rearrangement of list of keywords into one of the above sentence structures was constructed. This is done by making use of the sentence level construction rules for each of the above sentence structures, and expanding them till a single expression is obtained for both types of sentences, that retained the maximum information, had a grammatical structure, and was as simple as possible to attain the previous objectives. It then becomes a task of placing tag specific words into tag specific regions.

The expressions so obtained were:

Interrogative Sentences ⇒ pron_wh (pron)(adj)$^{+}$ (noun)$^{+}$

Imperative Sentences ⇒ verb (pron)(adj)* (noun)*

Pron_wh : Pronoun(Who,what,where,when,why,how)

Adj : Adjective

Verb: verb

Noun: noun

Pron: pronoun

+ : Regular Expression Semantic for 1 or more than 1 matches

3.3.7 Web scraping queries

Web scraping is a term used to describe the use of a program or algorithm to extract and process large amounts of data from the web. Web scraping using Python is used to extract the data into a useful form that can be imported.

After performing the above tasks that are translating entities to target language and reordering the keywords,the final query is passed to a web scraping algorithm where it has the ability to scrape the google search results for particular queries.

Web scraping algorithm uses python beautiful soup module and requests library.

- Beautiful soup is used for pulling data out of HTML and XML files. It works with the parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.
- Requests are used to send HTTP/1.1 requests extremely easily.

  Using requests library, its able to send HTTP request for google search engine by using final query and get the search results in a object ,then obtained result

object is passed into beautiful soup object where it converts the HTML or XML parse tree and from the pares tree its able to get the search data and use it for further tasks.

3.3.8 Named entity recognition:

Named Entity Recognition is the task of assigning substrings of a given string into one of different predefined categories, like person names, organizations, locations, time expressions, quantities, monetary values, percentages, etc.

NER systems allow the extraction of application/domain specific keywords and values that can later be used as parameters for API and function calls.

For example,if one wanted to lookup an individual in a database, based on the query provided to the bot, a “person” tag would have to be assigned to a word or series of words in the query.

Then, the word/words that belong to the “person” category are parameters to a function that lookup a given name in the database.

NER systems can be built in one of 3 different ways:

1. **Pattern Based NER’s**: Here, the entities to be extracted almost definitely have some sort of a structure that can be used to isolate certain parts of the query.

   Regular Expressions and suffix/prefix based heuristics are developed to capture these entities.

Eg: Dates, Phone Numbers, Email IDs etc

With more complicated named entities, regexes end up becoming very difficult to develop, as it is almost impossible to predict each and every variation in which a query can be asked by an user.

2. **Dictionary Based NER's**: named entities of the type Person Name, Restaurant Name, Location etc are easily detected with the help of an external lookup, usually to a database or an API connected to a database. This is also referred to as Ontology, or Lexicon search.

   The downfall of this approach lies in the fact that an absence of a named entity in an entity specific lexicon can lead to misclassifications.

3. **Context Based NER's**: ML systems for NER can be developed with a bidirectional sequence classifier such as a CRF or HMM, that can use words around a target word, to determine if it belongs to one of the many named entities.

   These models require the use of annotated data, with entities if present in a query, being correctly labelled.

   Bidirectionality in this case allows the model to learn what comes before and after a given target entity word or string.

Open Source ML based named entity extractors include SNER(Stanford Named Entity Recognizer), Spacy, NLTK etc.

The project heavily makes use of pattern and dictionary based NER systems, with the addition of word/character level heuristics for dealing with code mixed queries.

The NER system developed for this project is one of the first for code mixed queries, capable of identifying over 7 different named entities namely Location, Hotel Name, Number of People, Restaurant Name, Date, Time, Phone Number, Email ID and Activity.

3.3.9 User interface Development:

The user interface used in the project is the Telegram messenger platform.[3,4] Telegram provides a simple API to integrate with python through a library 'python-telegram-bot'. The telegram bot can be created from their service called 'BotFather', which provides a secret key for the bot and also links with an API which is associated with the bot.

The telegram bot interacts with the telegram server and the requests are forwarded to an app server which is hosted on a cloud platform and which decides the response based on the implementation and functionality of the bot. This interaction between telegram server and the deployed app server happens through a 'webhook'.

A webhook (also called a web callback or Reverse API) is a way for an app to provide other applications with real-time information. A webhook delivers data to other applications as it happens, meaning the data is available immediately. The webhook will make an HTTP request to the deployed app (typically a POST), and the data is then shared with the app to interpret it and respond according to implemented logic.

The design and flow of data in the telegram assistant is shown in Figure 4.

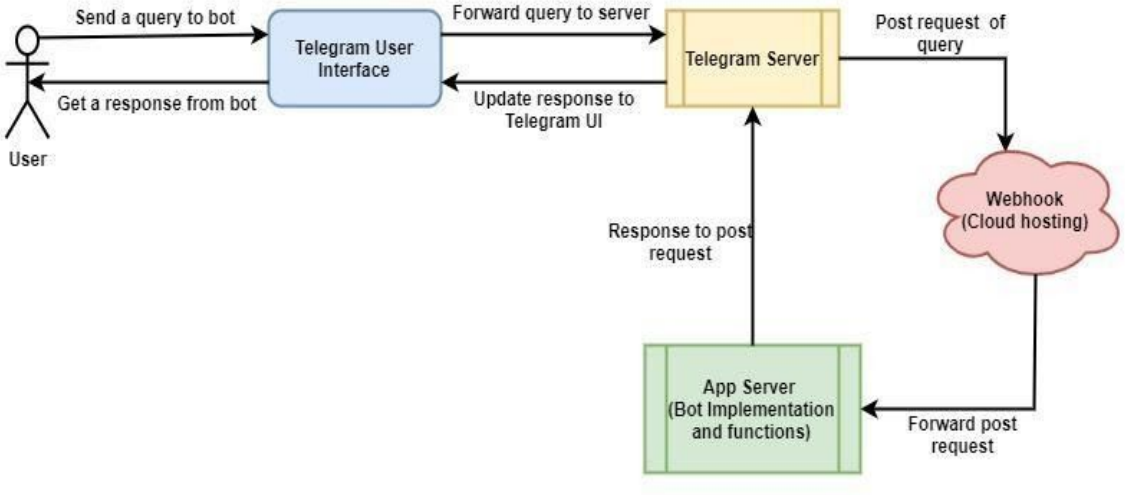

Figure 4: Telegram bot system design

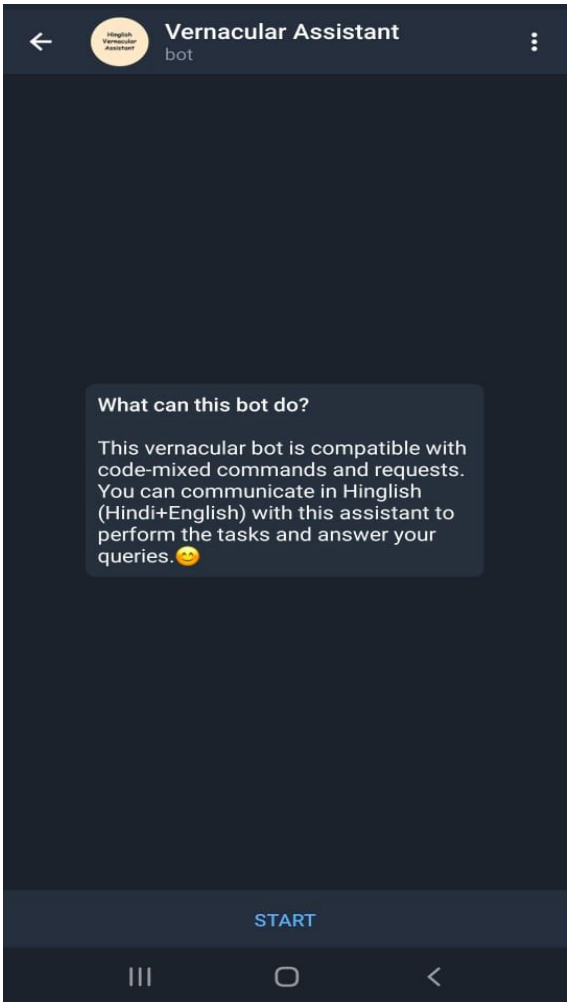

Figure 5: Snapshot of user interface

3.4 Tech Stack

add here

Table 4 : Technology Stack

Libraries, tools and technologies used	Functionality
Python	Scripting, API development (Flask) and Server Management (Backend)
Tensorflow	To use tensorflow models Machine Learning tasks
AWS, Heroku	For web hosting and cloud services
Beautifulsoup, Request	To web scrape desired responses
Joblib	To dump the trained model and outputs
Spyder & Jupyter Notebook	For Python development environment
Telegram Messenger	To provide intelligent assistant / chatbot interface
iNLTK, NLTK, Gensim	To handle text Processing and store/retrieve pre-trained word embeddings
Google SpeechKit	For speech to text conversion
Git & GitHub	For version control

Scikit-learn (sklearn)	For Statistical tools and Classification models
Python-telegram-bot, Telegram API	To create telegram bot application
Google Translation API (googletrans)	To translate Non-English words
IndicTrans	To perform transliteration
StarUML	To create UML visualizations
Angular JS	To create a web interface/dashboard for app

Chapter 4

Modelling and Implementation

In this section, we discuss the various diagrams that have been used to represent our project and interaction workflows.

We also discuss the interaction between the various modules involved at every stage in the technical pipeline.

3 different types of UML diagrams are considered, namely the Use Case diagram, Class diagram, and Sequence Diagram,

4.1 Use Case Diagram

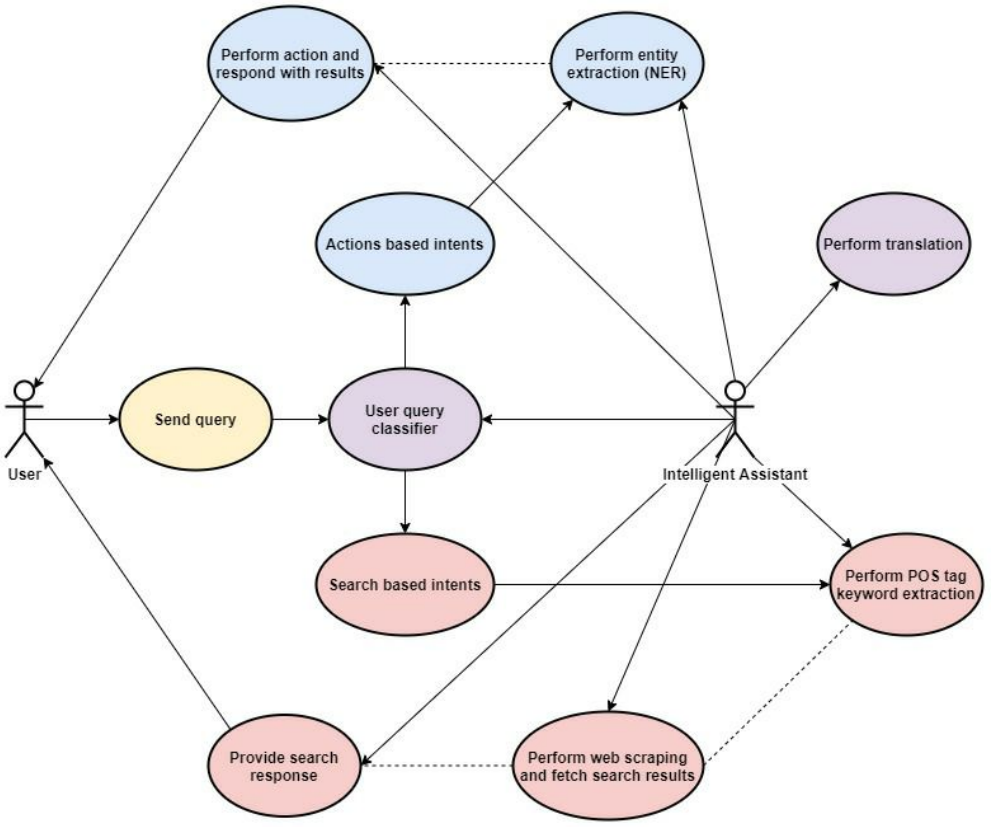

Figure 6: Use-case Diagram of Assistant

The use-case diagram of the project highlights the two key actors, the user and the intelligent assistant. User sends a codemixed query to the assistant which is first classified into a group of intent, 'action based' or 'search based'. The assistant performs a common set of tasks and activities on the user query irrespective of the intent type, which are - intent classification and query translation to common language, these are marked with purple use-case activities in the diagram.

Action based intents undergo entity extraction which is performed using Named-entity Recognition by the assistant and it responds to the user by

performing suitable action and giving the result as response, these are marked with blue use-case activities in the diagram.

Search based intents undergo keyword extraction to identify what is the user looking for and suitable results are scraped from the web and returned as response to the user, these are marked with red use-case activities in the diagram.

4.2 Class Diagram

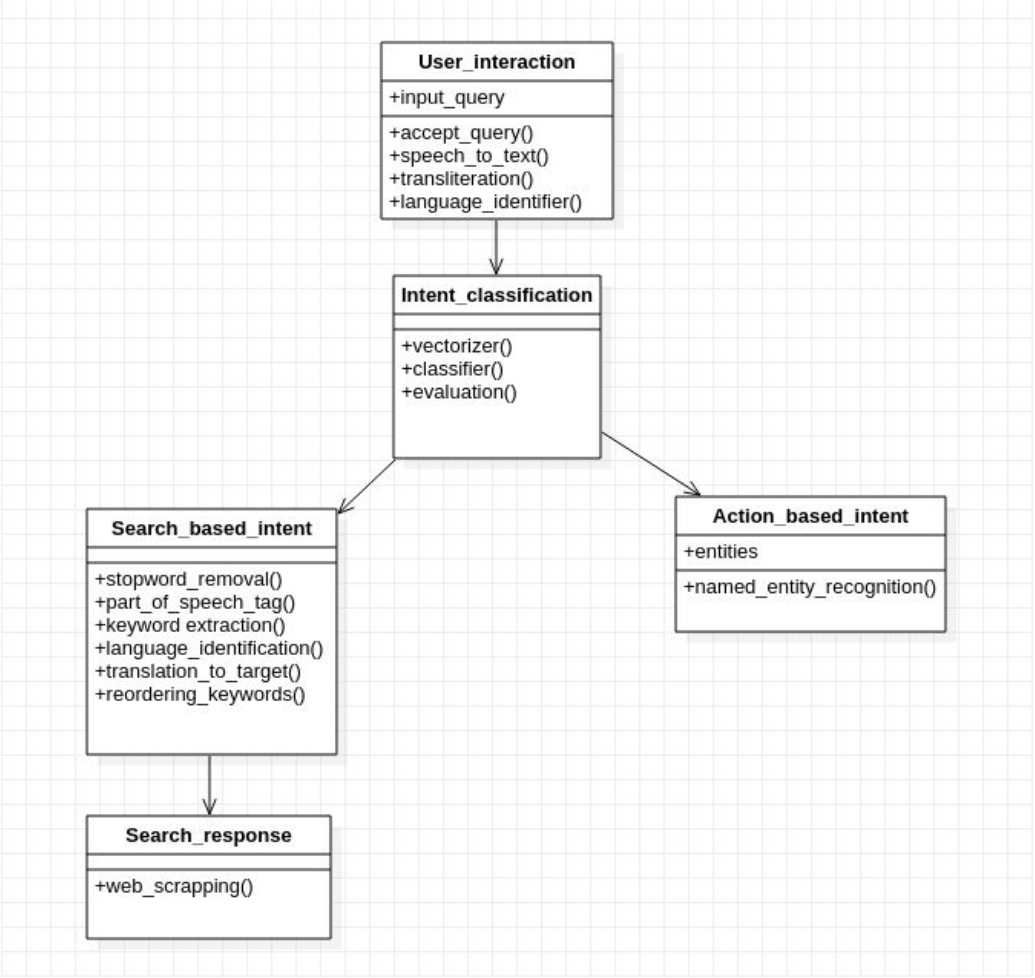

Figure 7 : Class Diagram of Assistant

The class diagram of the project highlights various class identified and their dependencies. User interaction class has attributes representing the intent and operations to accept the input, transliterate queries to Roman script, convert

speech to text and language identification. Intent classification class has the classifier model with operators for vectorizing, modelling and evaluation. Search based intent class has pipeline operations for stopword removal, part of speech tagging, keyword filtering, language identification, translation of keywords to target language and reordering of the keywords while action based intent class has an operation to identify the named entities. Search response class has an operation to scrape the web for desired responses.

4.3 Sequence Diagram

- **Keyword Extraction and Reordering with lookup to External Knowledge base like Google Search and other monolingual applications:**

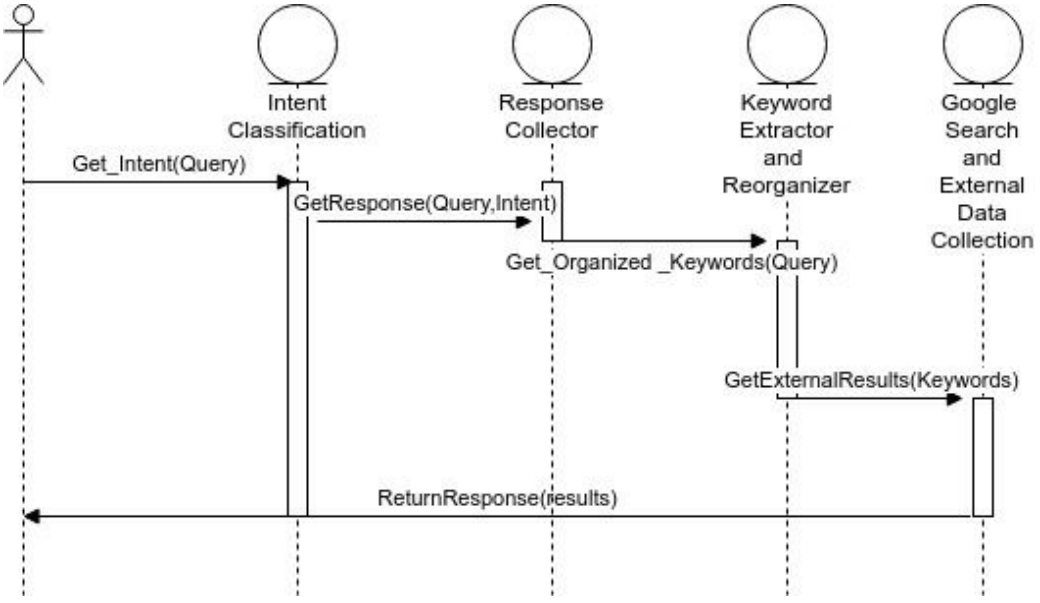

Figure 8: Sequence Diagram for Search based Intents

- **Named Entity Recognition and Extraction Workflow:**

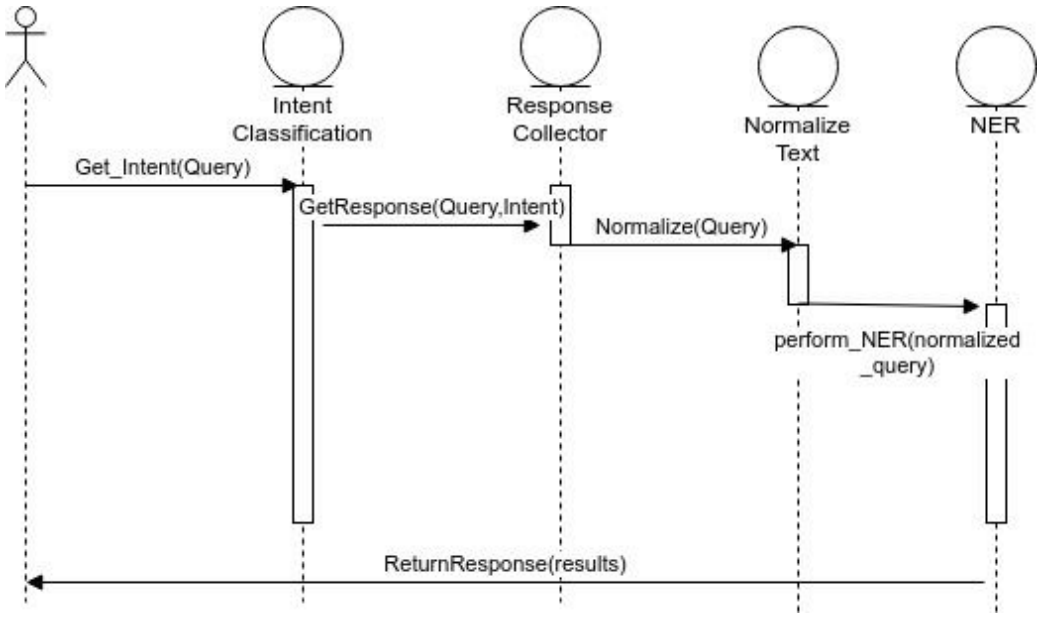

Figure 9: Sequence Diagram for Action based Intents

Chapter 5

Testing, Results and Discussion

para here

5.1 Testing

5.1.1 Word2Vec skip gram model loss per epochs

The overall loss of Word2Vec skip gram model was computed after finishing each training sample according to the loss function. The loss function comprises 2 parts. The first part is the negative of the sum for all the elements in the output layer (before softmax). The second part takes the number of the context words and multiplies the log of sum for all elements (after exponential) in the output layer.

Table 5 : Loss vs epoch of Word2Vec model

Epoch	**Loss**
0	74823.01
100	42615.75
200	40661.27
300	39915.00
400	39466.18
500	39160.74
600	38939.38
700	38776.58
800	38652.04

900	38552.62
1000	38471.69

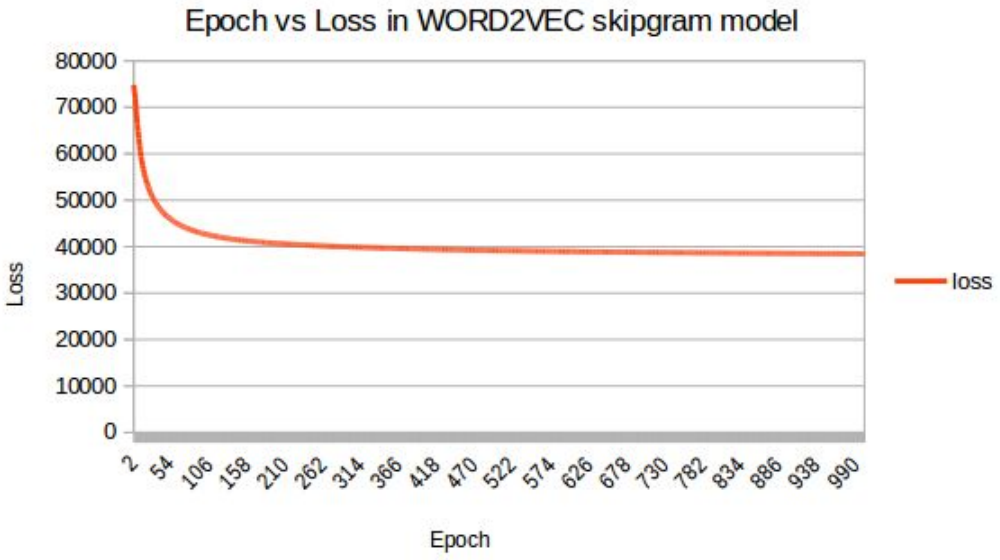

Figure 10: Epoch vs. Loss in Word2Vec skip gram model

Table 6 : Loss vs epoch of Word2Vec model on kanglish dataset

Epoch	**Loss**

100	49865.6749550515
200	47956.6692884845
300	47232.3183586102
400	46798.9909278872
500	46491.8156473194
600	46259.0972022274
700	46071.4506873235
800	45918.1971378264
900	45792.8355868095
1000	45687.9250333492

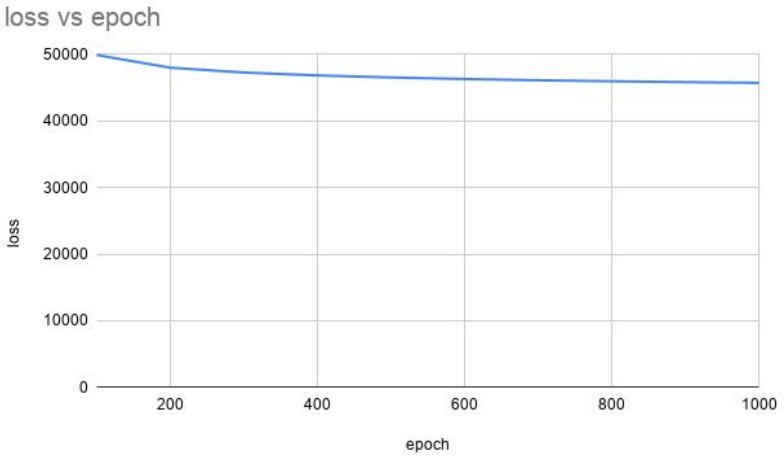

Figure 11: Epoch vs. Loss in Word2Vec skip gram model on kanglish dataset

5.1.2 Eyeball method of testing for Word2Vec skip gram model

One way to evaluate the word2vec model is to develop a "ground truth" set of words. Ground truth will represent words that should ideally be closest together in vector space. For example if the corpus is related to code switched medical queries, the vectors for "treatment" and "ilaaj" should have the smallest euclidean distance or largest cosine similarity. Few test cases using eyeball methods are shown in Table 6.

Table 7 : Test cases for Word2Vec skip gram model validation on Hinglish dataset

Word 1	**Word 2**	**Cosine similarity index**	**Hit**
vehicles	gaadi	0.940	HIT
treatment	ilaaj	0.950	HIT
food	khaana	0.880	HIT
health	svaasthy	0.724	MISS
insurance	beema	0.884	HIT
effects	asar	0.945	HIT
symptoms	lakshan	0.919	HIT
fees	shulk	0.935	HIT
letter	patr	0.948	HIT
nearest	paas	0.761	MISS

Table 8 : Test cases for Word2Vec skip gram model validation on kanglish dataset

Word 1	**Word 2**	**Cosine similarity index**	**Hit**
food	oota	0.9505	HIT
letter	patra	0.972133	HIT
lakshana	symptoms	0.8911551538108418	HIT
prabhaava	effect	0.97456	HIT
parinama	effects	0.9192778005362223	HIT
time	hotu	0.978835199729751	HIT
show	torisu	0.8754499278303974	MISS

5.1.3 Classification Metrics of intent classification

A study was made on code-switched intent classification by using various supervised classification models with various vectorizing techniques to identify the efficient classification model. The study was validated on standard classifier metrics including accuracy, precision, recall, F1-score and support.

5.1.3.1 Accuracy

Accuracy in classification problems is the number of correct predictions made by the model over all kinds of predictions made.

$$Accuracy = (TP + FP) / (TP + FP + FN + PN)$$

Table 7 : Comparison of accuracy of the classifier models

Classifier	**Count vectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	0.91265	0.88253	0.85240
K-nearest Neighbour classifier	0.84036	0.85240	0.65060
C-Support Vector Classifier	0.09036	0.09036	0.09036
Random Forest Classifier	0.89759	0.84939	0.89457

Linear Support Vector **Classifier**	0.95180	0.93975	0.84638
Logistic Regression classifier	0.94879	0.92771	0.44879
Decision Tree	0.90662	0.82831	0.91265

5.1.3.2 Precision

Precision is the fraction of predicted positive events that are actually positive.

$$Precision = (TP) / (TP + FP)$$

Table 8 : Comparison of precision of the classifier models

Classifier	**Count vectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	0.92	0.88	0.87
K-nearest Neighbour classifier	0.88	0.86	0.77
C-Support Vector Classifier	0.01	0.01	0.01
Random Forest Classifier	0.91	0.87	0.91
Linear Support Vector	0.96	0.95	0.87

Classifier			
Logistic Regression classifier	0.95	0.93	0.67
Decision Tree	0.92	0.84	0.93

5.1.3.3 Recall

Recall (also known as sensitivity) is the fraction of positive events that are predicted correctly.

$$Recall = (TP) / (TP + FN)$$

Table 9 : Comparison of recall of the classifier models

Classifier	**Count vectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	0.91	0.88	0.85
K-nearest Neighbour classifier	0.84	0.85	0.65
C-Support Vector Classifier	0.09	0.09	0.09
Random Forest Classifier	0.90	0.85	0.89
Linear Support Vector	0.95	0.94	0.85

Classifier			
Logistic Regression classifier	0.95	0.93	0.45
Decision Tree	0.91	0.83	0.91

5.1.3.4 F1-score

The F1-score is the harmonic mean of recall and precision, with a higher score as a better model.

$$F1 - Score = (2 * Precision * Recall) / (Precision + Recall)$$

Table 10 : Comparison of F1-score of the classifier models

Classifier	**Count vectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	0.91	0.88	0.85
K-nearest Neighbour classifier	0.84	0.85	0.67
C-Support Vector Classifier	0.01	0.01	0.01
Random Forest Classifier	0.90	0.85	0.89
Linear Support Vector Classifier	0.95	0.94	0.85

Logistic Regression classifier	0.95	0.93	0.44
Decision Tree	0.91	0.83	0.91

5.1.3.5 Support

The support is the number of samples of the true response that lie in that class.

Table 11 : Comparison of support of the classifier models

Classifier	**Count vectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	332	332	332
K-nearest Neighbour classifier	332	332	332
C-Support Vector Classifier	30	332	30
Random Forest Classifier	298	332	332
Linear Support Vector Classifier	316	332	332
Logistic Regression	315	332	332

classifier			
Decision Tree	332	332	303

5.1.4 Classification Metrics for NER

- Performance of NER for each action based intent, calculated by counting the total number of true positives and negatives,false positives and negatives for each entity associated with each intent, on a token basis. Precision and Recall are calculated on the number of False/True Positives and Negatives. This is done on each and every intent that makes use of the NER workflow, and is calculated on the basis of only intent specific entities, rather than all of the entities.

Table 12 : NER metrics w.r.t intent on Hinglish dataset

Intent	**Precision**	**Recall**	**F1-Score**
Hotel Booking	96	71.2	83.177
Restaurant Booking	100	60	75
Travel Booking	100	88.88	94.12

Reminder	91	82	86.15

Table 13 : NER metrics w.r.t intent on Kanglish dataset

Intent	**Precision**	**Recall**
HOTEL	89	35
RESTAURANT	96	65.15
TRAVEL_BOOKING	100	60
REMINDER	68	62

- Performance of NER, calculated by counting the total number of False Positives and Negatives and True positives and Negatives, over each and every query that makes use of a given entity, on a token basis.

Table 13 : NER metrics w.r.t entity

Entity	**Precision**	**Recall**	**F1-Score**
Date	100	87	93
Time	100	84	91.52

Hotel Name	96	55	70
Location	100	88.88	94.12
Restaurant Name	100	65.1	78
Activity	91	87.5	89.21
Number of people	100	100	100

5.2 Results

Samples of test case queries in code-switched made by the user and the assistant's responses are shared.

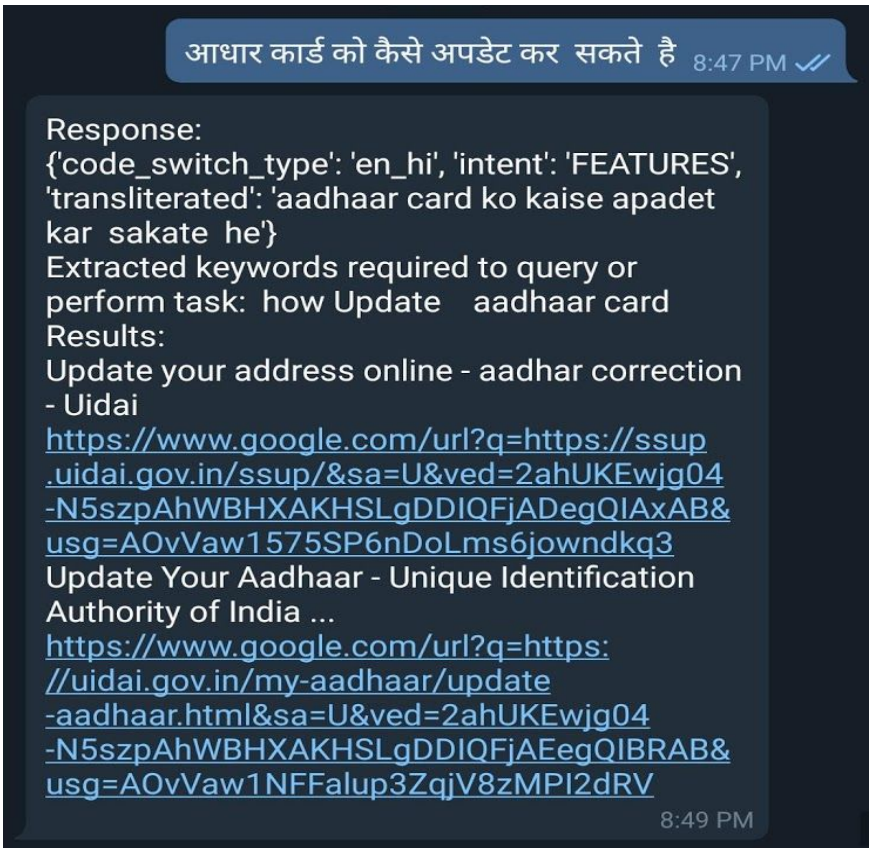

Figure 11: Test case of code-switched query 1

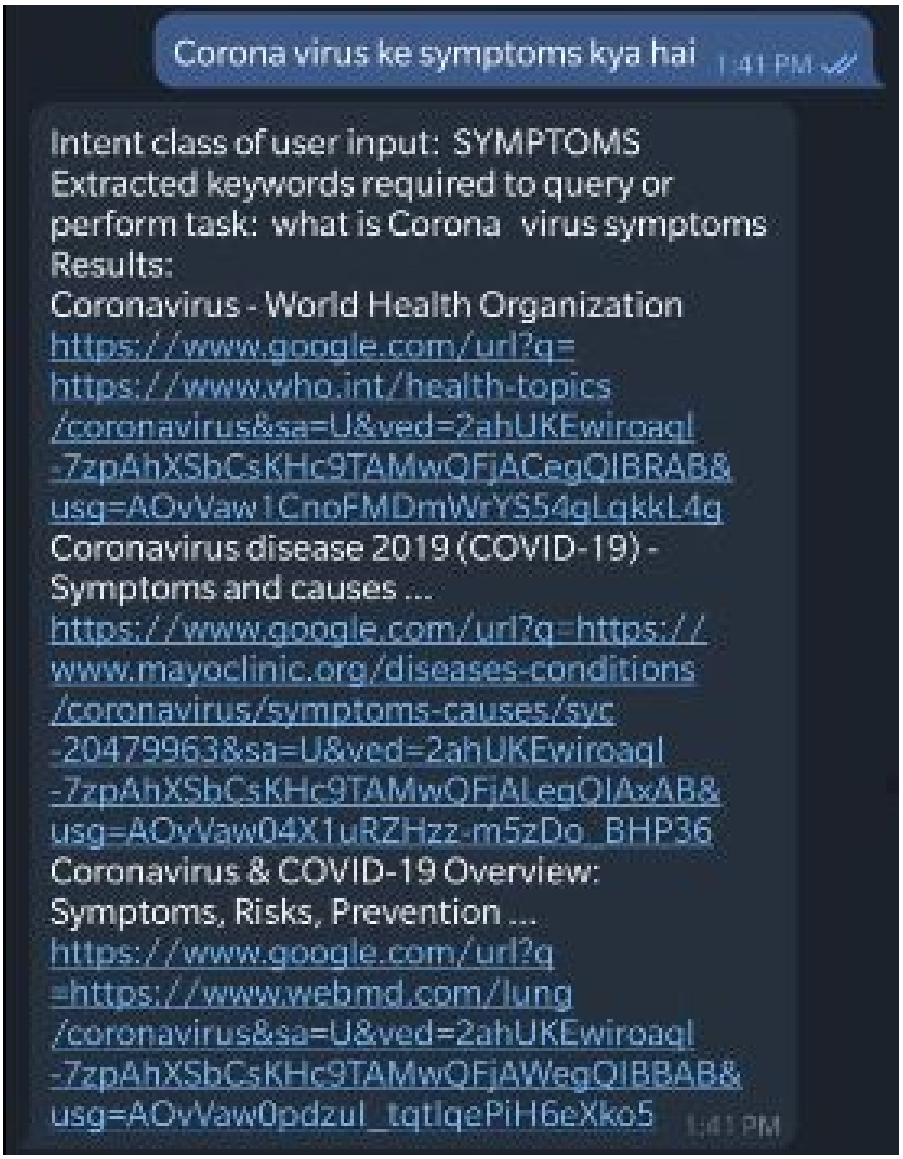

Figure 12: Test case of code-switched query 2

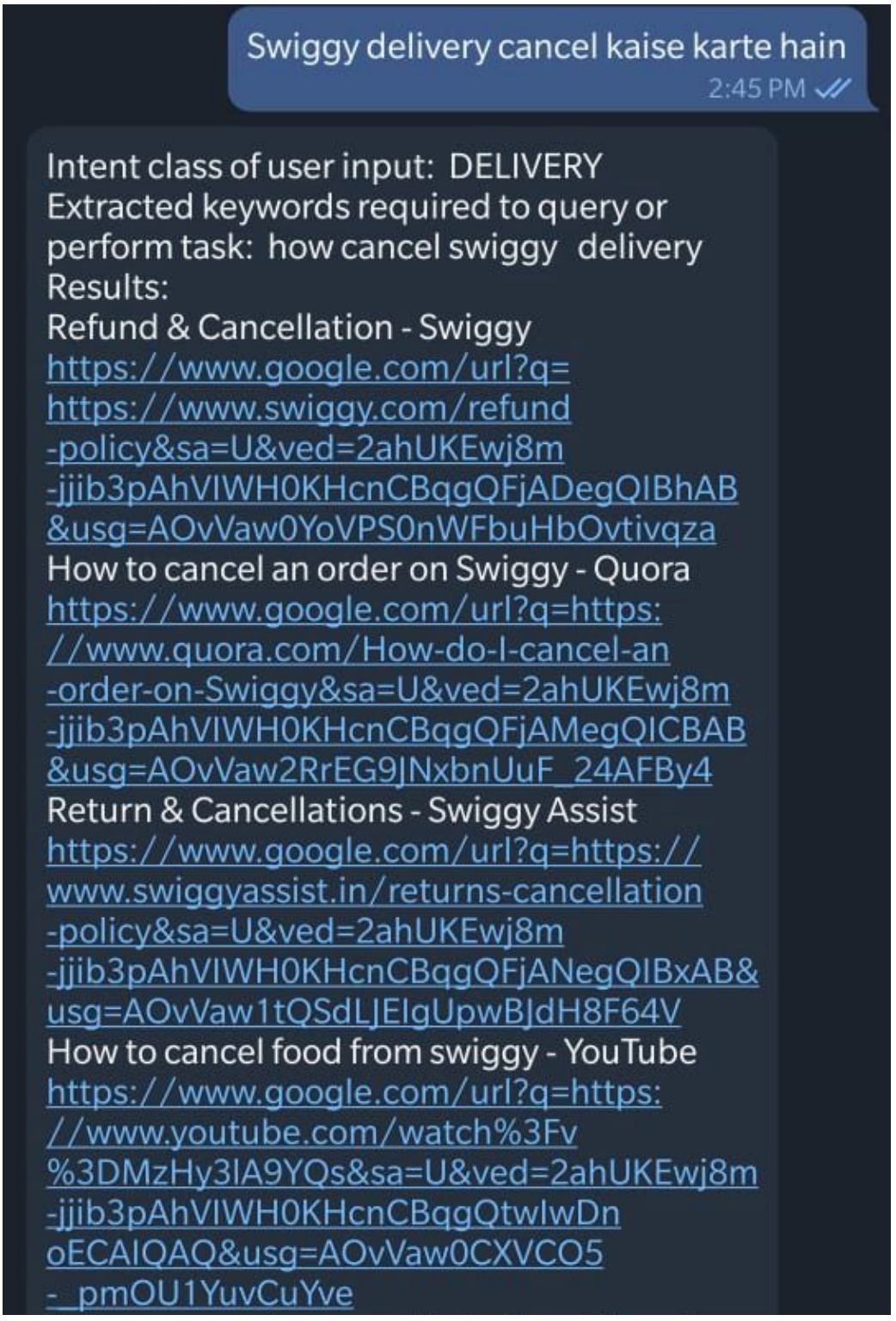

Figure 13: Test case of code-switched query 3

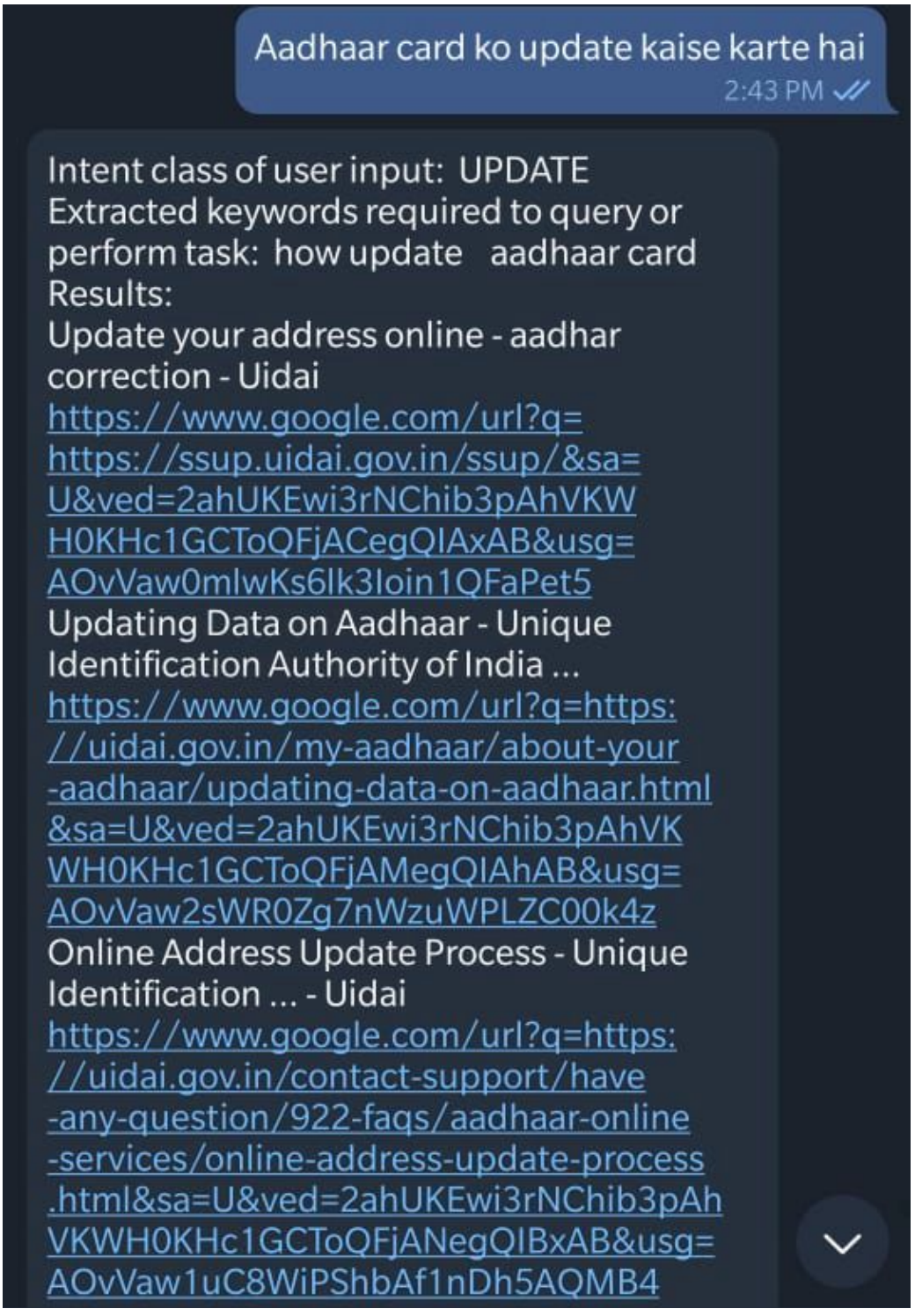

Figure 14 : Test case of code-switched query 4

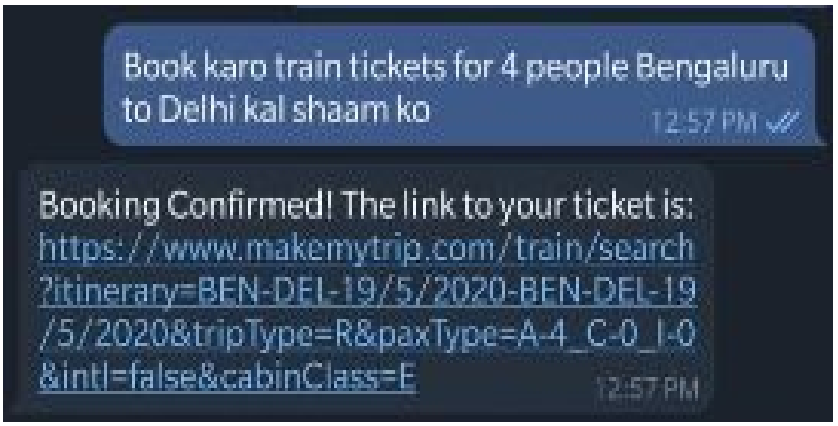

Figure 15: Test case of code-switched query 5

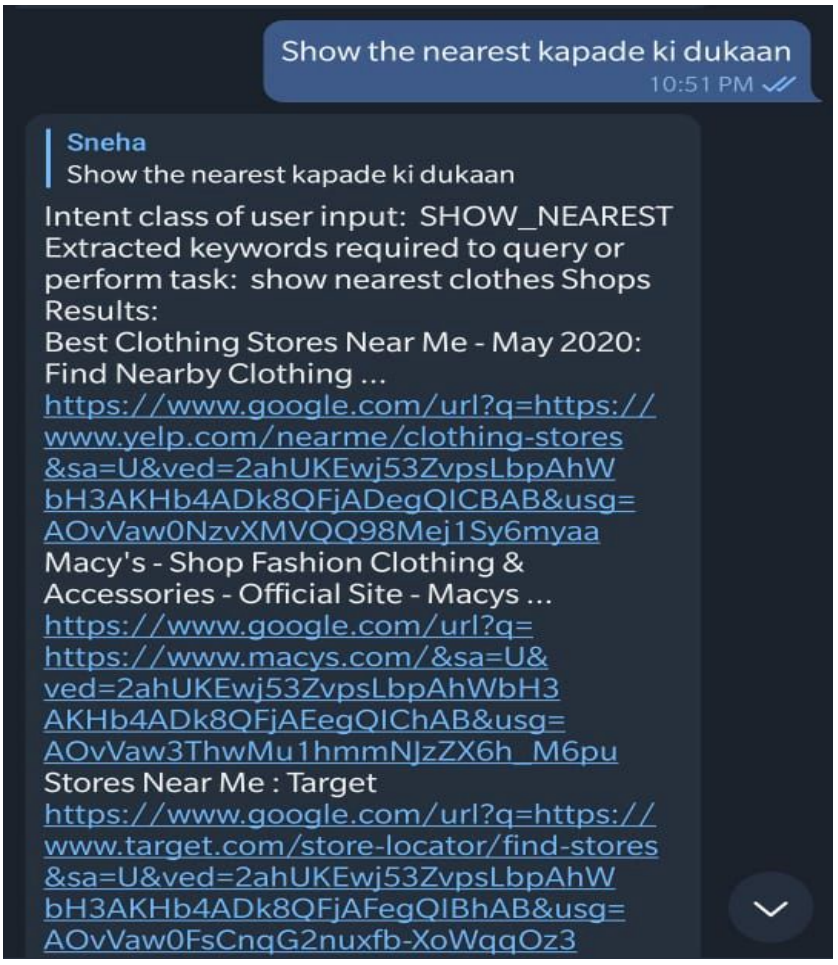

Figure 16 : Test case of code-switched query 6

5.3 Discussion

- Currently the corpus generated has a high multilingual index for all the classes except booking and insurance classes while language entropy is high for all classes other than insurance class.
- The study on intent classification shows that Linear support vector classification performs better than all the other classifiers when used along with either of count vectors or TF_IDF vectors with accuracy of 95% and 93% respectively. The precision score and recall score when Linear support vector classifier was used with count vector and TF_IDF was 96% and 95%.
- The responses provided by the assistant to the code-switched queries were observed to be relevant and pertinent.

Chapter 6 - Conclusion and Future Tasks

A robust intelligent assistant, which could respond to Hindi-English and Kannada-English code-switched input queries from the user, was developed from scratch. The pipeline involved working with various natural language processing tasks on code-switched data including speech to text transcription, intent classification, parts of speech tagging, named entity recognition, keyword extraction and keyword structuring. A code-switched corpus was also generated with code-switched queries of frequently asked questions which had high strength when tested on standard code-switching metrics.

There are, however, stages in the project pipeline that can further be improved.

The use of a speech to text SDK for Automatic Speech Recognition tasks while convenient, does not incorporate strategies to enhance code mixed speech transcription.

With inspiration from the pipeline proposed by Dau-Cheng et al[] , a speech to text pipeline is currently being developed to accommodate for English-Hindi and English-Kannada code mixed speech input. Due to monolingual speech to text models being available for all 3 of the above languages, made open source by Povey et al[] , a custom speech to text architecture that can make use of these monolingual speech to text models at a word level will allow us to build robust English-Hindi and English-Kannada speech recognition systems. Word level language classification models built on Speech can be trained using crowd sourced data from OpenSLR[] and IndicTTS[].

Pretrained Multilingual Supervised Word Embeddings(MUSE) published by Facebook et al [] can represent words of two or more than two different languages in the same embedding space. That is, words that have some sort of connection or similarity in different languages are present near one and other.

Along the same lines, mBERT(Multilingual Bidirectional Encoder Representations from Transformers) published by Google et al[] also possess the same capabilities, but have pretrained models that are used extensively in both industry and research. Making use of these pretrained models and fine

tuning with our dataset can boost performance and at the same time extend to other regional languages like tamil and telugu.

Bibliography

1. Xiao Li, Ye-Yi Wang, Dou Shen, and Alex Acero. 2010. Learning with click graph for query intent classification. ACM Trans. Inf. Syst. 28, 3, Article 12 (June 2010), 20 pages. DOI:https://doi.org/10.1145/1777432.1777435

2. Mendoza M., Zamora J. (2009) Identifying the Intent of a User Query Using Support Vector Machines. In: Karlgren J., Tarhio J., Hyyrö H. (eds) String Processing and Information Retrieval. SPIRE 2009. Lecture Notes in Computer Science, vol 5721. Springer, Berlin, Heidelberg

3. Roy, K. Sripath, et al. "Realization of a low cost smart home system using Telegram messenger and voice." International Journal of Pure and Applied Mathematics 116.5 (2017): 85-90.

4. Core.telegram.org. 2020. Telegram Bot API. [online] Available at: https://core.telegram.org/bots/api

5. Carrasco, Oscar Contreras. “Support Vector Machines for Classification.” Medium, Towards Data Science, 22 Aug. 2019, towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3.

6. Chia, Derek. “A Line-by-Line Implementation Guide to Word2Vec Using Numpy.” Medium, Towards Data Science, 13 Sept. 2019, towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281.

7. Long, Andrew. “Understanding Data Science Classification Metrics in Scikit-Learn in Python.” Medium, Towards Data Science, 9 Feb. 2019,

towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019.

8. “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition”, Dan Jurafsky and James.H.Martin, Prentice Hall Publishing, 2000