Ramaiah Institute of Technology

Department of Information Science and Engineering

Vernacular Code Switching in Intelligent Assistants

Team Members

Mohammed Sahil MH (1MS16IS040)
Shrinidhi KR (1MS16IS094)
Sneha V (1MS16IS098)
Adithya Rao (1MS16IS131)

Guided By

Ms. Rajeshwari S B. ***M.Tech, (Ph.D.)***
Assistant Professor
Dept of Information Science & Engineering
Ramaiah Institute of Technology(MSRIT)

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. Multilingual users tend to replace certain subsets of a language's vocabulary with another language when conversing, end up being misinterpreted and hence fail to elicit an appropriate response from the assistant. Yet, code switching is not inherently taken care of by today’s intelligent assistants. This project, therefore, specifically deals with vernacular code switching in Hindi-English and Kannada-English switches.

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 and Kanglish frequently asked questions (FAQs) and Common action dependent queries from a variety of different domains.

Scope

Home Assistants and Chatbots:

- Interact with a larger user base consisting of people who interact primarily in code mixed speech.
- Make a more robust assistant that can handle a larger variety of user queries.

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:

This project can setup experiments for a similar tasks with a slight variation in the languages dealt with, namely kannada-english code switching, and hindi-english code switching.

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

Introduction to Code-Switching

Code Switching - In linguistics, code-switching occurs when a speaker alternates between two or more languages, or language varieties, in the context of a single conversation.

Examples

1. Conveyed - nanu barak munchene, **he left**

   To be conveyed - He left before I came.

2. Conveyed - nanu **vehicle park** madi manage hoguttini

   To be conveyed - I will park the vehicle and then go home.

3. Conveyed - Report submit **kardena** before going home

   To be conveyed - submit the report before going home

Challenges and Issues

1. There are no fixed regions where code-switching tend to occur. Therefore identifying the code-switched regions is critical.

2. Dataset of text/speech segments that posses code switched FAQs is not available.

3. Building the intents dataset is laborious.

4. Utterance and pronunciation of Non-English words in code-switched text needs to be considered.

5. Misspelt words are tough to handle.

Corpus generation

Code-Switched Hinglish Dataset

Table 1 : Details of Hinglish corpus collection

	A	B	C
1	**Code switched query**	**High level class**	**Lower level class**
2	Mujhe kaise pata chalega swiggy order is confirmed	SWIGGY	ORDER
3	kya hum apana order on swiggy cancel kar sakta hai	SWIGGY	ORDER
4	updated aadhaar card kahaan se praapt hoga	AADHAAR	UPDATE
5	Mujhe dengue ka symptoms kaise pata chalega	MEDICAL	SYMPTOMS
6	Show me the directions to the nearest dhoodwala	GENERIC	SHOW_NEAREST
7	Remind me kal doctor ka appointment hai	GENERIC	REMINDER
8	kaunse tax benefits milate hai if i take health insurance	INSURANCE	HEALTH

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

Code-Switched Kanglish Dataset

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

	A	B	C
1	**Code switched query**	**High level class**	**Lower level class**
2	Nanna swiggy order yavaga barutte	SWIGGY	ORDER
3	social distances hege coronavirus stop maduttade	MEDICAL	PREVENTION
4	Book a cab to KR Market 10 gante ge	BOOKING	TRAVEL_BOOKING
5	Remind nale car servicing center hogabeku	GENERIC	REMINDER
6	Alarm set madu aru gante ge hospital appointment ide	GENERIC	REMINDER
7	aadhaar card ali fingerprints update madboda	AADHAAR	UPDATE
8	Vomitting, fever, headache yava disease ge symptoms	MEDICAL	SYMPTOMS

Standard code-switching metrics

- **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.

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

  k = number of languages involved

  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

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

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

Code-Switched Hinglish and Kanglish Dataset

Table 3 : Corpus statistics on code-switching metrics

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**

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**

Literature Survey

- **Khanuja, Simran, et al. "GLUECoS: An Evaluation Benchmark for Code-Switched NLP." arXiv preprint arXiv:2004.12376 (2020).**
  Presents an evaluation benchmark, GLUECoS, for code-switched languages that spans several NLP tasks in English-Hindi.
  Adaption - 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) and Language Entropy (LE)**

- **Rong, Xin. "Word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).**
  Provides detailed derivations and explanations of the parameter update equations of the word2vec models, including the original continuous bag-of-word (CBOW) and skip-gram (SG) models.
  Adaption - The Skip-gram architecture to generate Word2Vec was implemented to the generated Hinglish and Kanglish corpus.

- **Bhat, I. A., Mujadia, V., Tammewar, A., Bhat, R. A., & Shrivastava, M. (2014, December). IIIT-H system submission for FIRE2014 shared task on transliterated search. In Proceedings of the Forum for Information Retrieval Evaluation (pp. 48-53).**
  Gave insight as to how language identification models can be developed at a word level making use of bigram based character level language models. Mentioned the use of an open source transliteration engine called “Indictrans”.Also made their models and architecture reproducible and open-source, leading to shorter durations in development time and cross checking results.
  Limitation - Application of Transliteration and Language Identification in the use case of intent classification has not been explored.

- **Jayarao, P., & Srivastava, A. (2018, December). Intent Detection for code-mix utterances in task oriented dialogue systems. In 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT) (pp. 583-587). IEEE.**
  Performed Experiments to identify which combination of Vectorizer and Model give the best results for code mixed hindi english utterances, and compared the same to the results from a dataset comprising of both monolingual and code mixed utterances.
  Limitation - Does not expand on identifying vectorizer and model combinations for Kannada english utterances. Also does not provide an user crafted dataset for application specification intent classification.

Literature Survey

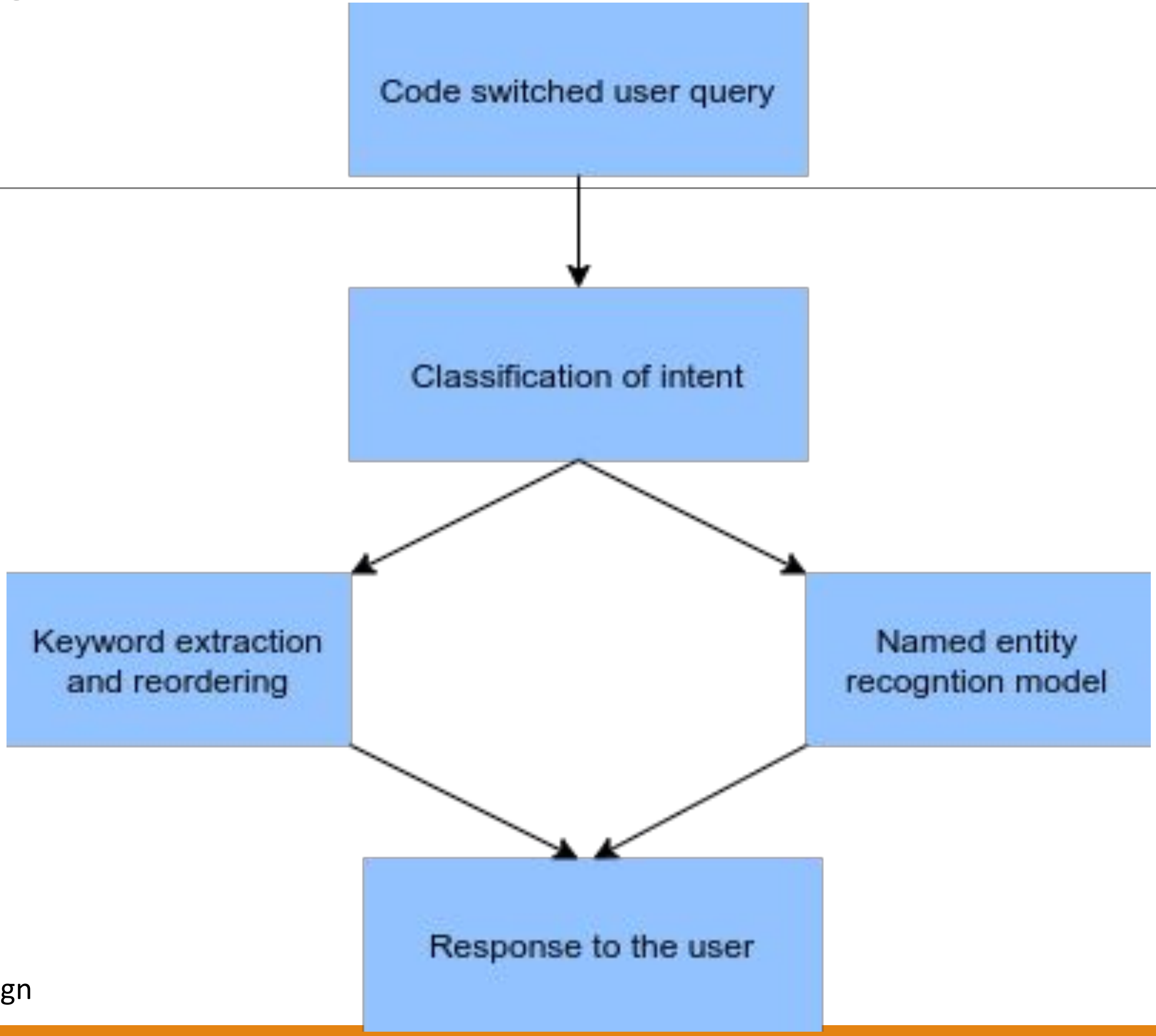
- **Singh, Kushagra, Indira Sen, and Ponnurangam Kumaraguru. "A twitter corpus for Hindi-English code mixed POS tagging." Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media. 2018.**
  Published an annotated dataset for Code Mixed Hindi-English POS tagging, by making use of tweets from a variety of sources. Compared and contrasted among RNN LSTM and Conditional Random Fields for Sequence Labelling.
  Limitation - Does not talk about specific use cases in which code mixed POS taggers could be taken advantage of. Also does not expand upon building POS taggers for other regional languages, like Kannada.

- **Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition."**
  Adaption - This considered as a base for understanding various NLP tasks and it provided an insight into basics of POS tagging, Context free grammar, and sentence construction rules for imperative and interrogative sentences.

- **Bhargava, Rupal, Bapiraju Vamsi, and Yashvardhan Sharma. "Named entity recognition for code mixing in indian languages using hybrid approach." Facilities 23.10 (2016).**
  It identifies the challenge that is faced in recognizing named entities in Indian Social Media Text which is Code Mixed. It describes the proposed approach for shared task CMEE-IL (Code Mix Entity Extraction in Indian Language), FIRE 2016. Proposed algorithm uses a hybrid approach of a dictionary cum supervised classification approach for identifying entities in Code Mix Text of Indian Languages such as Hindi- English and Tamil-English.
  Limitation - Does not make use of POS Tagging and Chunk Parsing to build feature vectors.

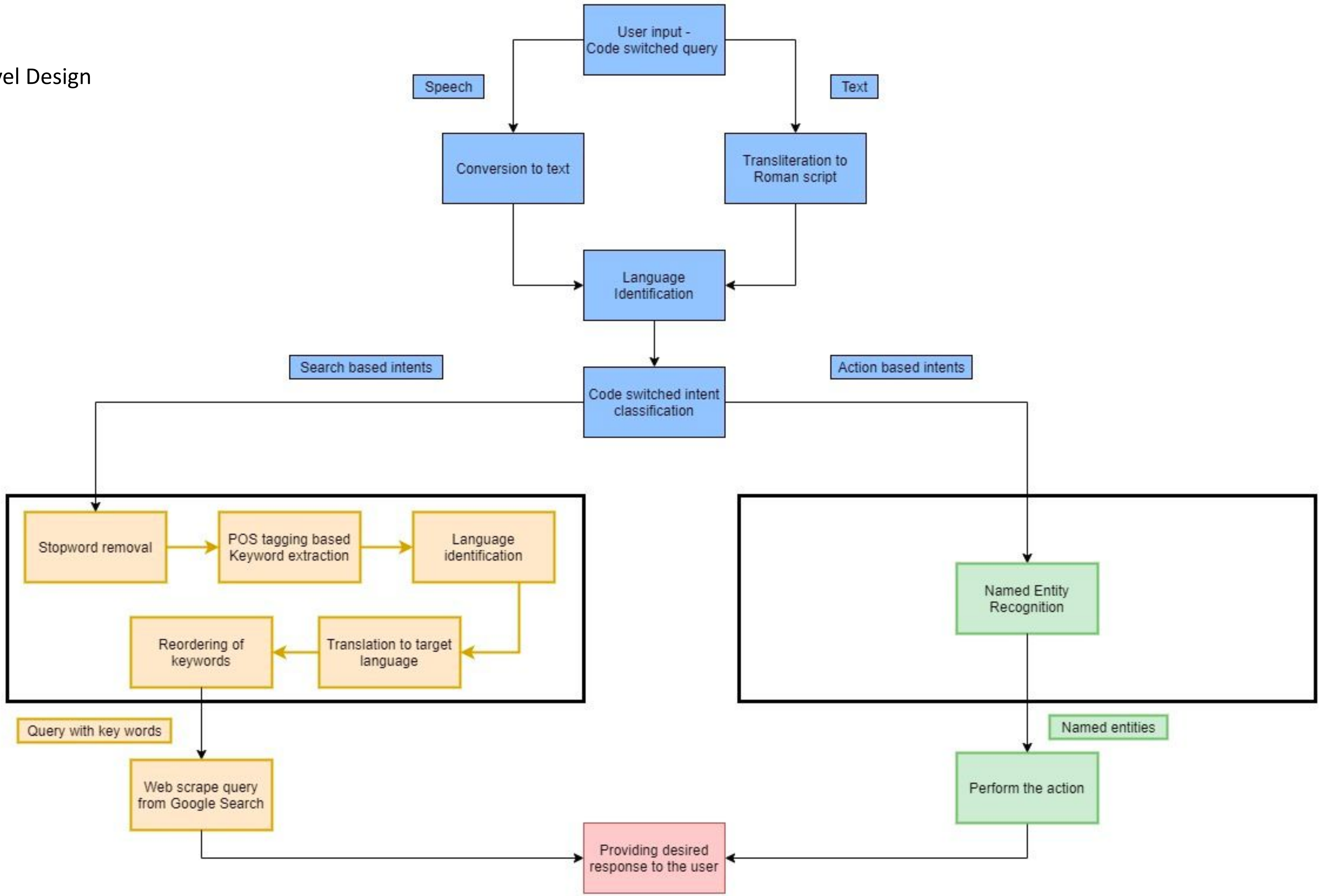
Design Architecture

Proposed Model Overview



High Level Design

Low Level Design



Pipeline explained

Intent classification-

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.

Vectorizers -

- **Countvectorizer** - 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.

Skip-gram algorithm for Word2Vec

The Skip-gram architecture includes the following:

- **Data Preparation** - Define the corpus, clean, normalise and tokenize words

  Input - “Mujhe dengue ka symptoms kaise pata chalega”

  Processed - ["dengue","symptoms","pata","chalega"]

- **Hyperparameters** -

  Learning rate - 0.01

  Epochs - 1000

  Window size - 3

  Embedding size(Dimension of vector) - 5

Skip-gram algorithm for Word2Vec

The Skip-gram architecture includes the following:

- **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

  Example - “dengue” = [0.2056069 0.62899894 0.6566051 1.96063547 0.56355276]

  "symptoms" = [0.2056069 0.62899894 0.6566051 1.96063547 0.56355276]

  sim(symptoms) => lakshan = 0.9727, lakshana = 0.9664

  sim(dengue) => corona = 0.9664

Intent Classification

Classifiers applied
Naive Bayes Classifier
K-nearest Neighbour classifier
Random Forest Classifier
Linear Support Vector Classifier
Logistic Regression classifier
Decision Tree

Why not neural networks or deep learning approach?

Observation- Deep learning architectures require enormous amount of data to perform better without overfitting. Since in our scenario we generated the dataset which is small, the neural network models showed lesser accuracy on test examples.

While the probabilistic and statistical approach perform better at small scale data for test examples as well. Among these Support Vector classifier which considers even the non-linearity and hyperplane decision boundary gave the best results.

Language Identification, Translation, and Transliteration

The above tasks and the approaches used to solve the same have been described as below:

Task	**Solutions Considered**
Language Identification: Given a sentence, tag each word of the sentence with the language it belongs to.	1. CRF Based sequence classification 2. Language Specific Lexicon’s as a Trie 3. Character N-Gram Based Language Modelling(LITCM) **Chosen: Character N-Gram Based Language Modelling**
Language Translation: Given a sentence or a word in one language, translate it to another language.	**Chosen:Google Translate API(Google Trans)**
Language Transliteration: If a sentence is in Devanagiri Script, convert it to Roman Script.	**Chosen: Indic Transliteration tool (LITCM)**

Preprocessing Pipeline

The preprocessing pipeline has 3 stages:

Input - show me nearest kapade ki dukaan jo achchha hai

1. **Stop Word Removal**

   Method - Get rid of words that don't contribute to enhancing meaning of the query

   Output - me, ki, jo, hai

2. **POS Tagging** -

   Method - For Hindi-English and Kannada-English Code Switched 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 switched Hindi-English social media tweets

   Output - show - verb, nearest - adjective, kapade - noun, dukaan - noun, achchha- adjective

3. **POS filtering**

   Method - Get rid of words that aren't Nouns, Adjectives, Verbs, or Pronouns

   Output - As all of the words in the previous phase are belonging to one of the above POS tags, the response remains the same.

Reordering the keywords

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

Eg: Show me kal diya hua homework

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

Eg: ivatu weather Bangalore ali hegide?

The expressions so obtained were:

Interrogative Sentences $\Rightarrow$ (What,When,Why,Where,Who,How) (pron)(adj)$^{+}$ (noun)$^{+}$

Imperative Sentences $\Rightarrow$ verb (pron)(adj)* (noun)*

Adj : Adjective

Verb: verb

Noun: noun

Pron: pronoun

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

* : Regular Expression Semantic for 0 or more matches

Web scraping 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 beautifulsoup 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.

Named Entity Recognition

Named Entity Recognition is the process of tagging a word, or a substring within a query, with a singular category or topic, usually termed as a named entity. It is both a localization, and a classification task.

1.Pattern Based NER's: Regular Expressions and suffix/prefix based heuristics are developed to capture these entities as they have a specific pattern of occuring in a query string.

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: This is also referred to as Ontology, or Lexicon search. An external database comprising of category wise samples is referenced. 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.

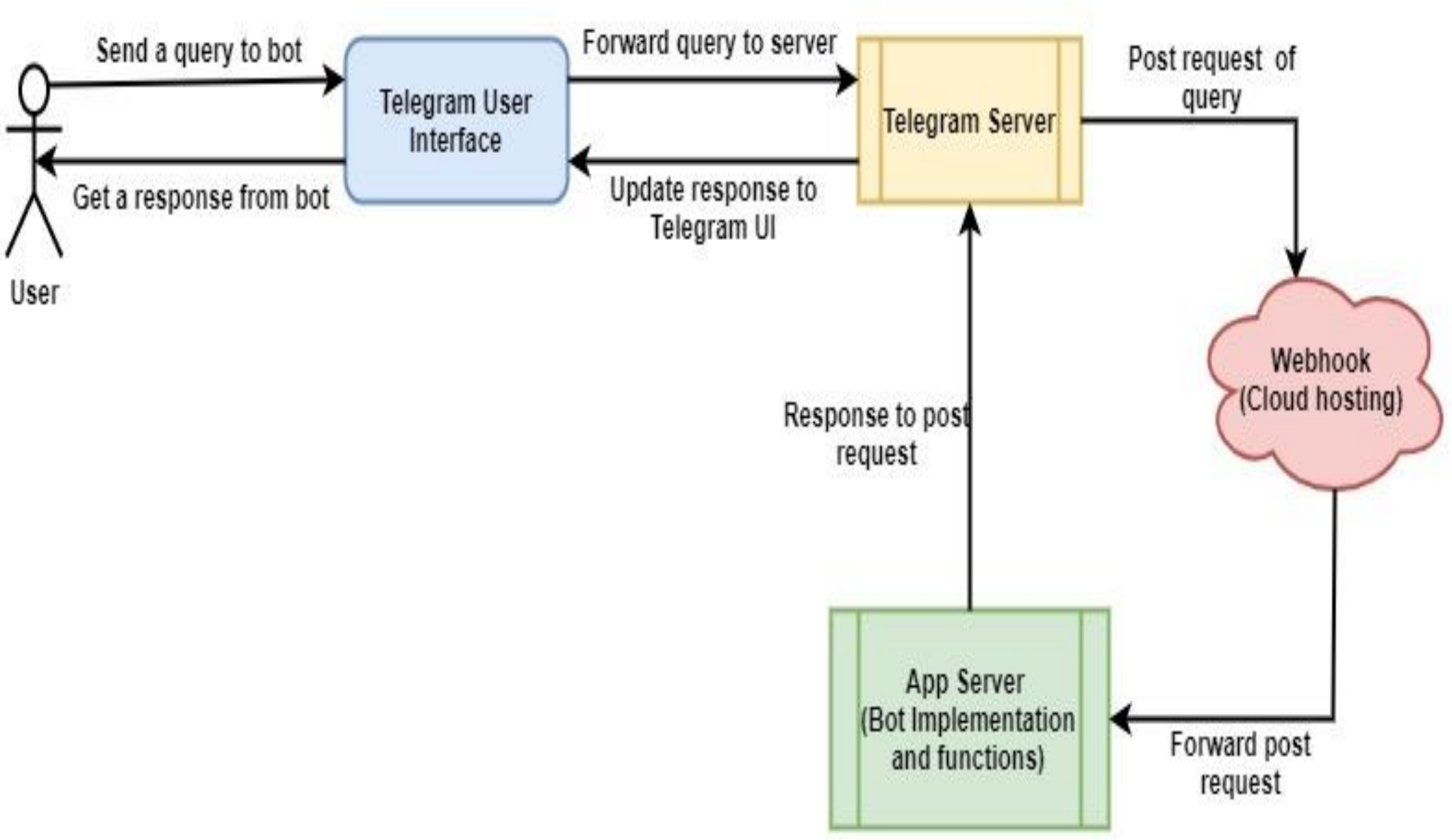
Eg: Location/Address, Hotel Name, Person Name etc

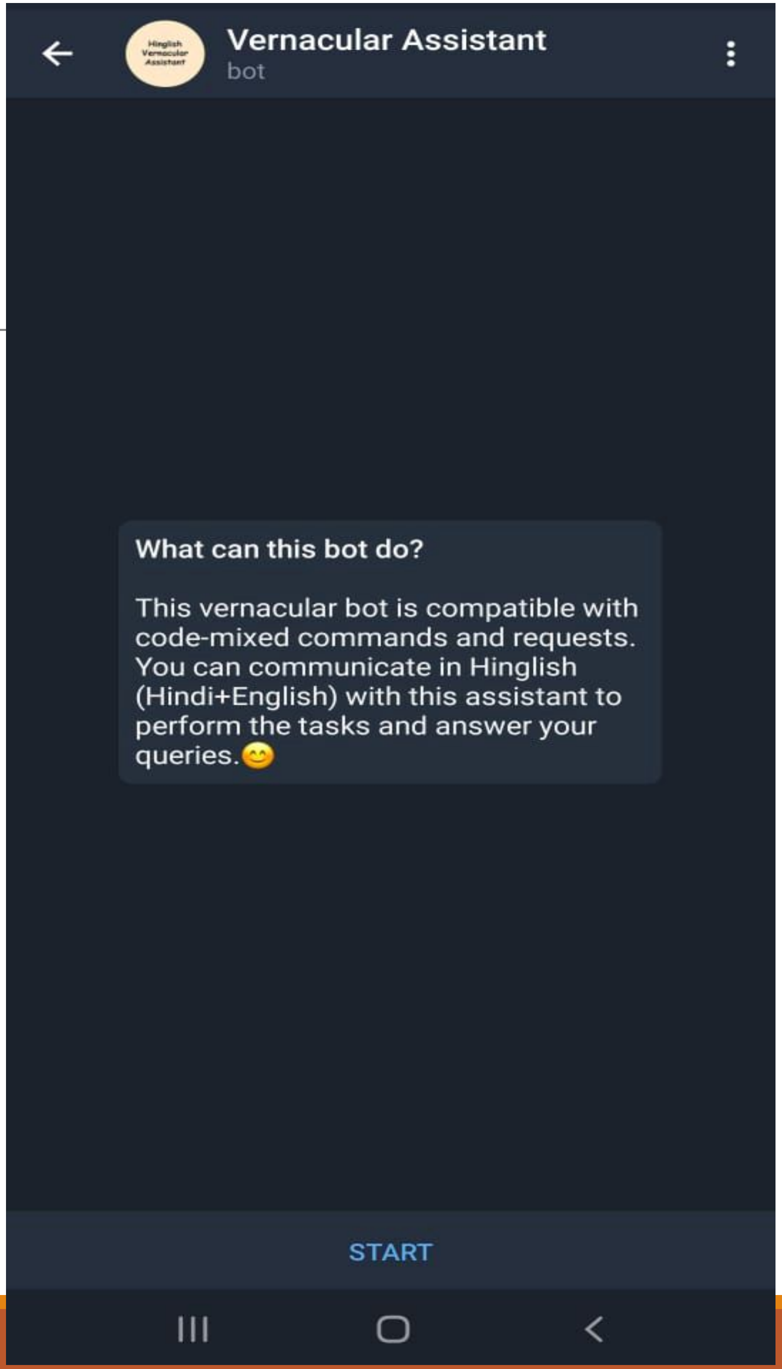
3.Context Based NER's: Uses words around a target word, to determine if it belongs to one of the many named entities. These models require the use of annotated data, with entities if present in a query, being correctly labelled. CRF's can be used to train a context based NER model.

Named Entity Recognition

Intent Name	Named Entities Involved
Hotel Booking	Hotel Name, Date, Time, Number of Reservations
Restaurant Booking	Restaurant Name, Date, TIme, Number of Reservations
Travel Booking	Date, Time, Location 1, Location 2, Vehicle Type
Reminder	Activity, Date, Time

Telegram User Interface



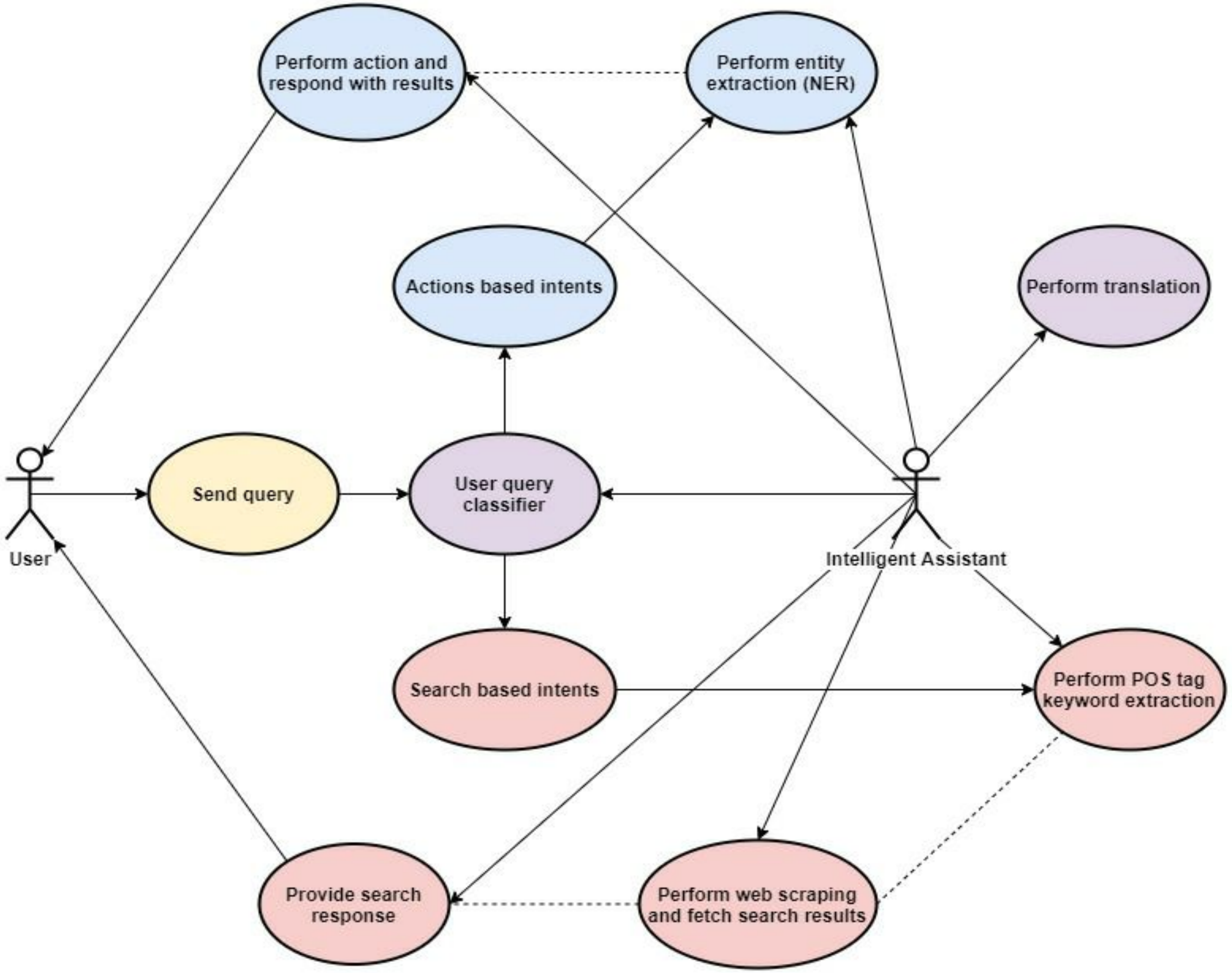


Tech stack

- **Scripting, API development and Server Management(Backend):** Python
- **Assistant Interface:** Telegram Messenger (python-telegram-bot library)
- **Text Processing:** iNLTK, NLTK
- **Translation API**: Google Translation API (googletrans python library)
- **Version control:** Git & GitHub
- **Language Identification and Transliteration:** litcm, indictrans
- **Web interface:** HTML, Javascript, python-requests(Flask)

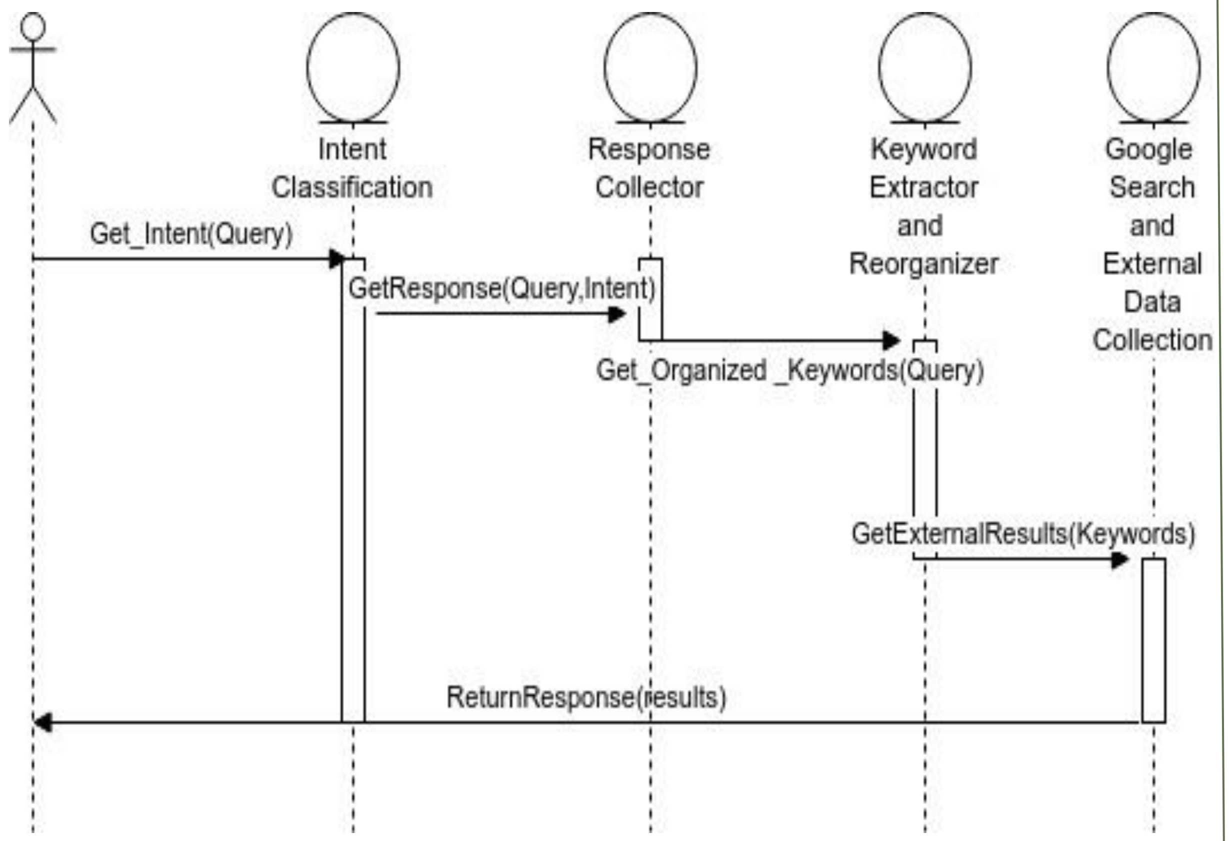
Modelling and Implementation

Use case model

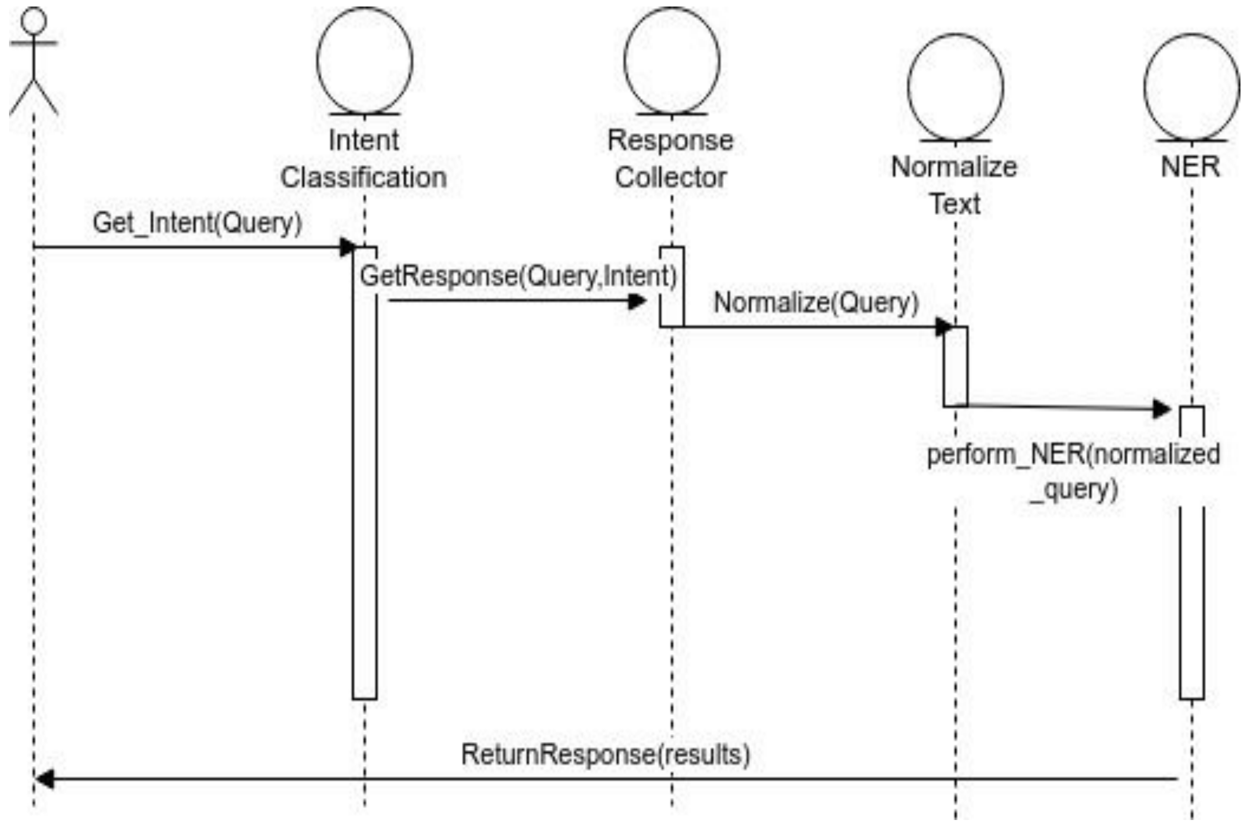


Sequence diagram

Sequence Diagram for Search based Intents

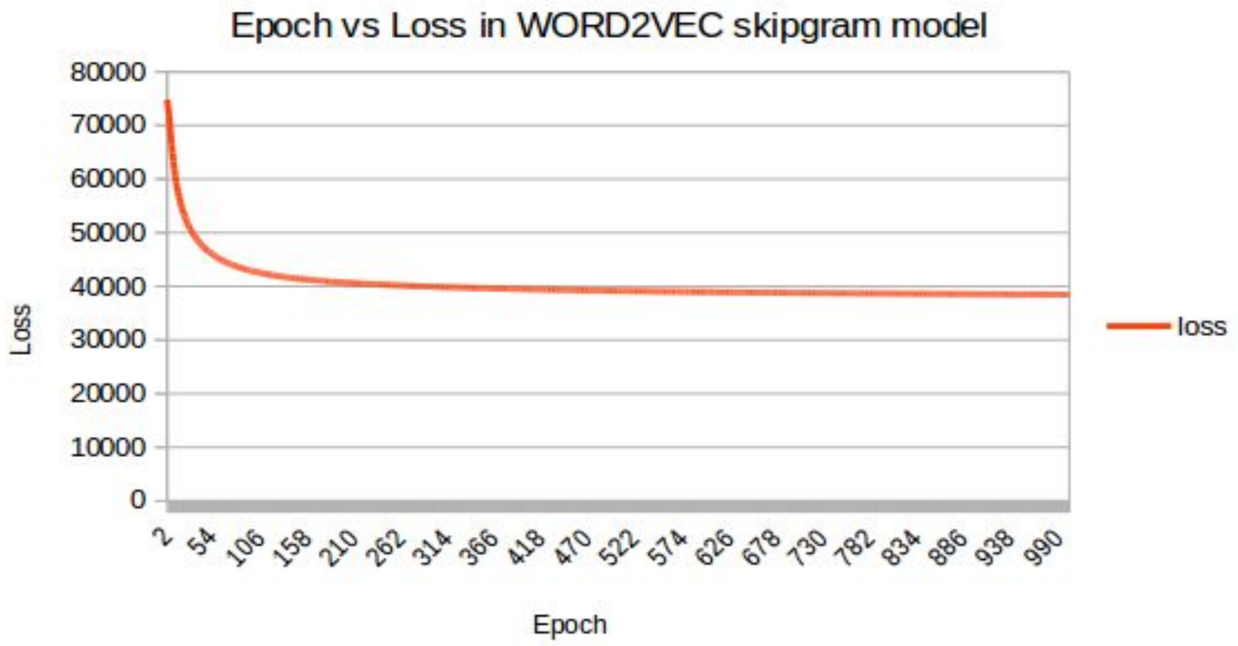


Sequence Diagram for Action based Intents

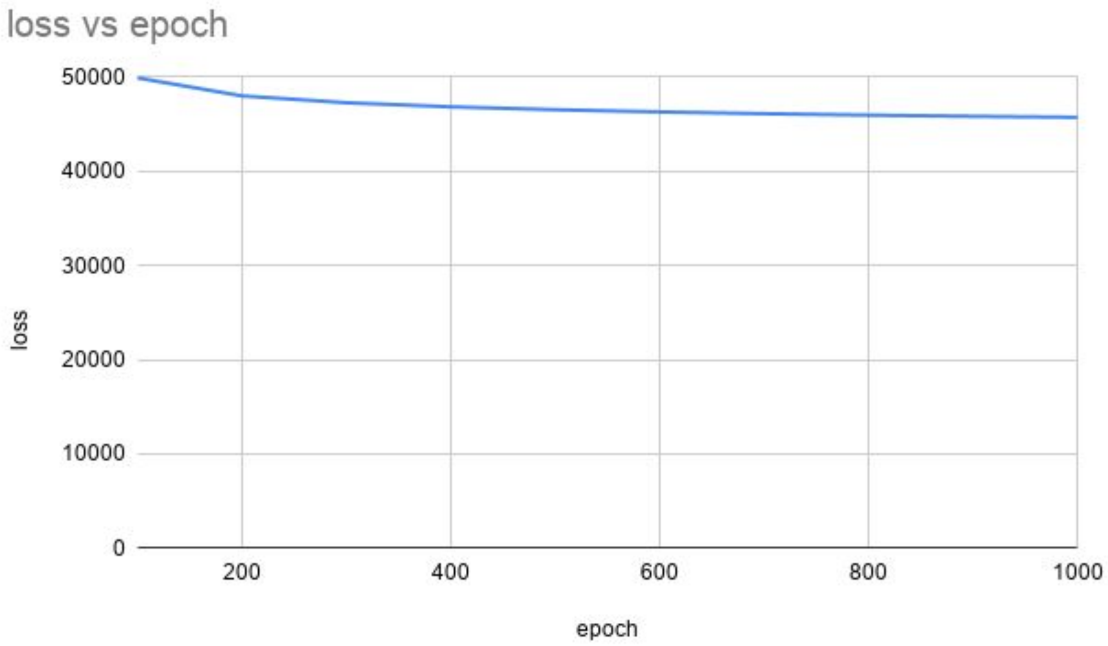


Testing, Results and Discussion

Word2Vec skip gram model loss per epochs



Hinglish Dataset



Kanglish Dataset

Eyeball method of testing for Word2Vec skip gram model

Threshold > 0.85

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

Hinglish Dataset

Word 1	Word 2	Cosine Similarity index	Hit
food	oota	0.9505	**HIT**
letter	patra	0.9721	**HIT**
lakshana	symptoms	0.8911	**HIT**
prabhaava	effect	0.97456	**HIT**
parinama	effects	0.9192	**HIT**
time	hotu	0.9788	**HIT**
show	torisu	0.8442	MISS

Kanglish Dataset

Accuracy in intent classification

Classifier	Countvectorizer	TF-IDF	Word2Vec
Naive Bayes Classifier	Hin:0.91265 Kan:0.95354	Hin:0.88253 Kan:0.90725	Hin:0.85240 Kan:0.89919
K-nearest Neighbour classifier	Hin;0.84036 Kan:0.81451	Hin:0.85240 Kan:0.85887	Hin:0.65060 Kan:0.60008
Random Forest Classifier	Hin:0.89759 Kan:0.87096	Hin:0.84939 Kan:0.84677	Hin:0.89457 Kan:0.85483
Linear Support Vector Classifier	**Hin:0.95180** **Kan:0.97580**	Hin:0.93975 Kan:0.93951	Hin:0.84638 Kan:0.91129
Logistic Regression classifier	Hin:0.94879 Kan:0.97580	Hin:0.92771 Kan:0.91935	Hin:0.44879 Kan:0.75
Decision Tree	Hin:0.90662 Kan:0.88306	Hin:0.82831 Kan:0.77822	Hin0.91265 Kan:0.77419

Precision in intent classification

Classifier	Countvectorizer	TF-IDF	Word2Vec
Naive Bayes Classifier	Hin:0.92 Kan:0.95	Hin:0.88 Kan:0.92	Hin:0.87 Kan:0.91
K-nearest Neighbour classifier	Hin0.88 Kan:0.87	Hin:0.86 Kan:0.88	Hin:0.77 Kan:0.79
Random Forest Classifier	Hin:0.91 Kan:0.89	Hin:0.87 Kan:0.88	Hin:0.91 Kan:0.87
Linear Support Vector Classifier	**Hin:0.96** **Kan:0.98**	Hin:0.95 Kan:0.95	Hin:0.87 Kan:0.92
Logistic Regression classifier	Hin:0.95 Kan:0.98	Hin:0.93 Kan:0.93	Hin:0.67 Kan:0.80
Decision Tree	Hin:0.92 Kan:0.90	Hin:0.84 Kan:0.83	Hin:0.93 Kan:0.82

Recall in intent classification

Classifier	Countvectorizer	TF-IDF	Word2Vec
Naive Bayes Classifier	Hin:0.91 Kan:0.94	Hin:0.88 Kan:0.91	Hin:0.85 Kan:0.90
K-nearest Neighbour classifier	Hin:0.84 Kan:0.81	Hin:0.85 Kan:0.86	Hin:0.65 Kan:0.60
Random Forest Classifier	Hin:0.90 Kan:0.87	Hin:0.85 Kan:0.85	Hin:0.89 Kan:0.85
Linear Support Vector Classifier	**Hin:0.95** **Kan:0.98**	Hin:0.94 Kan:0.94	Hin:0.85 Kan:0.91
Logistic Regression classifier	**Hin:0.95** **Kan:0.98**	Hin:0.93 Kan:0.92	Hin:0.45 Kan:0.75
Decision Tree	Hin:0.91 Kan:0.88	Hin:0.83 Kan:0.78	Hin:0.91 Kan:0.77

F1-score in intent classification on Hinglish dataset

Classifier	**Countvectorizer**	**TF-IDF**	**Word2Vec**
Naive Bayes Classifier	Hin:0.91 Kan:0.94	Hin:0.88 Kan:0.91	Hin:0.85 Kan:0.90
K-nearest Neighbour classifier	Hin:0.84 Kan:0.81	Hin:0.85 Kan:0.86	Hin:0.67 Kan:0.61
Random Forest Classifier	Hin:0.90 Kan:0.87	Hin:0.85 Kan:0.85	Hin:0.89 Kan:0.85
Linear Support Vector Classifier	**Hin:0.95** **Kan:0.98**	Hin:0.94 Kan:0.94	Hin:0.85 Kan:0.91
Logistic Regression classifier	**Hin:0.95** **Kan:0.98**	Hin:0.93 Kan:0.92	Hin:0.44 Kan:0.74
Decision Tree	Hin:0.91 Kan:0.88	Hin:0.83 Kan:0.77	Hin:0.91 Kan:0.78

Classification Metrics for NER with respect to Intents

Intent	Precision	Recall	F1-Score
Hotel Booking	0.96	0.712	0.8317
Restaurant Booking	1.0	0.60	0.75
Travel Booking	1.0	0.89	0.9412
Reminder	0.91	0.82	0.8615

Hinglish Dataset

Intent	Precision	Recall	F Score
Hotel Booking	0.89	0.35	0.5024
Restaurant Booking	0.96	0.6515	0.776
Travel Booking	1.00	0.60	0.75
Reminder	0.68	0.62	0.6486

Kanglish Dataset

Demonstration

Conclusion

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 researching on 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.

Further work

1. Building the intent database for further Indian regional languages
2. Build a robust Kannada-English POS tagger
3. Pretrained Multilingual Supervised Word Embeddings(MUSE) published by Facebook Research can represent words of two or more than two different languages in the same embedding space.
4. 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.

Thank you.

Any questions ?