Enhancing Conversational AI Model Performance and Explainability for Sinhala-English Bilingual Speakers

2022-056

The Team

Supervisor
Dr. Lakmini Abeywardhana

Co - Supervisor
Ms. Dinuka Wijendra

Dissanayake D.M.I.M.
IT19069432

Hameed M.S.
IT19064932

Jayasinghe D.T.
IT19075754

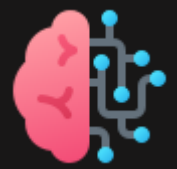
Sakalasooriya S.A.H.A.
IT19051208

Overall

Project Description

1. Research Background
2. Research Problem
3. Research gap
4. Research Objectives
5. Overall system architecture
6. Gantt chart

Background



Chatbots versus Conversational AIs

Chatbots vs Conversational AIs

Why NLP?

Entities and Intents

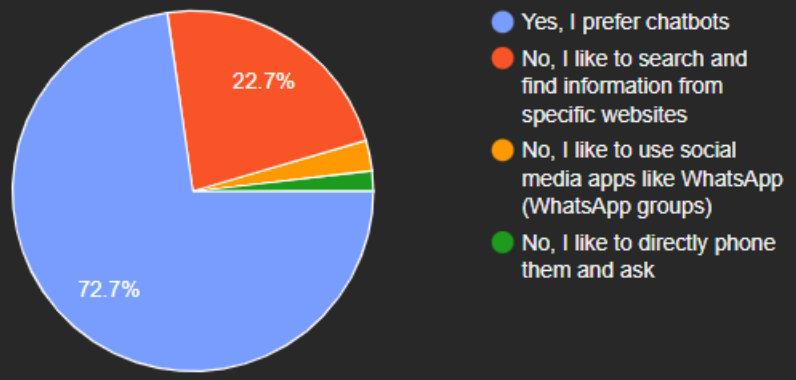
Sparse and Dense Feature Engineering

Maintenance & Testing

Survey Findings

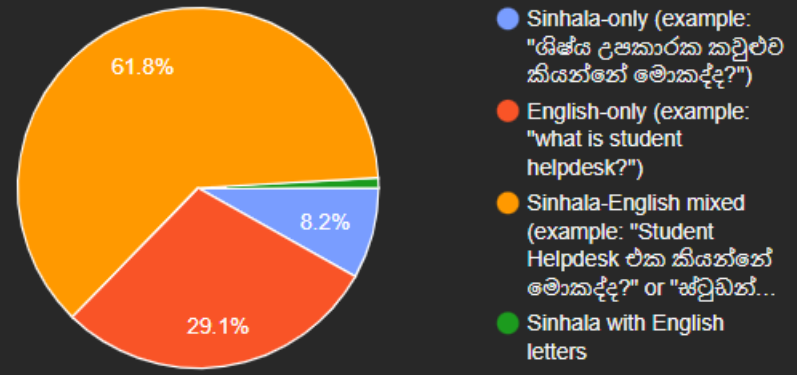
Do you prefer to find Information using chatbots or by surfing the internet? (for example: finding available degrees at SLIIT) 🌍

110 responses



What languages would you prefer to use the most for chatting if you had to interact with a chatbot? 🙋 ❓

110 responses



Research Problem

What's Lacking?

Support for Sinhala-English code-switched text processing in chatbots

Explanations for how chatbot models make predictions

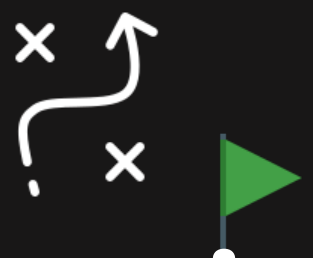
Bilingual Speakers and Keyboard Interfaces

Maintenance and Evaluations for Non-ML Experts

Research Gap

A Brief Comparison

Tool / Research	Integrated Explainable AI Support	Integrated Code-less Data Improvement	Model Evaluation Expertise	Equivalent Token Mapping	Entity Annotation
Rasa Open Source	No	No	ML Expert	No	Manual, one by one
DialogFlow	No	Yes, Cloud based	Only Analytics. No Model Evaluation	No	Automated Expansion, Fuzzy Matching
Sally (This Research)	Yes	Yes	None ML Expert	Yes	Variation Matching



Objectives

What to Expect?

NLP Tools attachable to modern chatbot frameworks for Sinhala-English Code-Switched text

Efficient Entity Annotation

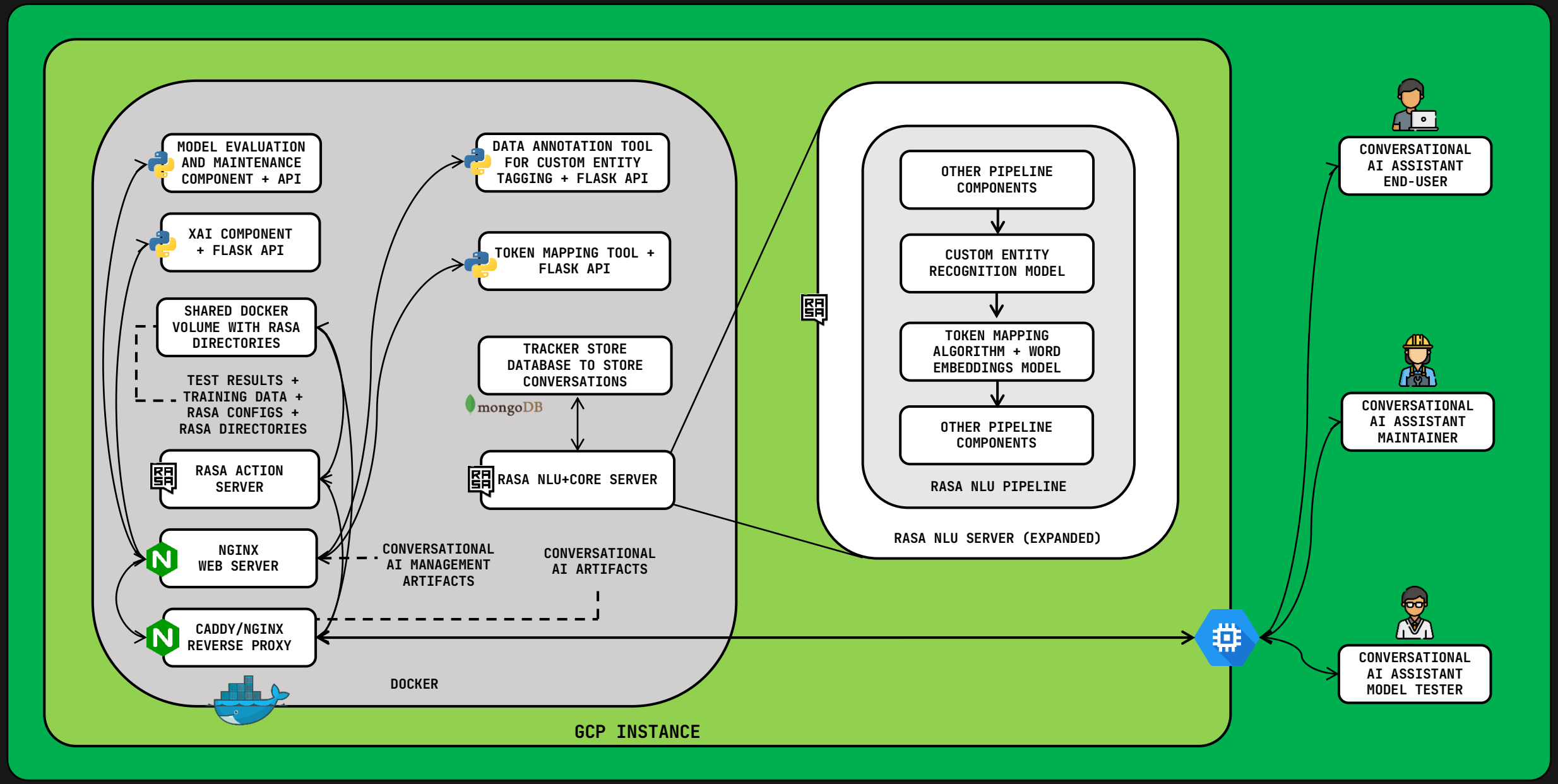
Bilingual Keyboard Interface

ML model Explanations for chatbots

Code-less Chatbot Maintenance

ML Model Evaluations for Non-Experts

Overall Architecture


MODEL EVALUATION AND MAINTENANCE COMPONENT + API
XAI COMPONENT + FLASK API
SHARED DOCKER VOLUME WITH RASA DIRECTORIES
TEST RESULTS + TRAINING DATA + RASA CONFIGS + RASA DIRECTORIES
RASA ACTION SERVER
NGINX WEB SERVER
CADDY/NGINX REVERSE PROXY
DOCKER
DATA ANNOTATION TOOL FOR CUSTOM ENTITY TAGGING + FLASK API
TOKEN MAPPING TOOL + FLASK API
TRACKER STORE DATABASE TO STORE CONVERSATIONS
mongoDB
RASA NLU+CORE SERVER
CONVERSATIONAL AI MANAGEMENT ARTIFACTS
CONVERSATIONAL AI ARTIFACTS
GCP INSTANCE
OTHER PIPELINE COMPONENTS
CUSTOM ENTITY RECOGNITION MODEL
TOKEN MAPPING ALGORITHM + WORD EMBEDDINGS MODEL
OTHER PIPELINE COMPONENTS
RASA NLU PIPELINE
RASA NLU SERVER (EXPANDED)
CONVERSATIONAL AI ASSISTANT END-USER
CONVERSATIONAL AI ASSISTANT MAINTAINER
CONVERSATIONAL AI ASSISTANT MODEL TESTER

Technologies

Backend Development

- Python
- TensorBoard
- Rasa
- pandas
- spaCy
- Flask
- Gensim

Frontend Development

- React
- Socket.IO

NoSQL Database Development

- MongoDB

Technologies

Deployment

Source-code management

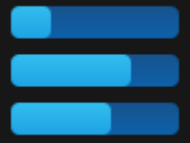
Docker

Nginx

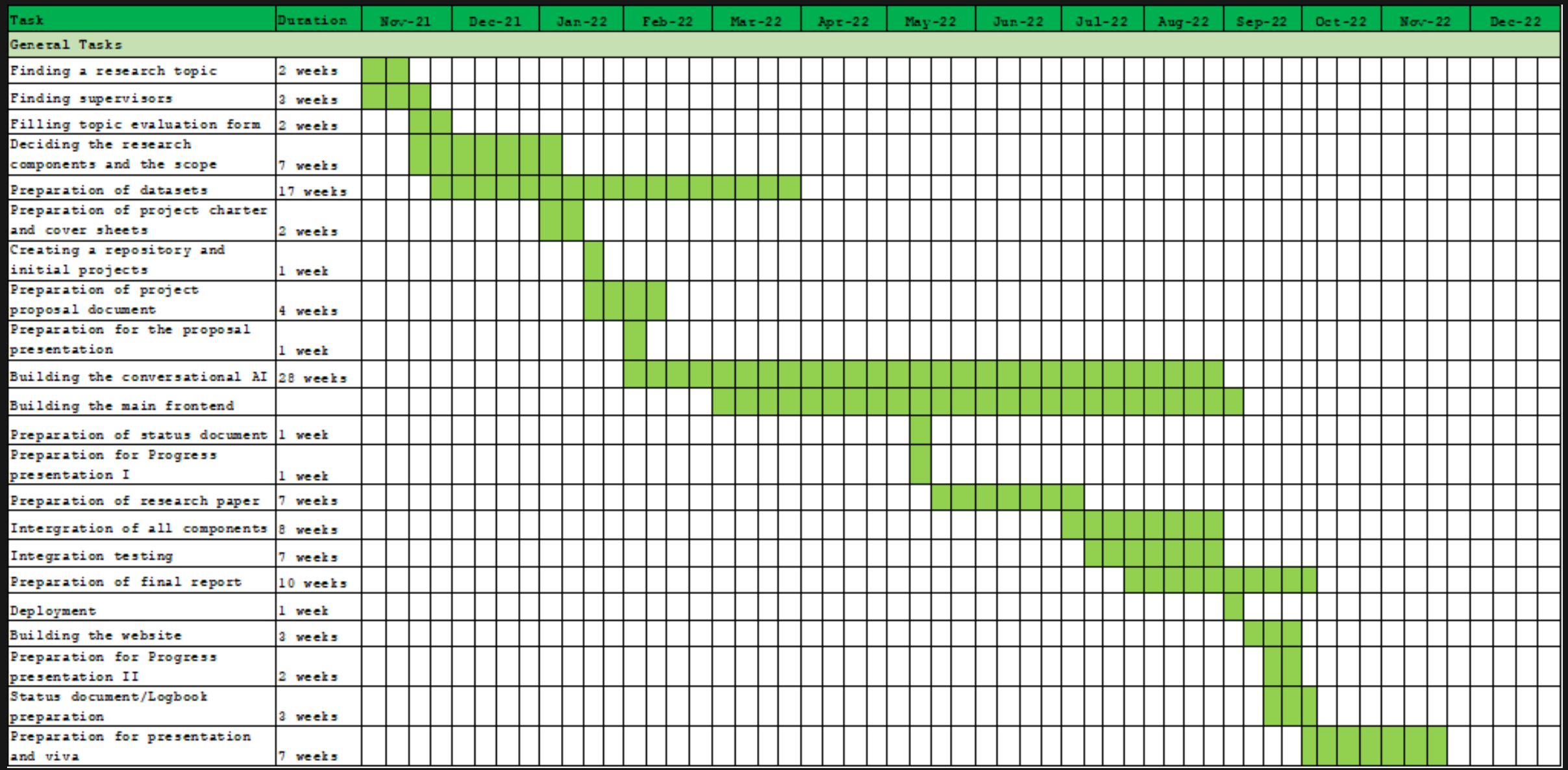
Git

GCP

Caddy



Gantt Chart


Task
Duration
Nov-21
Dec-21
Jan-22
Feb-22
Mar-22
Apr-22
May-22
Jun-22
Jul-22
Aug-22
Sep-22
Oct-22
Nov-22
Dec-22
General Tasks
Finding a research topic
2 weeks
Finding supervisors
3 weeks
Filling topic evaluation form
2 weeks
Deciding the research components and the scope
7 weeks
Preparation of datasets
17 weeks
Preparation of project charter and cover sheets
2 weeks
Creating a repository and initial projects
1 week
Preparation of project proposal document
4 weeks
Preparation for the proposal presentation
1 week
Building the conversational AI
28 weeks
Building the main frontend
Preparation of status document
1 week
Preparation for Progress presentation I
1 week
Preparation of research paper
7 weeks
Intergration of all components
8 weeks
Integration testing
7 weeks
Preparation of final report
10 weeks
Deployment
1 week
Building the website
3 weeks
Preparation for Progress presentation II
2 weeks
Status document/Logbook preparation
3 weeks
Preparation for presentation and viva
7 weeks

DIME: Dual Interpretable Model-Agnostic Explanations

Using global explanations to generate local interpretations in intent classification models using explainable AI

Dissanayake D.M.I.M.
IT19069432
Data Science

Research Component
Background

What is XAI?

Accuracy-Interpretability Trade-off

Global vs Local Explanations

Research Component
Gap

Research/ Tool	Intrinsic/ Post hoc	Local/ Global Explanations	Model Specificity	Feature Engineering	Feature Contribution Score
LIME [1]	Post hoc	Local	Model Agnostic	Ridge Regression	Local Linear Surrogate Model
SHAP [2]	Post hoc	Global or Local	Model Agnostic	All features with Approximation	Shapley Values
DIME (This Research)	Post hoc	Local using Global	Model Agnostic	Global Feature Importance	Shapley Values

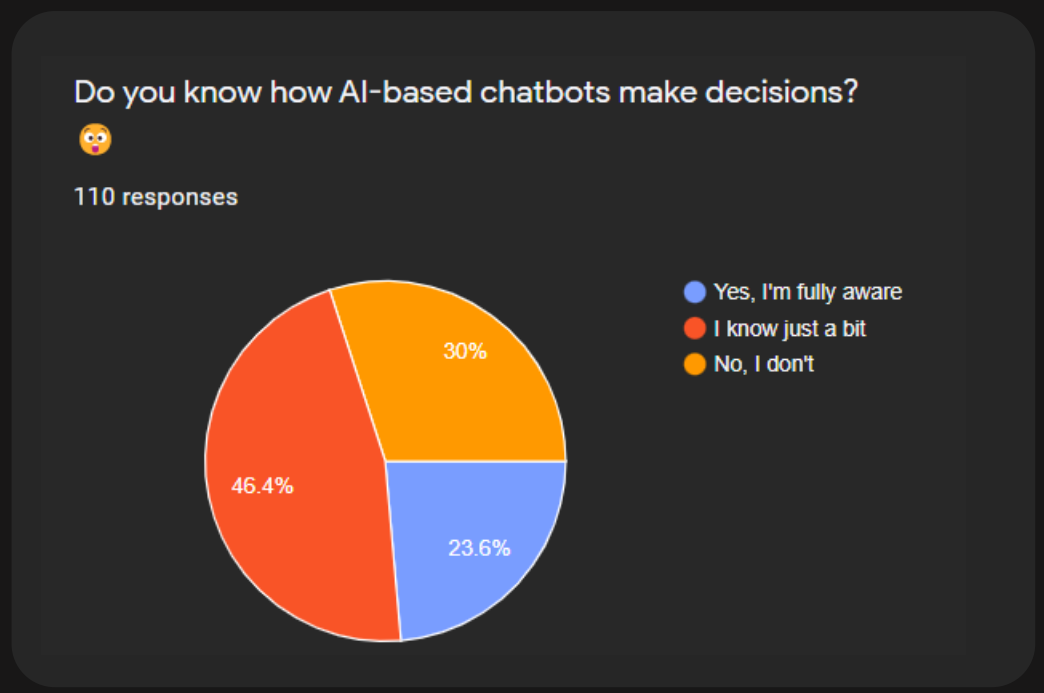
Research Component
Question

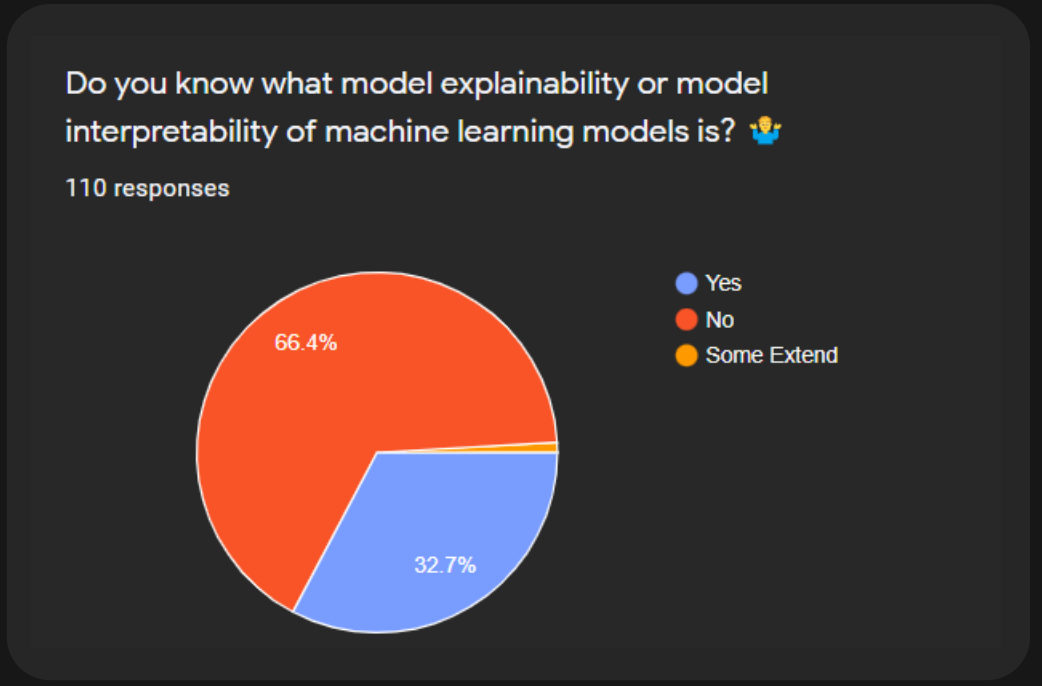
Cannot Trust & Debug ML Models

Why not blending Local & Global Explanations?

Explainability in modern Chatbot Frameworks

Survey Findings





Specific Objective

Develop DIME, an Explainable AI approach to deliver local model explanations with the help of global feature importance.

Sub Objectives

Find Global and Local Explanations logically

Develop a python package for DIME

Modify DIET intent classifier to get all confidence scores

Visualize explanations in an interpretable manner

Integrate DIME with Rasa seamlessly

Functional Requirements

DIME should provide methods to calculate global and local explanations

DIME should provide a local server as a visualization tool

DIME should be applicable to any ML model that outputs confidence scores for predictions

DIME should ask the users number of features to generate explanations

DIME should utilize caching to optimize calculations

Non-functional Requirements

- Efficient Calculations
- Reliable Explanations
- Simple & Interpretable Visualizations
- Modular package

Proposed Methodology

1. Build a Dataset for Rasa conversational AI Training
2. Train a Rasa Conversational AI
3. Develop the DIME Algorithm Using the Trained Model
4. Develop DIME Server to Support any Rasa Chatbot by default
5. Compare Different Techniques for Global Feature Importance
6. Evaluate DIME & Integrate with the Overall System

Global feature Importance

Finding Global Feature Importance for “මොනවද”:

Original Dataset →  → Observe the Accuracy

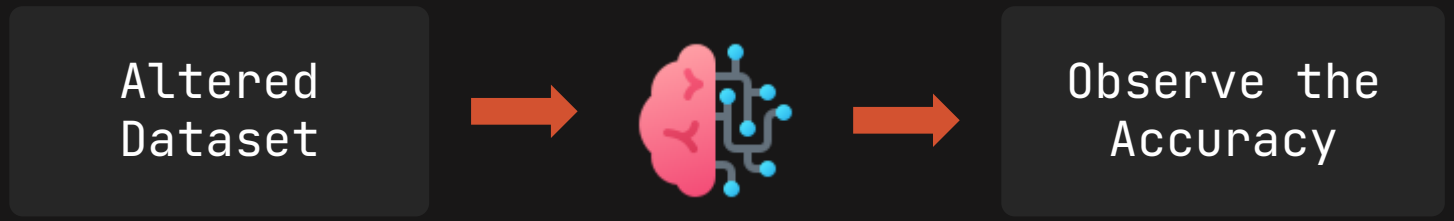
1. ඔයාලගෙ මොනවද තියෙන ඩිග්‍රී?
2. SLIIT එකේ තියෙන උපාධි මොනවද?
3. Foundation courses කියන්නෙ මොනවද
4. Repeat එකකට කියක් වෙනවද
...

Correct
Correct
Incorrect
Correct
...

Accuracy for original dataset = 3/4

Global feature Importance

Finding Global Feature Importance for “මොනවද”:



1. ඔයාලගෙ මොනවද තියෙන ඩිග්‍රී?
2. SLIIT එකේ තියෙන උපාධි මොනවද?
3. Foundation courses කියන්නෙ මොනවද
4. Repeat එකකට කියක් වෙනවද
...

Correct
Incorrect
Incorrect
Correct
...

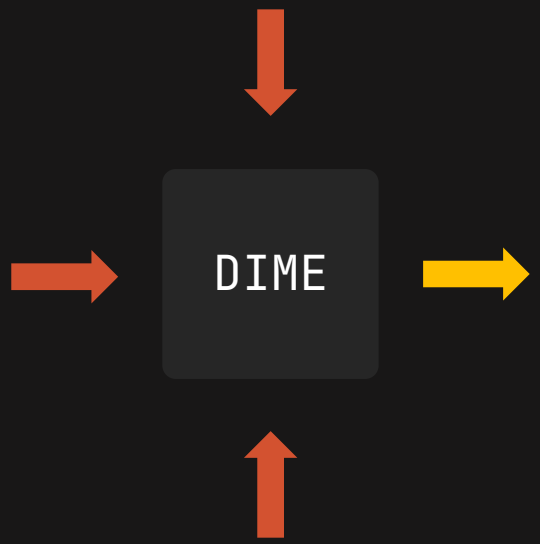
Accuracy for original dataset = 1/2
Global Feature Importance for “මොනවද” = change in the accuracy = 0.25

Local Model Explanations

Global Feature Importance Scores:

ඩිග්‍රී	0.2062
SLIIT	0.1401
එකේ	0.0001
උපාධි	0.0012
මොනවද	0.0001
Foundation	0.0089
කියන්නෙ	0.0001
Repeat	0.0099
එකකට	0.0003
කීයක්	0.002
...	...

Number of Features: 2



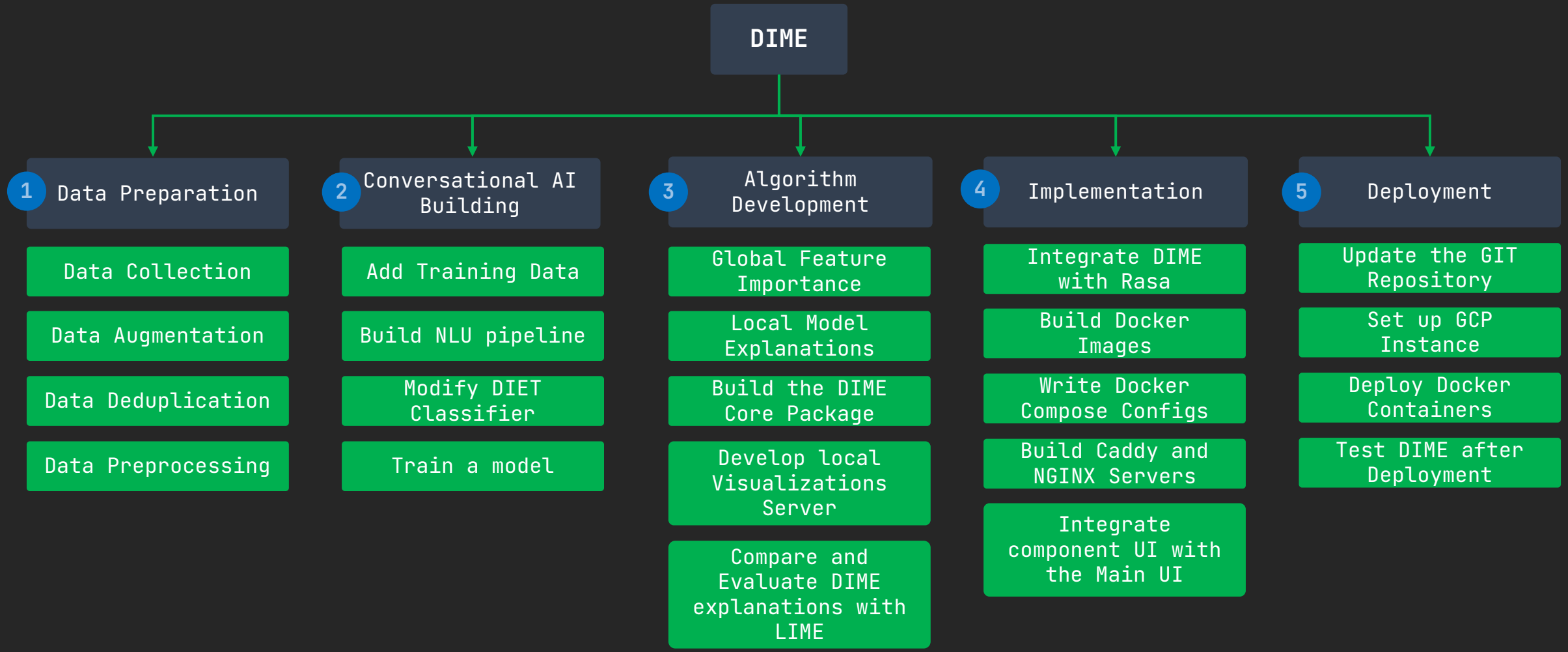
Local Example:

ඔයාලගෙ තියෙන උපාධි වර්ග මොනවද?

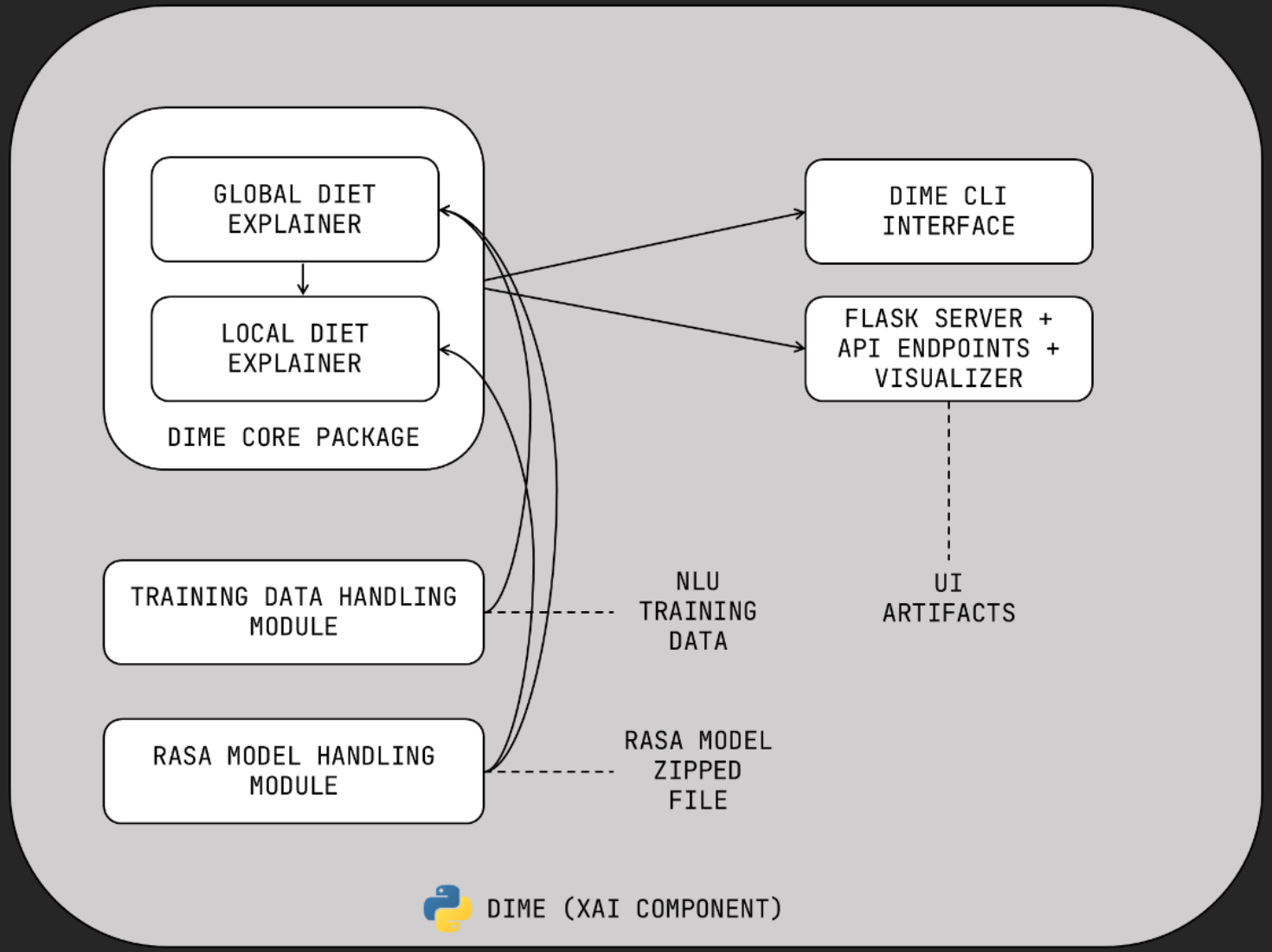
Predicted Class: Degrees

Explanations Plot:

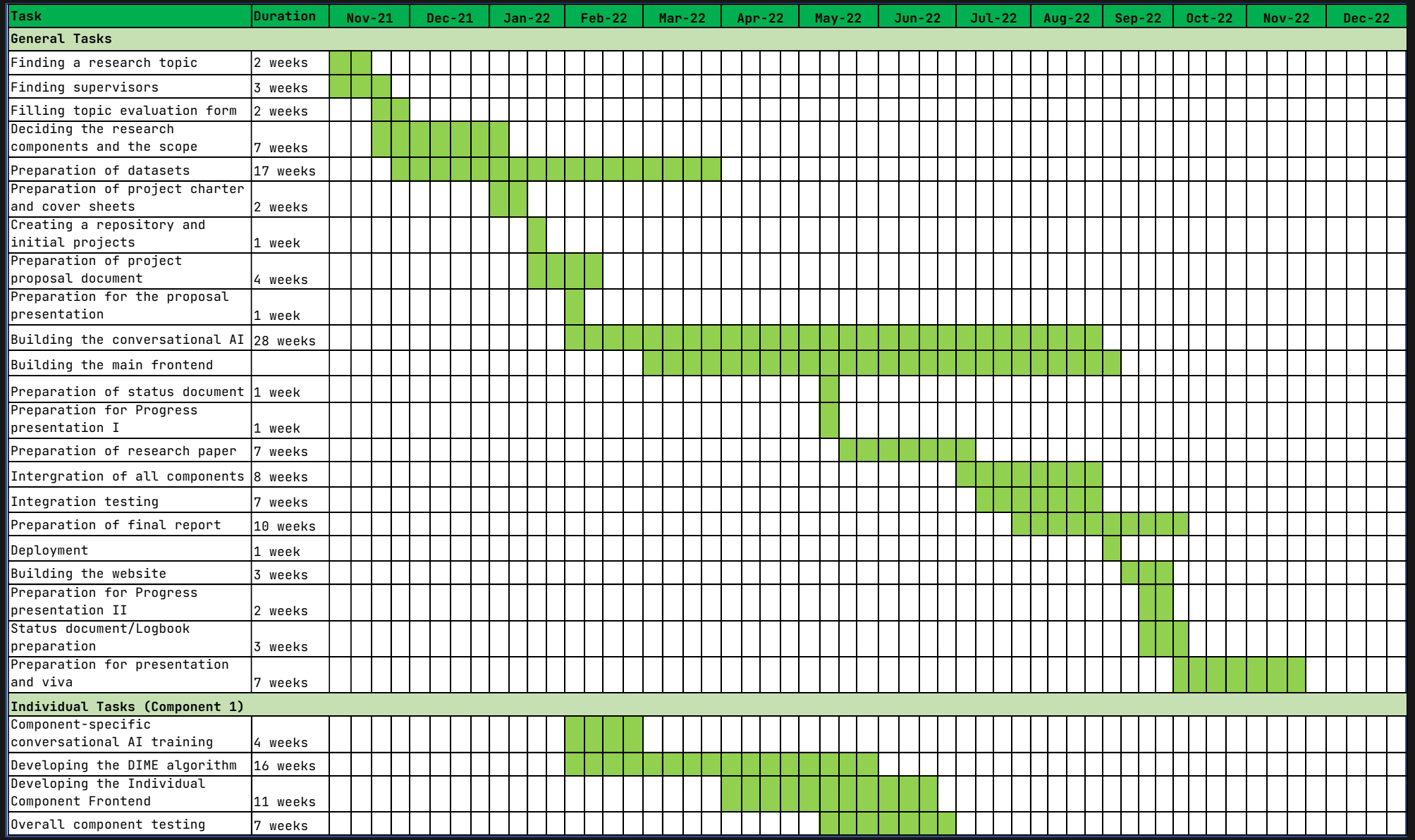
Work Breakdown Structure


DIME
1 Data Preparation
Data Collection
Data Augmentation
Data Deduplication
Data Preprocessing
2 Conversational AI Building
Add Training Data
Build NLU pipeline
Modify DIET Classifier
Train a model
3 Algorithm Development
Global Feature Importance
Local Model Explanations
Build the DIME Core Package
Develop local Visualizations Server
Compare and Evaluate DIME explanations with LIME
4 Implementation
Integrate DIME with Rasa
Build Docker Images
Write Docker Compose Configs
Build Caddy and NGINX Servers
Integrate component UI with the Main UI
5 Deployment
Update the GIT Repository
Set up GCP Instance
Deploy Docker Containers
Test DIME after Deployment

Individual Component Architecture


GLOBAL DIET EXPLAINER
LOCAL DIET EXPLAINER
DIME CORE PACKAGE
DIME CLI INTERFACE
FLASK SERVER + API ENDPOINTS + VISUALIZER
TRAINING DATA HANDLING MODULE
NLU TRAINING DATA
UI ARTIFACTS
RASA MODEL HANDLING MODULE
RASA MODEL ZIPPED FILE
DIME (XAI COMPONENT)

Individual Gantt Chart


Task
Duration
Nov-21
Dec-21
Jan-22
Feb-22
Mar-22
Apr-22
May-22
Jun-22
Jul-22
Aug-22
Sep-22
Oct-22
Nov-22
Dec-22
General Tasks
Finding a research topic
2 weeks
Finding supervisors
3 weeks
Filling topic evaluation form
2 weeks
Deciding the research components and the scope
7 weeks
Preparation of datasets
17 weeks
Preparation of project charter and cover sheets
2 weeks
Creating a repository and initial projects
1 week
Preparation of project proposal document
4 weeks
Preparation for the proposal presentation
1 week
Building the conversational AI
28 weeks
Building the main frontend
Preparation of status document
1 week
Preparation for Progress presentation I
1 week
Preparation of research paper
7 weeks
Intergration of all components
8 weeks
Integration testing
7 weeks
Preparation of final report
10 weeks
Deployment
1 week
Building the website
3 weeks
Preparation for Progress presentation II
2 weeks
Status document/Logbook preparation
3 weeks
Preparation for presentation and viva
7 weeks
Individual Tasks (Component 1)
Component-specific conversational AI training
4 weeks
Developing the DIME algorithm
16 weeks
Developing the Individual Component Frontend
11 weeks
Overall component testing
7 weeks

References

[1]. M. T. Ribeiro, S. Singh, C. Guestrin, "'why should I trust you?': Explaining the predictions of any classifier," 16 Feb 2016. [Online]. Available: https://arxiv.org/abs/1602.04938

[2]. S. Lundberg, S. Lee, "A Unified Approach to Interpreting Model Predictions," 2017. [Online]. Available: https://arxiv.org/abs/1705.078744

[3]. T. Bunk, D. Varshneya, V. Vlasov, A. Nichol, "DIET: Lightweight Language Understanding for Dialogue Systems," 2020. [Online]. Available: https://arxiv.org/abs/2004.09936

Code-less Maintenance and Model Performance Evaluation

Enabling non-machine learning experts to effectively improve and evaluate conversational AI machine learning models

Hameed M.S.
IT19064932
Software Engineering

Research Background

Training Machine Learning Models

Model Performance Evaluation and Tools used

Model data Improvement and Tools used

Research Gap for Data Improvement

Name/Reference	Tool/Research	Requires knowledge about the backend	Conversation Driven Development	Requires knowing CLI commands
Rasa [1]	Tool	Yes	No	Yes
Rasa X [2]	Tool	Yes	Yes	No
This Research Component	Tool	No	Yes	No

Research Gap for Model Evaluation

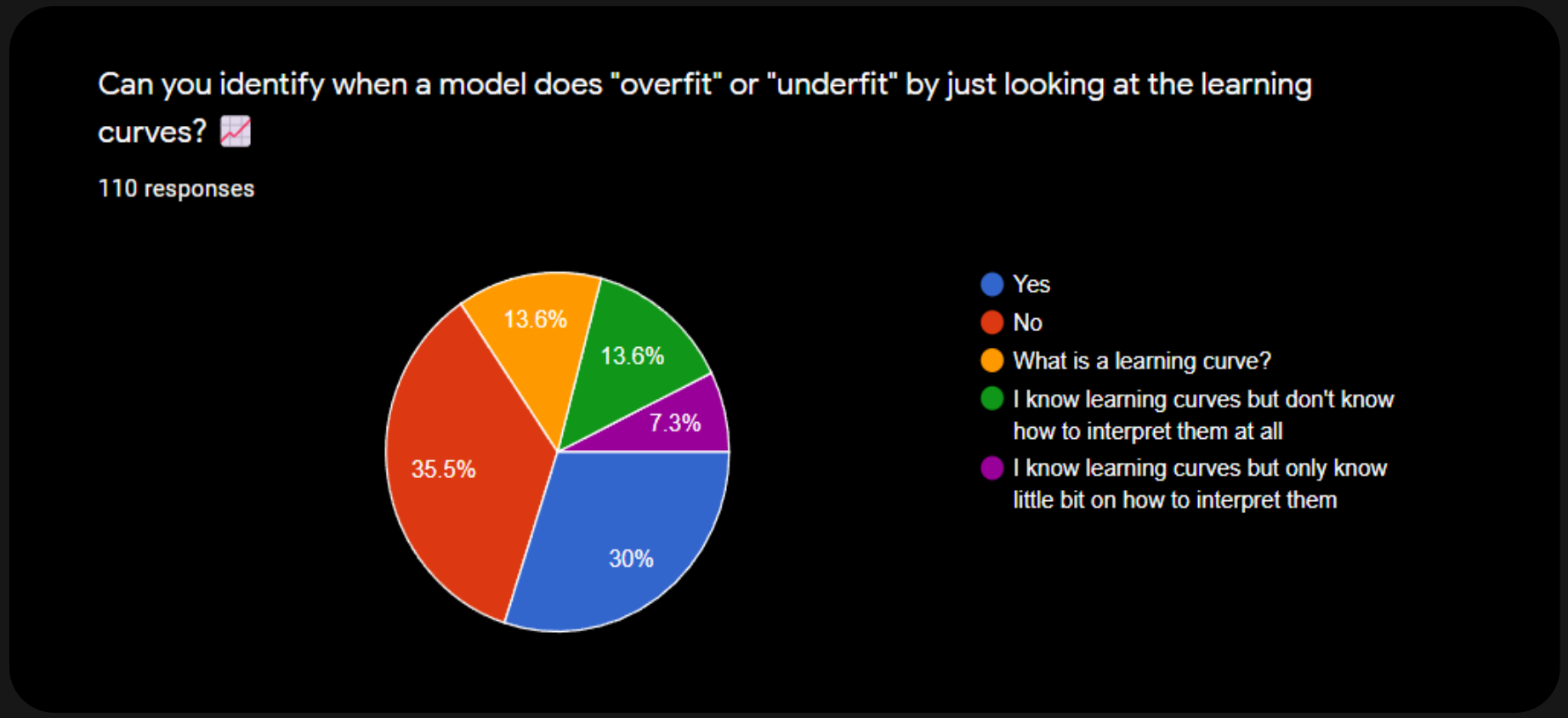
Name/Reference	Tool/Research	Overfitting/Underfitting indicators	Suggest the best epoch range to train the model	Requires knowledge about the backend to understand the feedback
Rasa [1]	Tool	No	Yes (Early stopping) but no visual indication	Yes
This Research Component (LCE)	Tool	Yes	Yes	No

Research Question

Can machine learning models be improved using Conversation Driven Development without having the knowledge to handle the backend?

Can machine learning model performance be evaluated without knowing how to interpret learning curves?

Survey Findings



Specific Objective

Develop an efficient and code-less approach to improve and evaluate conversational AI machine learning models for non-machine learning experts

Sub Objectives

Developing an interface to allow making improvements to model training data without any coding knowledge.

Developing a solution for non-technical users to efficiently retrain and deploy new machine learning models.

Developing an algorithm to identify any overfittings or underfittings in a model and indicate it in the frontend.

Research Component
Requirements

Functional Requirements	Non-Functional Requirements
LCE should denote any overfittings or underfittings in the trained model	Model evaluations should be reliable
LCE should suggest the appropriate range of epochs to use to train the model	Model evaluations should be consistent
Should be able to use conversations users have had with the conversational AI, to improve the machine learning model (CDD)	LCE should not hinder the model training time

*LCE = Learning Curve Explainer

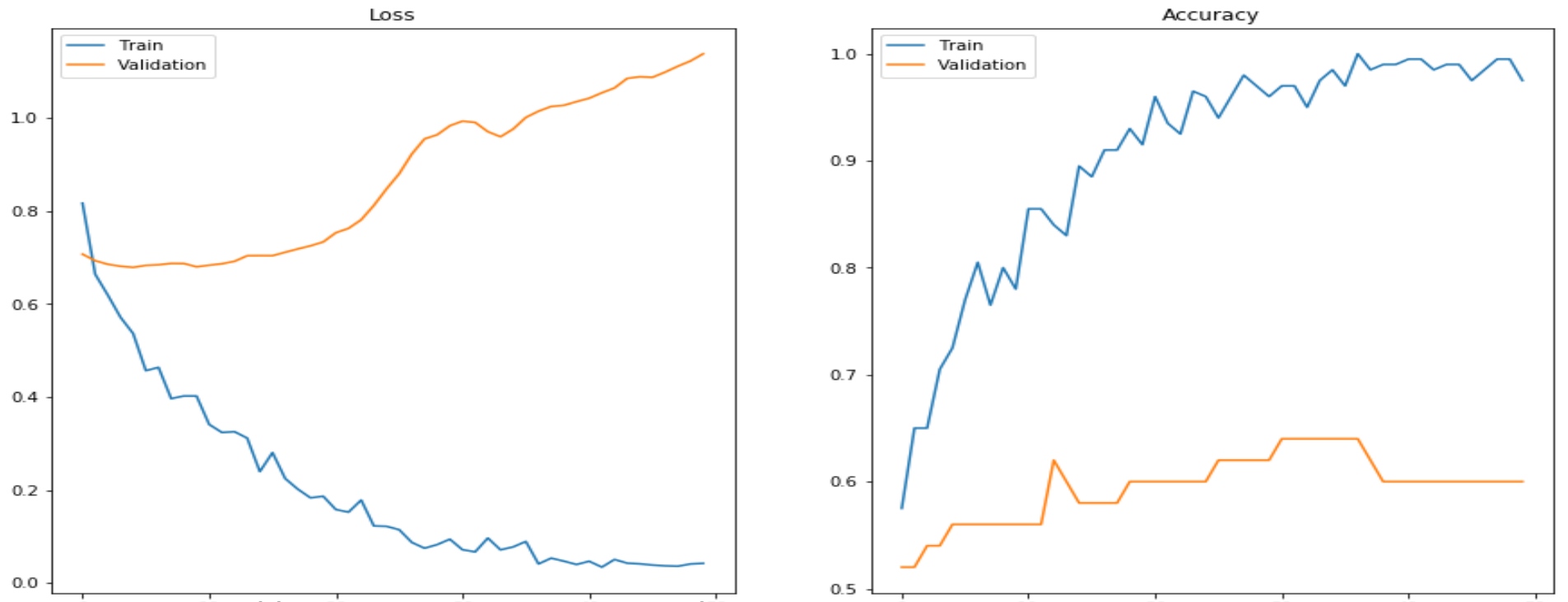
Proposed Methodology

1. Build a Dataset for Rasa conversational AI Training
2. Train a Rasa Conversational AI
3. Develop the LCE Algorithm Using the Trained Model
4. Develop LCE Server to Support any Rasa Chatbot by default
5. Compare Different Approaches for Smoothing Techniques
6. Integrate with the Overall System

Model Evaluation

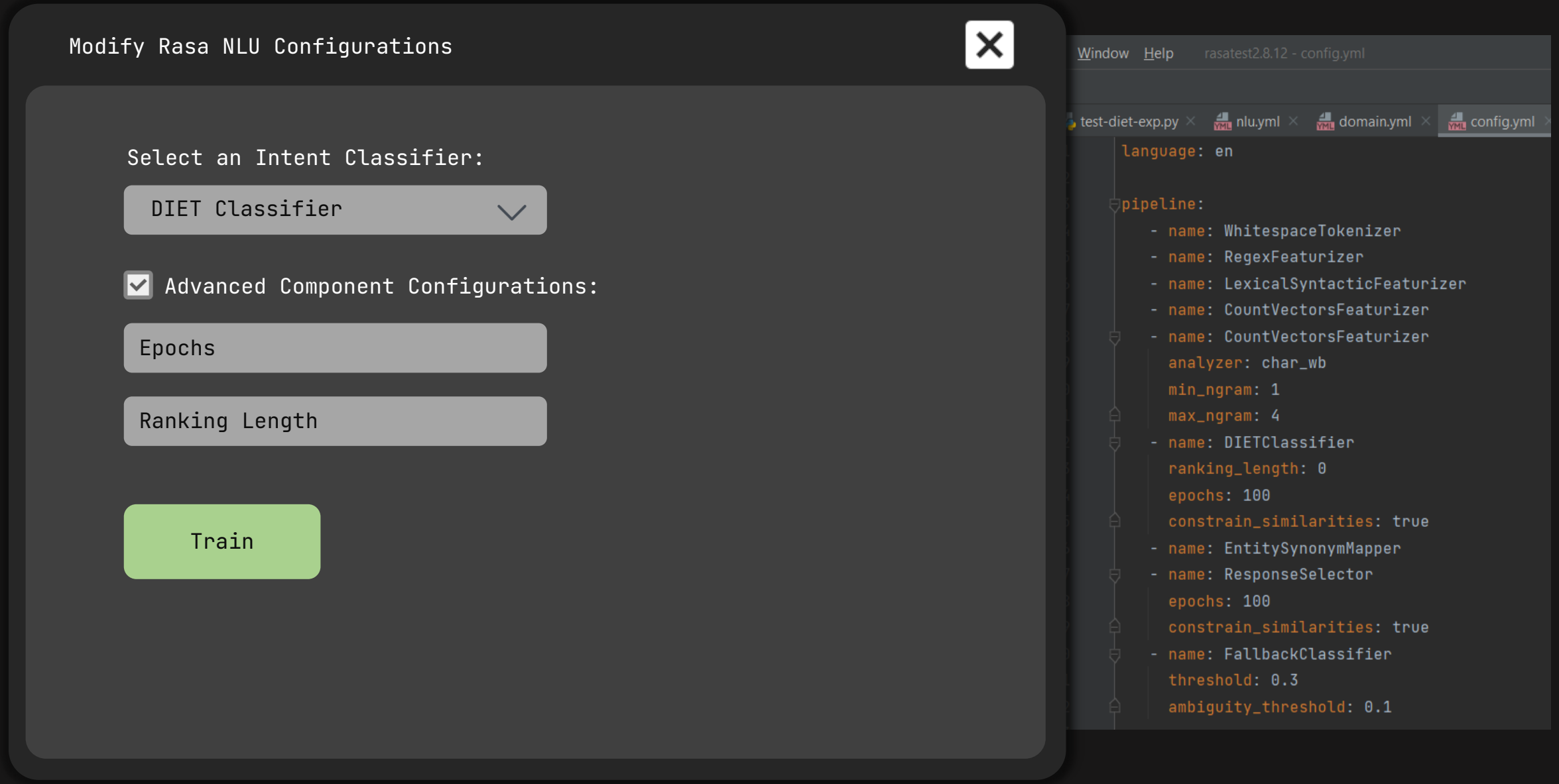
1. Build a Dataset for Rasa conversational AI Training
2. Train a Rasa Conversational AI with TensorBoard enabled
3. Feed the data point values to the LCE algorithm
4. ...e regularization techniques to smooth out the curve if the model has too many spikes
5. Identify overfittings or underfittings using the spikes in the curve
6. Provide evaluation to the user along with recommended epoch to use

Sample Image of a Model Overfitting



Source: https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323

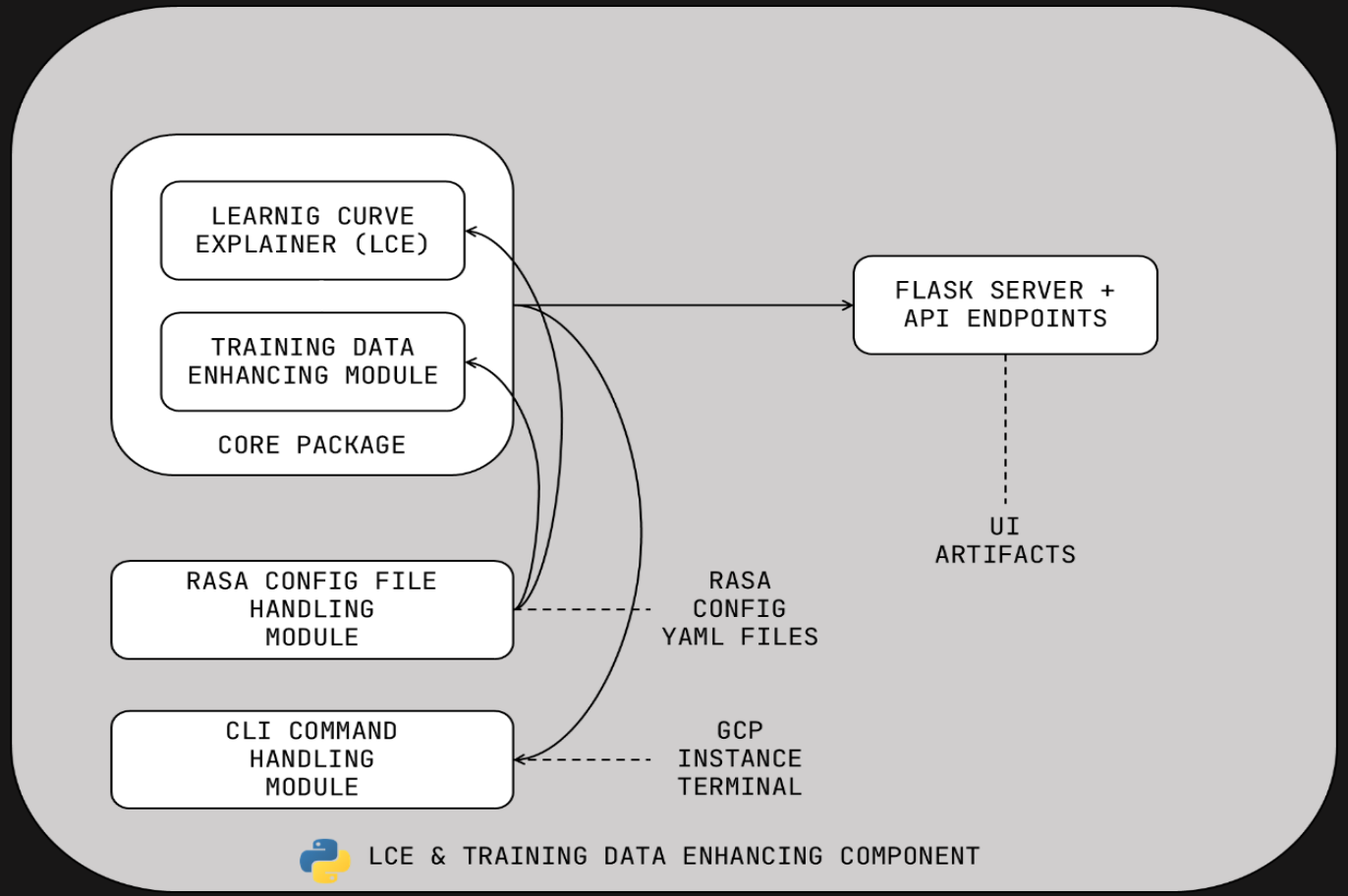
Data Improvement


Modify Rasa NLU Configurations
Select an Intent Classifier:
DIET Classifier
Advanced Component Configurations:
Epochs
Ranking Length
Train
Window Help
rasatest2.8.12 - config.yml
test-diet-exp.py
nlu.yml
domain.yml
config.yml
language: en
pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
analyzer: char_wb
min_ngram: 1
max_ngram: 4
- name: DIETClassifier
ranking_length: 0
epochs: 100
constrain_similarities: true
- name: EntitySynonymMapper
- name: ResponseSelector
epochs: 100
constrain_similarities: true
- name: FallbackClassifier
threshold: 0.3
ambiguity_threshold: 0.1

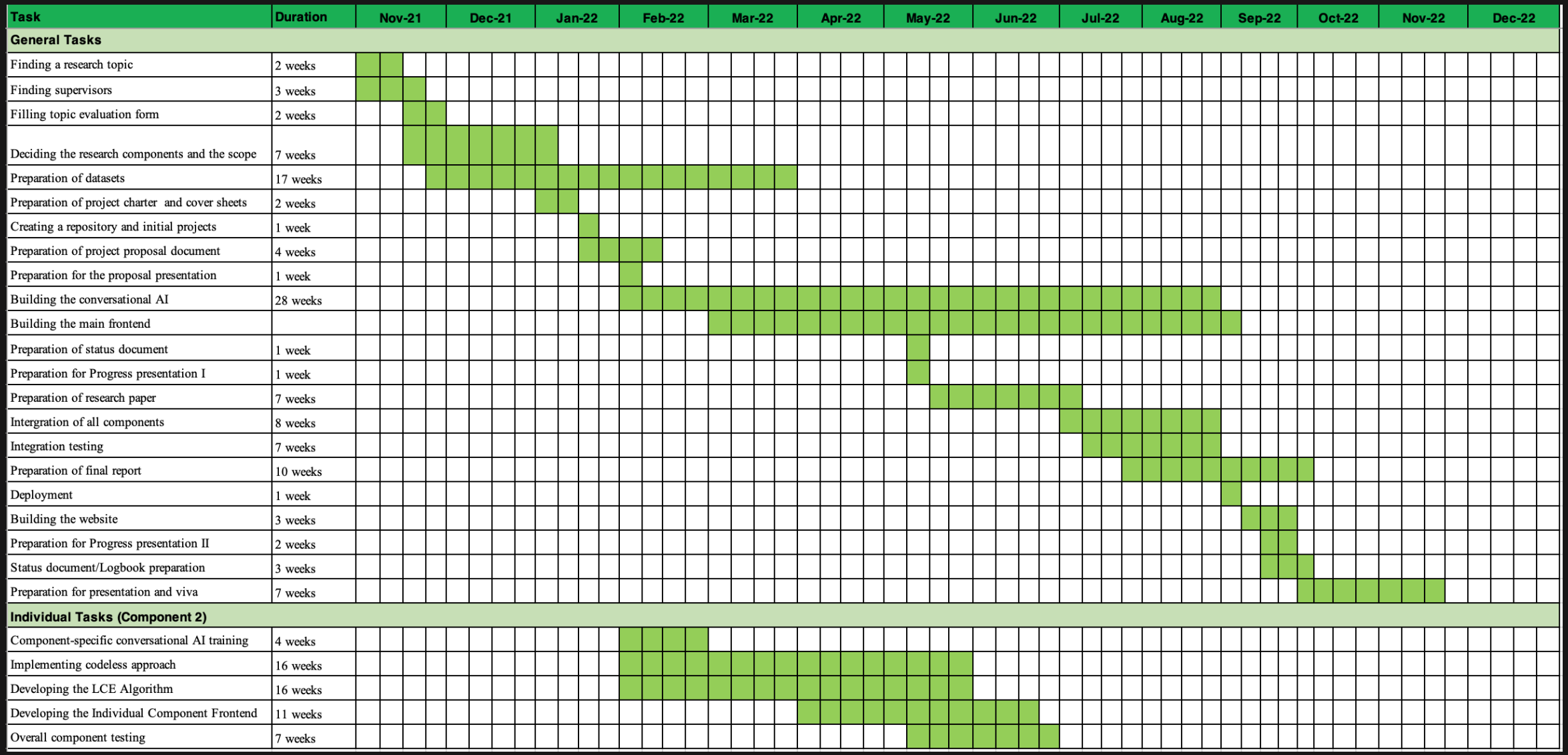
Work Breakdown Structure

1. Data Preparation
   - Data Collection
   - Data Augmentation
   - Data Deduplication
   - Data Preprocessing
2. Building Conversational AI
   - Add Training Data
   - Build NLU pipeline
   - Decide number of validation data points
   - Enable TensorBoard
   - Train a model
   - Testing NLU Model
3. Algorithm Development
   - Read TensorBoard Results
   - Develop Learning Curve Explainer algorithm to interpret TensorBoard Results
   - Build the LCE Package
   - Build a Locally executable Server
4. Implementation
   - Integrate LCE with Rasa
   - Build the Frontend
   - Implement code-less Maintenance
   - Build Docker & Docker-compose files
   - Build Caddy and NGINX Servers
   - Integrate component UI with the Main UI
5. Deployment
   - Update the GIT Repository
   - Set up GCP Instance
   - Deploy Docker Containers
   - Test LCE and Code-less Maintenance after Deployment

Individual Component Architecture



Individual Gantt Chart


Task
Duration
Nov-21
Dec-21
Jan-22
Feb-22
Mar-22
Apr-22
May-22
Jun-22
Jul-22
Aug-22
Sep-22
Oct-22
Nov-22
Dec-22
General Tasks
Finding a research topic
2 weeks
Finding supervisors
3 weeks
Filling topic evaluation form
2 weeks
Deciding the research components and the scope
7 weeks
Preparation of datasets
17 weeks
Preparation of project charter and cover sheets
2 weeks
Creating a repository and initial projects
1 week
Preparation of project proposal document
4 weeks
Preparation for the proposal presentation
1 week
Building the conversational AI
28 weeks
Building the main frontend
Preparation of status document
1 week
Preparation for Progress presentation I
1 week
Preparation of research paper
7 weeks
Intergration of all components
8 weeks
Integration testing
7 weeks
Preparation of final report
10 weeks
Deployment
1 week
Building the website
3 weeks
Preparation for Progress presentation II
2 weeks
Status document/Logbook preparation
3 weeks
Preparation for presentation and viva
7 weeks
Individual Tasks (Component 2)
Component-specific conversational AI training
4 weeks
Implementing codeless approach
16 weeks
Developing the LCE Algorithm
16 weeks
Developing the Individual Component Frontend
11 weeks
Overall component testing
7 weeks

References

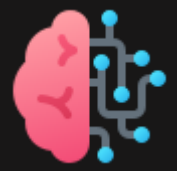
[1]. T. Bocklisch, J. Faulkner, N. Pawlowski, en A. Nichol, “Rasa: Open source language understanding and dialogue management”, *arXiv preprint arXiv:1712. 05181*, 2017.

[2]. “Introduction to rasa X,” *Open source conversational AI*, 10-Dec-2021. [Online]. Available: https://rasa.com/docs/rasa-x/. [Accessed: 22-Jan-2022].

SEETM: Sinhala-English Equivalent Token Mapper

Developing rule-based approaches to process code-mixed textural data and make word embeddings models lightweight using token mapping.

Jayasinghe D.T.
IT19075754
Data Science



Research Background

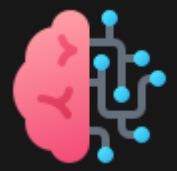
Bilingual and Multilingual Speakers

Code-Mixing

Code-Switching

Equivalent Words

Code-Mixing vs Code-Switching



Research Background

Bilingual and Multilingual Speakers

Code-Mixing

Code-Switching

Equivalent Words

Equivalent words Examples

Mother	අම්මා	அம்மா	დედა	ਮੰਮੀ	Мама

English	Sinhala	Tamil	Georgian	Punjabi	Russian

Research Questions

1

There are limited number of NLP researches on processing Sinhala-English code-switching text data

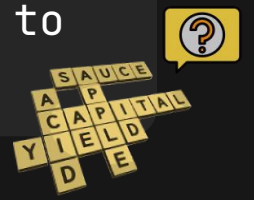
2

There are no Keyboard Interfaces in websites that can handle Sinhala English Code-switching text

3

Deep Learning NLP models need a lot of data.

It is hard to train Sinhala-English deep learning models due to low resources



Eg:
In Word embedding models trained for Sinhala-English code-switched text data, If training data has the word “Mother” but not “අම්මා” the model considers අම්මා as an Out-of-vocabulary token. But they have the same meaning.

Specific

Objective

Assigning the same word vector to equivalent words in a Sinhala-English code-switched text corpus

SEETM

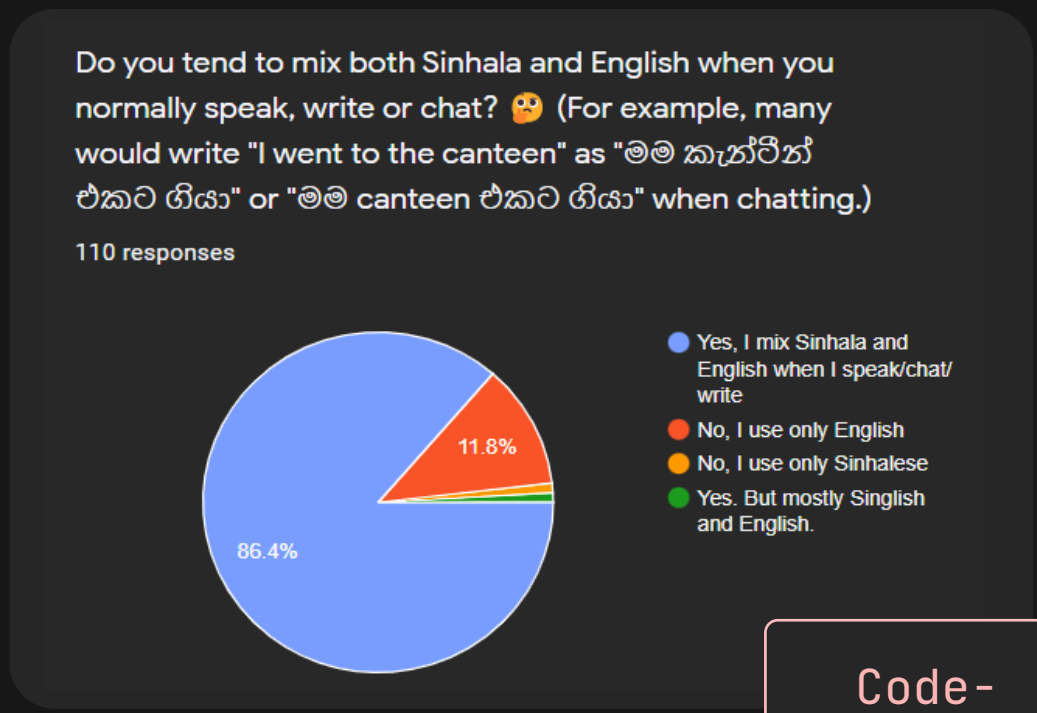
Sub Objectives

OOV words Handling in word embeddings

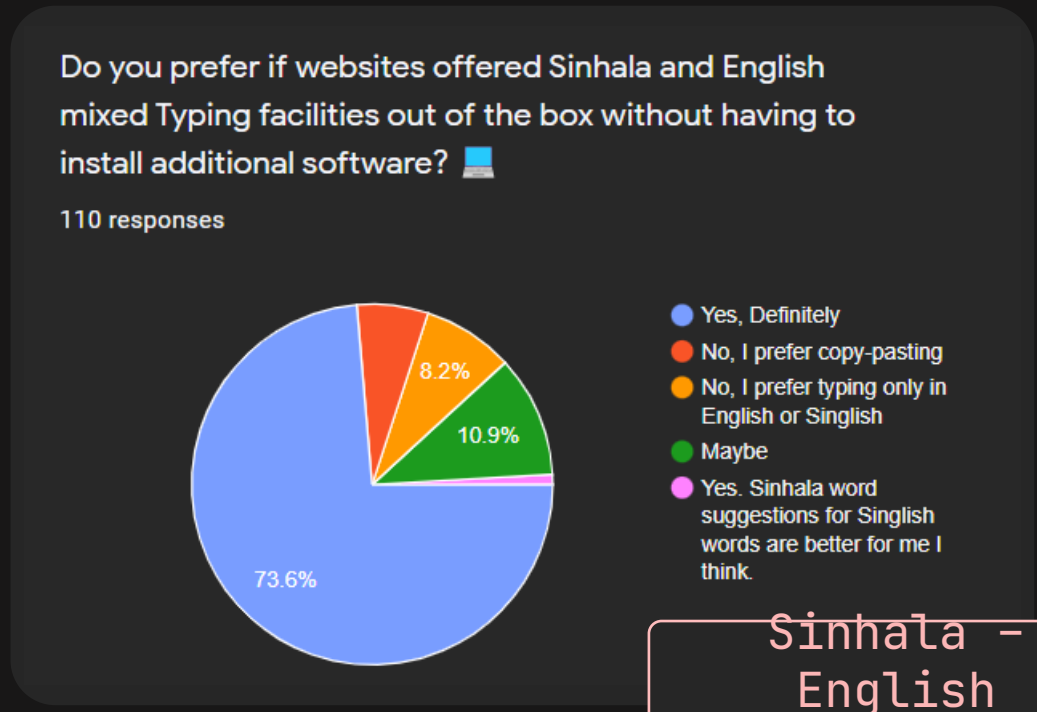
Bilingual Keyboard Interface

Lightweight Word Embedding Models

Survey Findings



Code-Switching



Sinhala - English Keyboard Interfaces

Research Gap

Research Paper Reference	Languages Used	Code-switched/ Code-mixed	Word Embeddings Model Type	Chatbot Framework	Special Data Pre-processing Techniques Proposed
[1] Paper	Sinhala	-	Word2Vec	-	Basic Text Preprocessing
[2] Paper	Sinhala (Main), English, Singlish	Code-Mixed Data	-	-	Dictionary Mapping (For Characters)
This Research Component	Sinhala (Main), English	Code-Switched Data	Word2Vec	Rasa	Equivalent Token Mapping And Character Mapping

Functional Requirements

1. SEETM should generate a single representation for equivalent words.

2. Mapped equivalent words should get a single vector representation.

3. SEETM should handle out-of-vocabulary words in Word2Vec models when at least one of the representations of equivalent tokens are present in training data.

4. Users should be able to type in both Sinhala and English in the User Interface.

Non-Functional Requirements

Efficiency

Reliability

Modularity

Scalability

Usability

Proposed Methodology

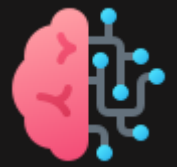
1. Build a Dataset for Token Mapping
2. Develop the SEETM Algorithm Using the Dataset
3. Train and Evaluation the SEETM Using Word2Vec Models
4. Developing the Frontend of the Component
5. Develop the Character Mapping For Keyboard Interface
6. Test and Integrate with the Overall System

Component Architecture

Training Data

ඔයාලගෙ මොනවද තියෙන ඩිග්‍රි?

Word Embeddings model

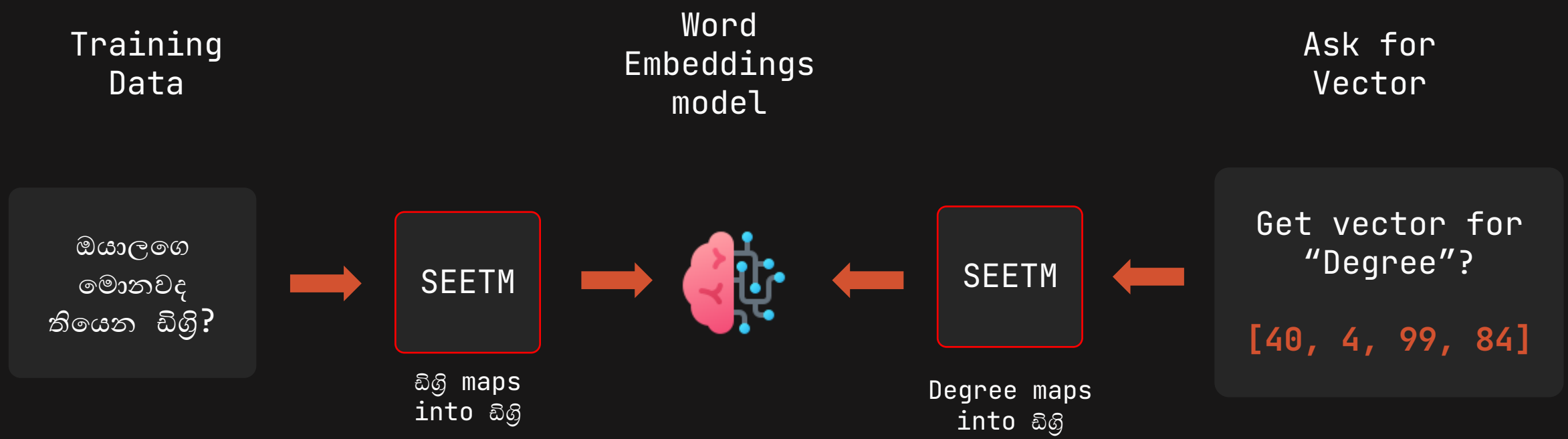


Ask for Vector

Get vector for “Degree”?

OOV

Component Architecture



* Here, SEETM maps English Words to Sinhala Equivalent

Work Breakdown Structure

SEETM

1 Data Preparation

- Data Collection
- Data Augmentation
- Data Deduplication
- Data Preprocessing

2 SEETM Algorithm Development

- Develop Sinhala-English Character Map
- Develop Equivalent Token Mapping Algorithm: SEETM
- Evaluate Token Mapping Manually
- Training a Word2Vec Model
- Evaluating Token Mapping Effect on Word2Vec Model

3 Sinhala-English Code-Switching Keyboard Interface

- Develop the Keyboard Interface Using Keystroke Mapping
- Utilizing the Character Map within the Keyboard Interface
- Enable Efficient Language Switching

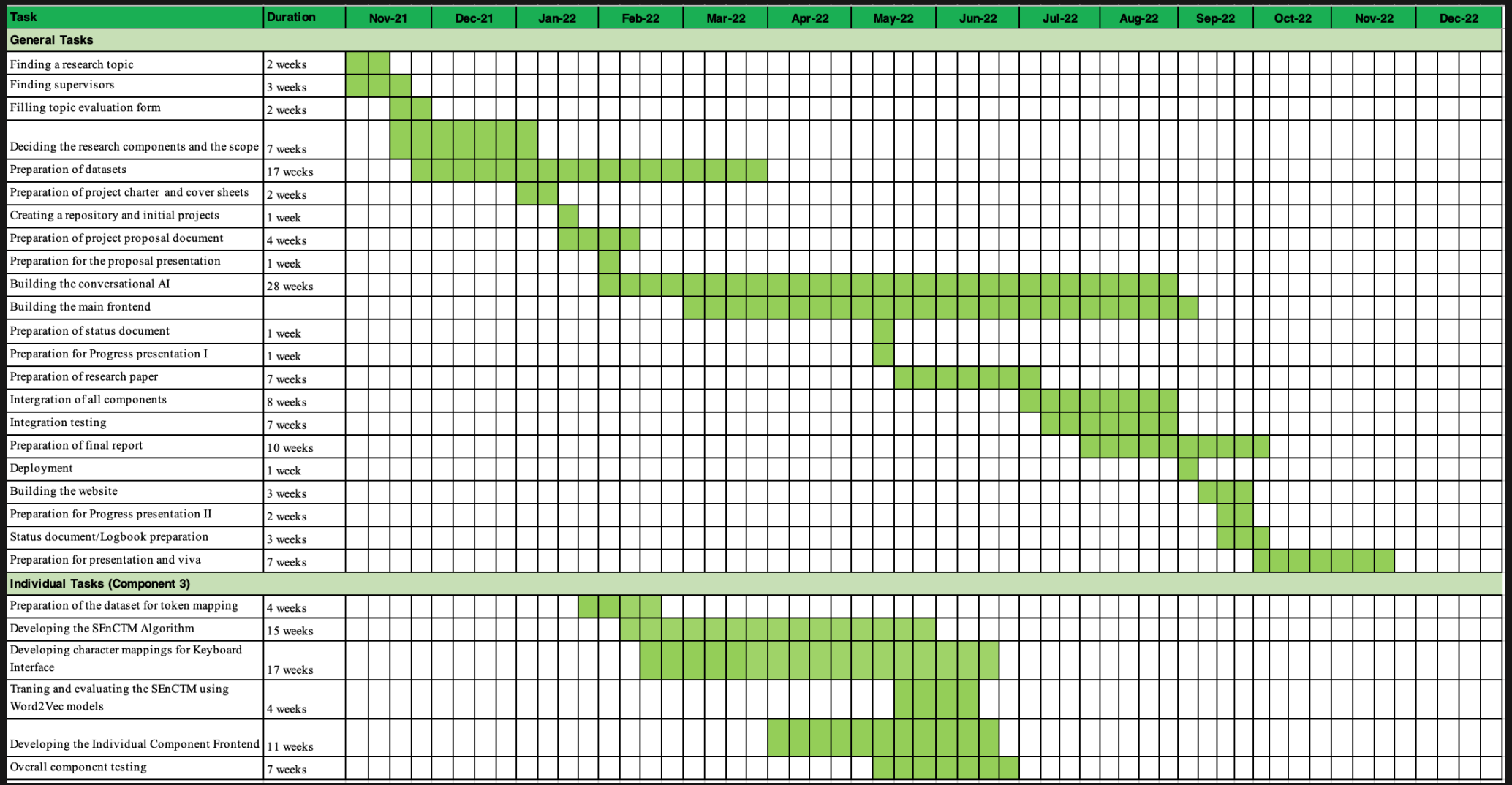
4 Implementation

- Integrate SEETM with Rasa
- Build Docker Images
- Write Docker Compose Configs
- Build Caddy and NGINX Servers
- Integrate component UI with the Main UI

5 Deployment

- Update the GIT Repository
- Set up GCP Instance
- Deploy Docker Containers
- Test SEETM and Keyboard Interface after Deployment

Gantt Chart



Task	Duration
General Tasks	
Finding a research topic	2 weeks
Finding supervisors	3 weeks
Filling topic evaluation form	2 weeks
Deciding the research components and the scope	7 weeks
Preparation of datasets	17 weeks
Preparation of project charter and cover sheets	2 weeks
Creating a repository and initial projects	1 week
Preparation of project proposal document	4 weeks
Preparation for the proposal presentation	1 week
Building the conversational AI	28 weeks
Building the main frontend	
Preparation of status document	1 week
Preparation for Progress presentation I	1 week
Preparation of research paper	7 weeks
Intergration of all components	8 weeks
Integration testing	7 weeks
Preparation of final report	10 weeks
Deployment	1 week
Building the website	3 weeks
Preparation for Progress presentation II	2 weeks
Status document/Logbook preparation	3 weeks
Preparation for presentation and viva	7 weeks
Individual Tasks (Component 3)	
Preparation of the dataset for token mapping	4 weeks
Developing the SEnCTM Algorithm	15 weeks
Developing character mappings for Keyboard Interface	17 weeks
Traning and evaluating the SEnCTM using Word2Vec models	4 weeks
Developing the Individual Component Frontend	11 weeks
Overall component testing	7 weeks

References

[1]. T. KasthuriArachchi and E. Y. A. Charles, "Deep Learning Approach to Detect Plagiarism in Sinhala Text," *2019 14th Conference on Industrial and Information Systems (ICIIS)*, 2019, pp. 314-319, doi: 10.1109/ICIIS47346.2019.9063299.

[2]. A. Kugathasan and S. Sumathipala, "Standardizing Sinhala Code-Mixed Text using Dictionary based Approach," *2020 International Conference on Image Processing and Robotics (ICIP)*, 2020, pp. 1-6, doi: 10.1109/ICIP48927.2020.9367353.

SIENA: Annotating entities using reverse-stemming & other techniques

to develop a data annotation tool for code-mixed text data for efficient custom entity tagging.

Sakalasooriya S.A.H.A.
IT19051208
Data Science

Research Background

Named Entity Tagging
(places, company names, countries)

Need for Custom Entities
(domain specific name entities)

Named Entity Recognition

Faster ways to Annotate Entities
(Regular expression, Fuzzy matching)

Entities in Different Languages

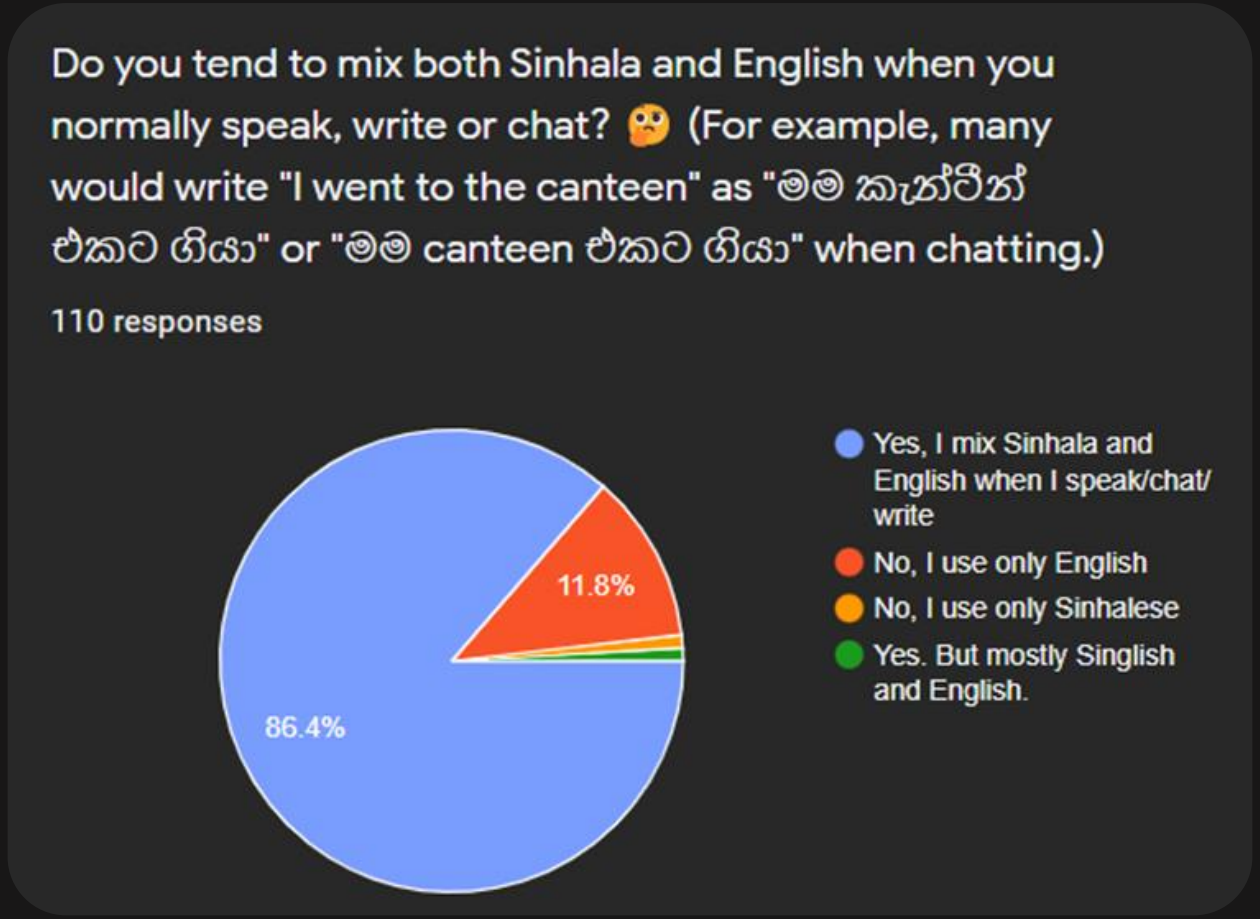
Research Gap

Tool name	Collaboration features	Name Entity recommendations	Focused on Sinhala – English mixed text
ANEA	no	yes (via external knowledge source)	no
brat	no	yes (via semantic class disambiguation)	no
GATE	yes	no	no
YEDDA	no	yes (via maximum matching algorithm)	no
SIENA (This research component)	no	yes	yes

SIENA can identify small variations in words

- පුටුව
- පුටුවක්
- පුටුවේ
- පුටුවක

Survey Findings



Research

Question

Why custom name entity tagging is very time consuming?

💤🤗

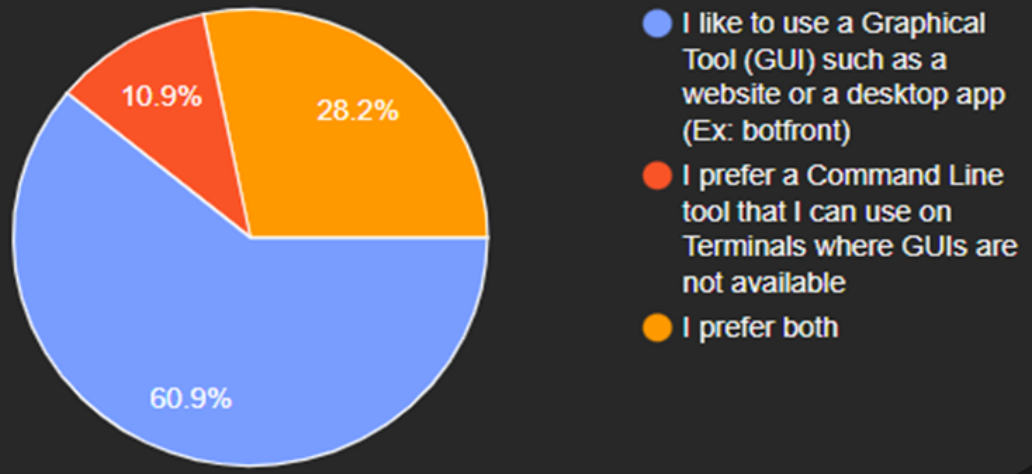
Any solution?

🤔

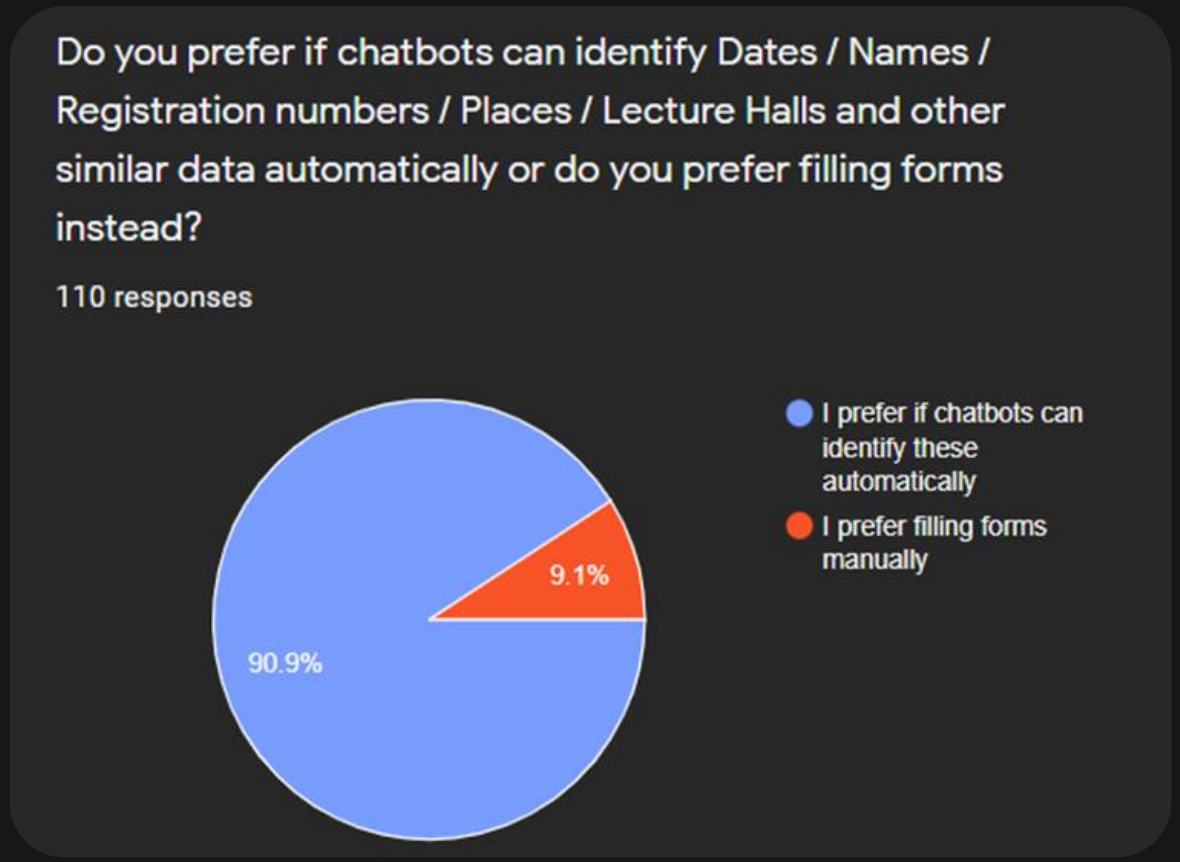
Survey Findings

Say there is a tool to annotate data when preparing them to train a ML model. What kind of a tool do you prefer out of the given options? 🤔 (Data annotating can be something like for each data point, identify the correct class, or for each sentence, identify names and tag the position)

110 responses



Survey Findings



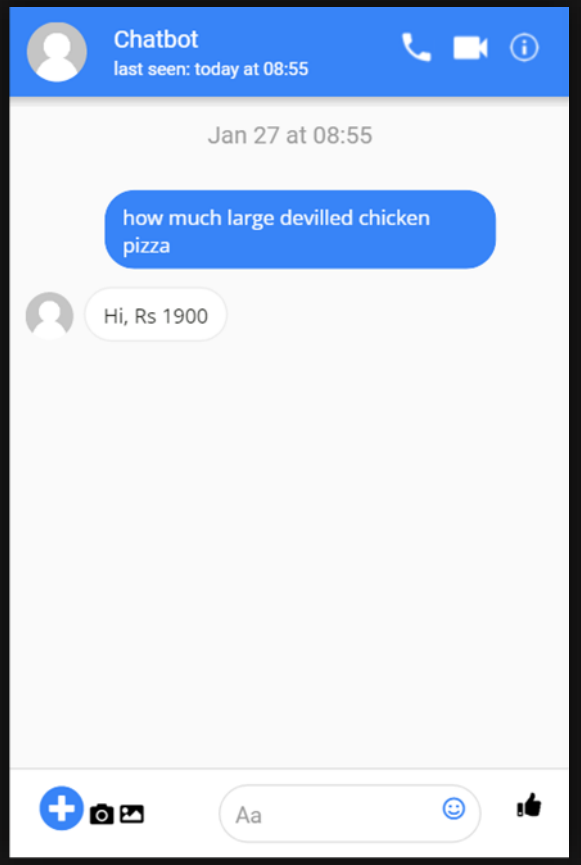


Figure 1

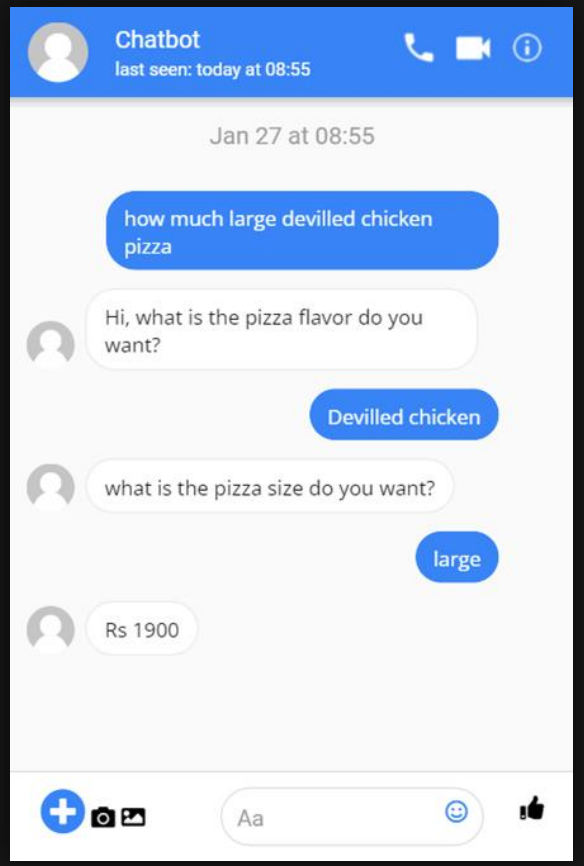


Figure 2

Specific Objectives

Increase the efficiency of the text annotation process in a Sinhala-English code-switching corpus by providing accurate name entity recommendations.

Course type

IT ඩිග්‍රි එකක් කරන්න නම් A/L pass වෙන්න ඕනේ } Annotated by user

SIENA Recommendation : Course type

IT ඩිග්‍රියක් කරන්න නම් A/L pass වෙන්න ඕනේ

Sub

Objectives

Define the Recommendation hierarchy

Make SIENA Compatible with Frameworks

Make Knowledge base as Moduler Component

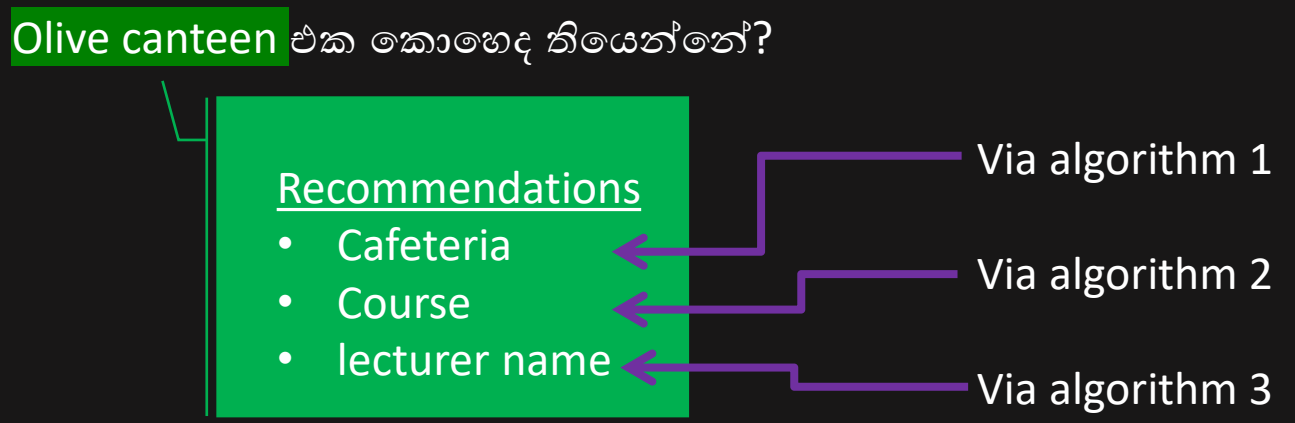
Develop Visualizations Technique to Provide User Friendly Suggestions

Sub

Objectives

Define the recommendation hierarchy

Order recommendations according to the algorithm evaluation results



Sub

Objectives

Make SIENA compatible with frameworks

Annotated corpus need to be used in NER libraries to build NER models

spaCy

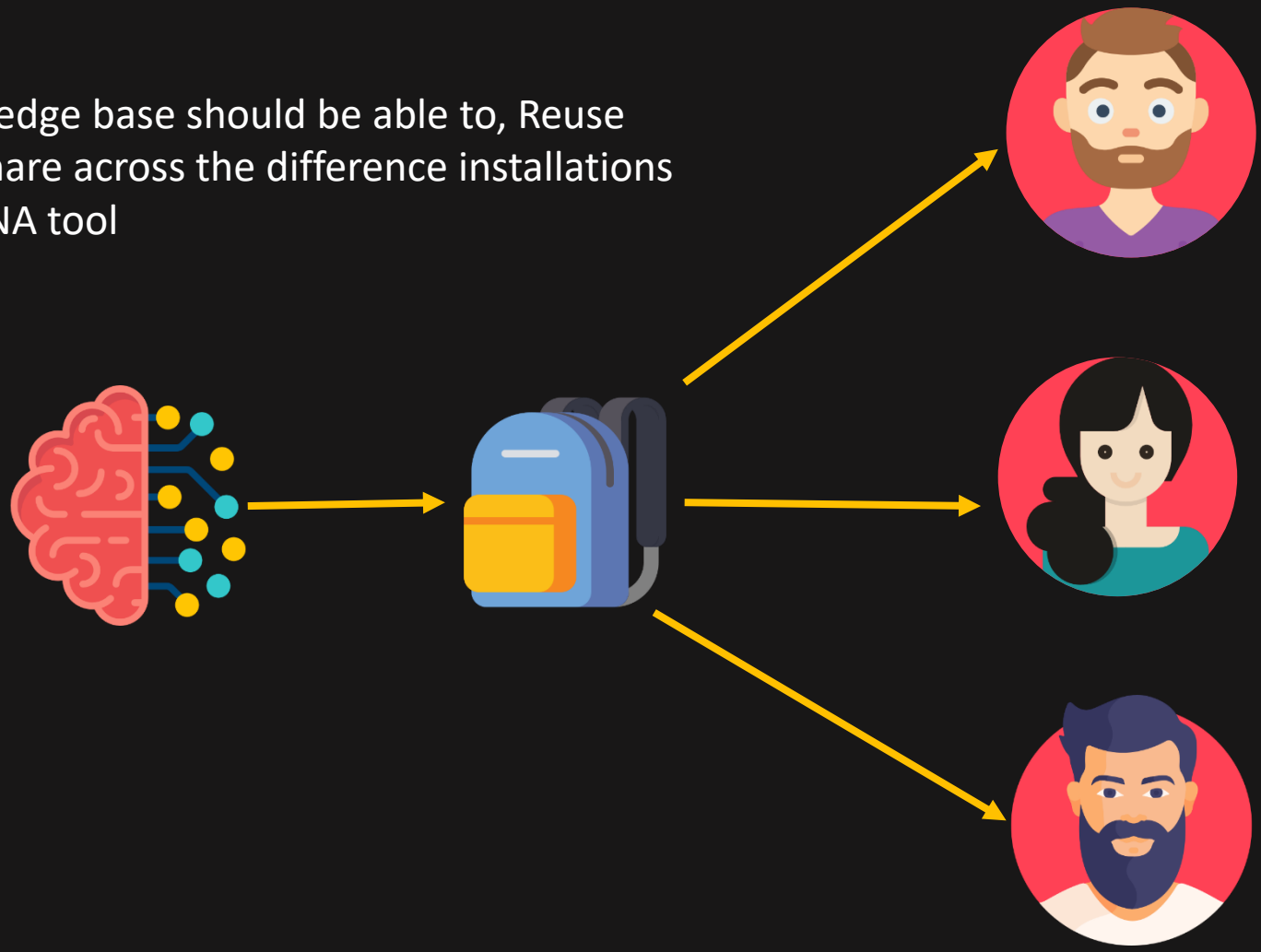
spaCy v3.0 requires predefined binary format

Sub

Objectives

Make knowledge base as moduler component

Knowledge base should be able to, Reuse and Share across the difference installations of SIENA tool



Sub

Objectives

Develop visualizations technique to provide user friendly suggestions

SIENA recommendations should not be a distraction to the user

User interface should be clean and elegant

Dark mode should be supported to reduce eye strain

Functional Requirements

User should be able to find recommended name entities

User should be able to import / upload corpus into SINEA

User should be able to export annotated text from SIENA

User should be able to import / upload portable knowledge base into SINEA

User should be able to export portable knowledge base from SINEA

Non-functional Requirements

SIENA should be able to handle large text corpus

SIENA should be able to easily maintain

SIENA should be able to easily install on user’s computer

SIENA should be reliable

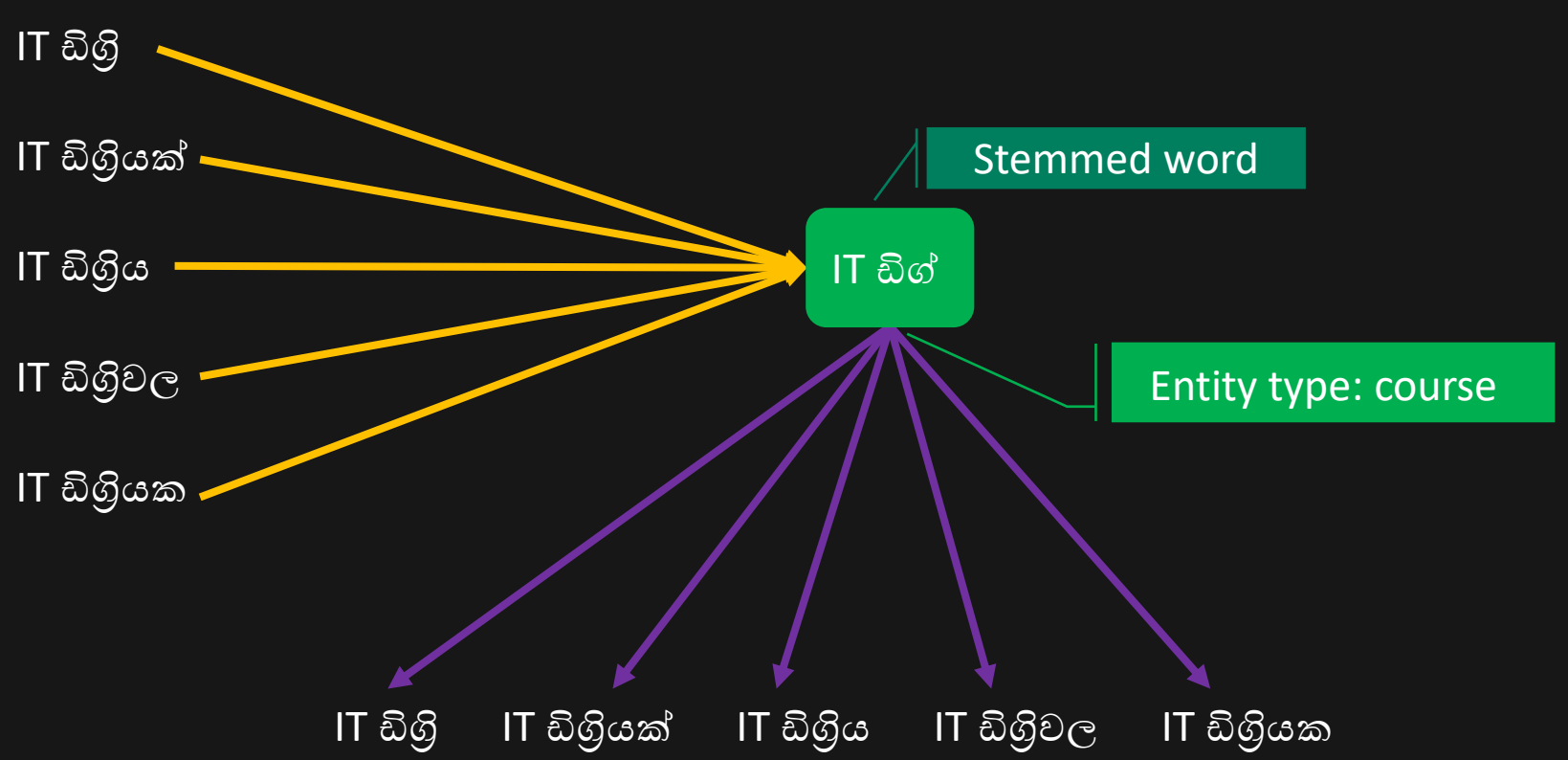
SIENA should be secure

Methodology

1. Revere stemming approach development
2. Word-wise cosine similarity approach development
3. N-gram approach development

Methodology

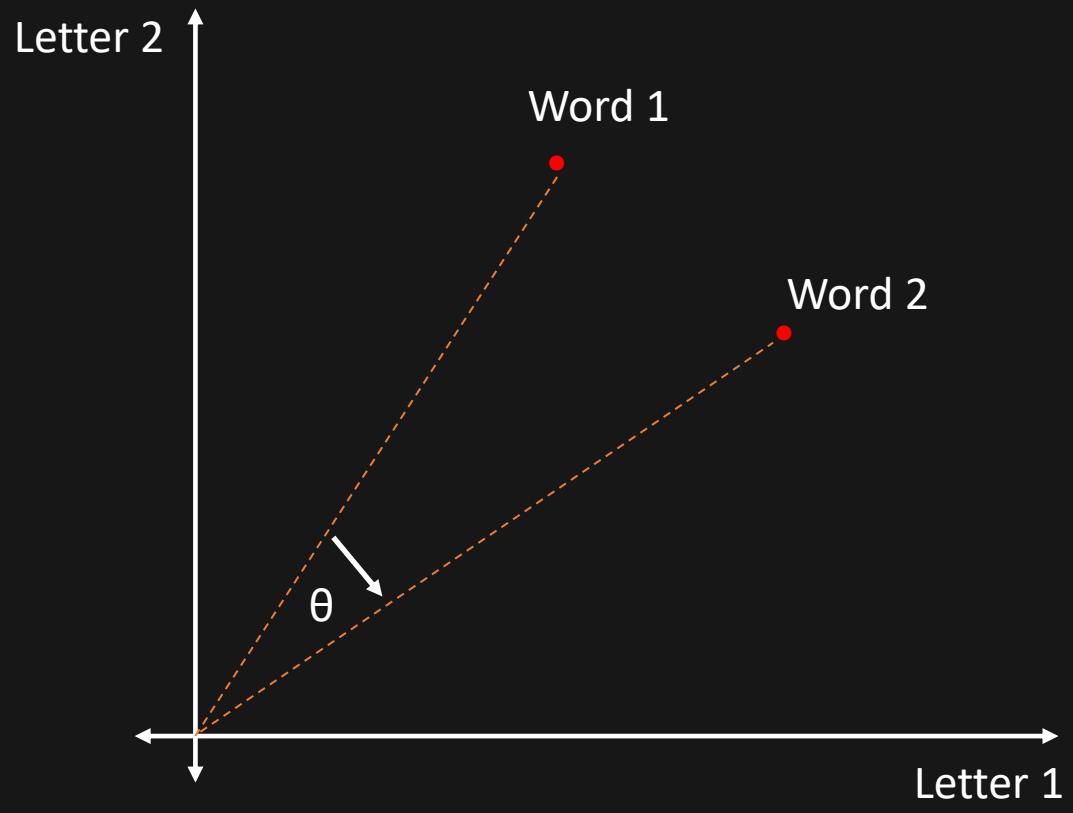
Revere stemming approach



All the Sinhala suffixes are referred by [5] “Basaka mahima – J.B Dissanayake”

Methodology

Word-wise cosine similarity



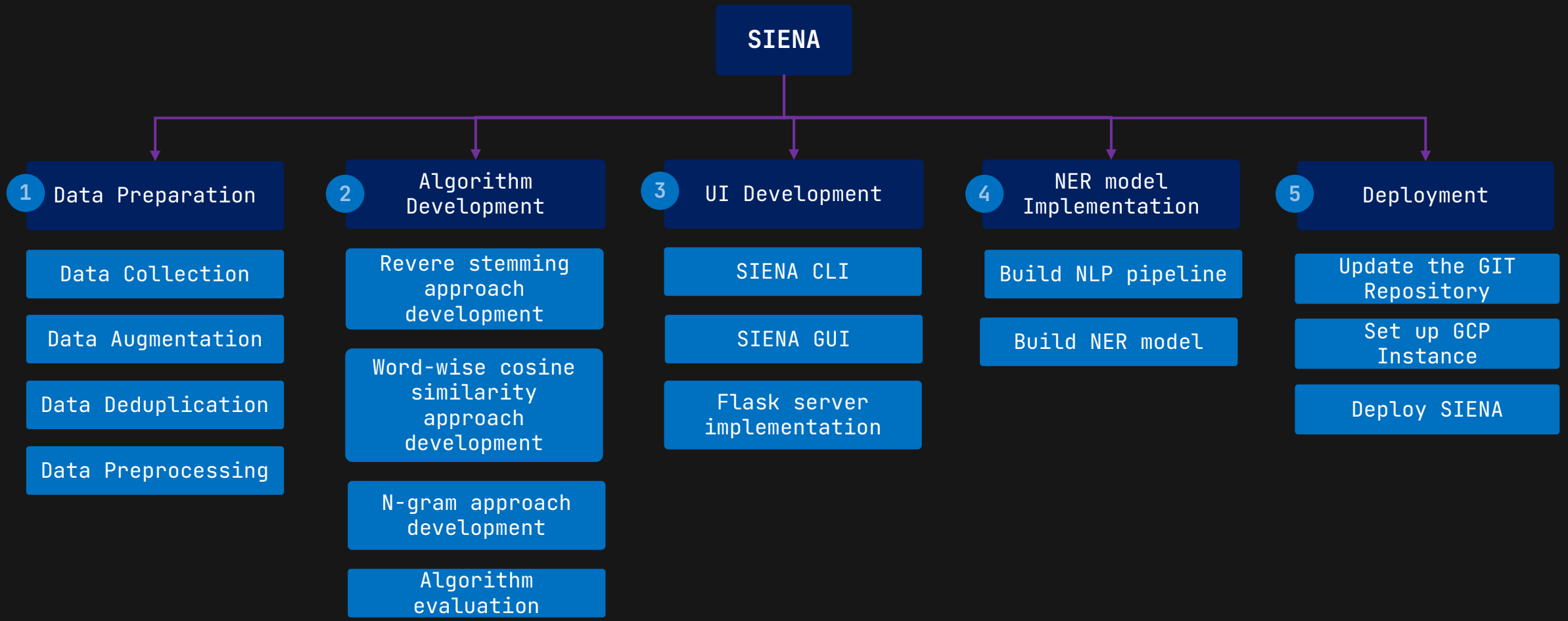
Methodology

N-gram approach

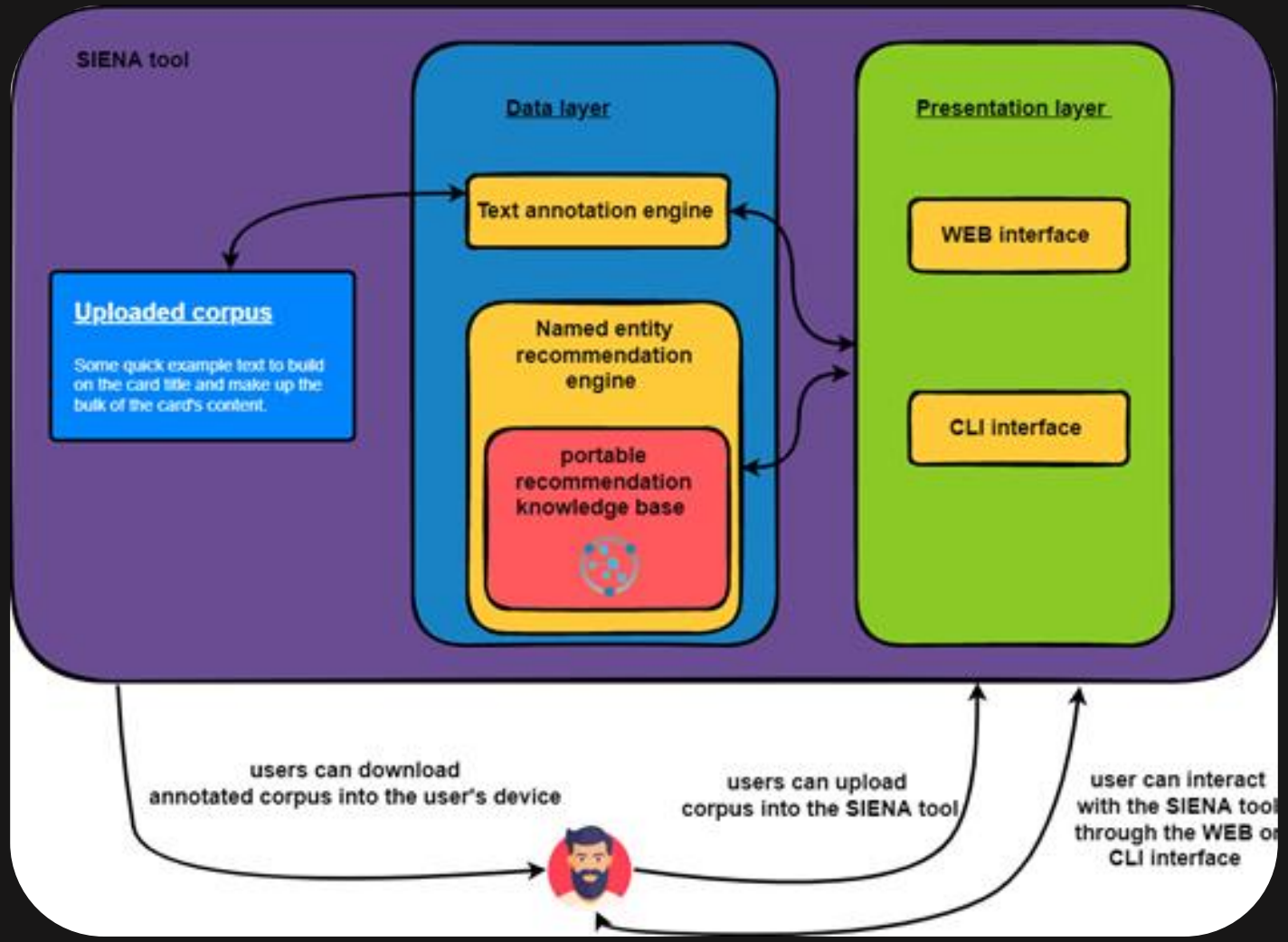
Letters of “IT ඩිග්‍රියක්” can three gram into
“IT “
“T ඩ”
“ඩිග්‍ර”
“ඩිග්‍රිය”
“ග්‍රියක්”

When comparing two words we can count how many n-grams are matching

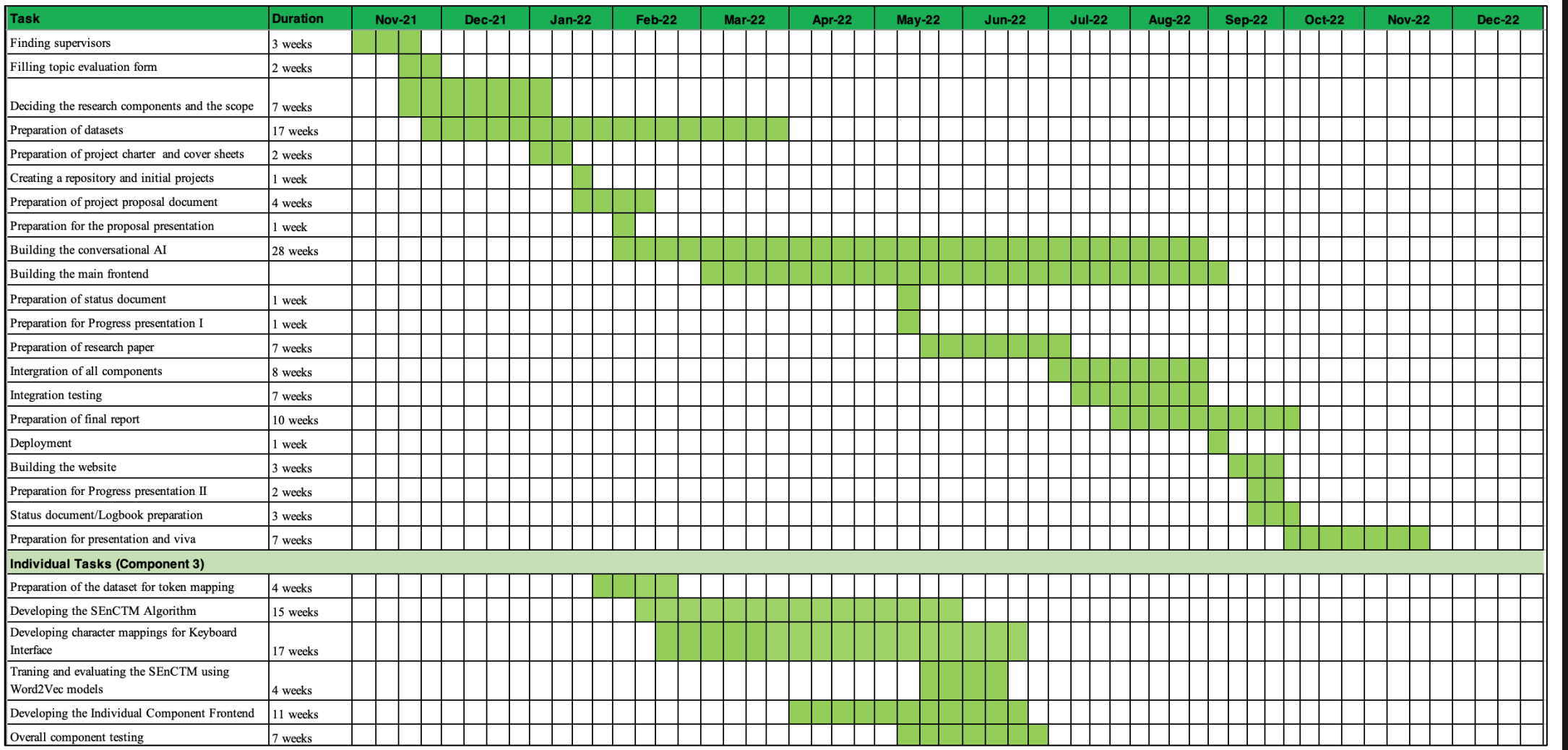
Work Breakdown Structure



Individual Component
Architecture



Gantt chart



Task	Duration	Nov-21	Dec-21	Jan-22	Feb-22	Mar-22	Apr-22	May-22	Jun-22	Jul-22	Aug-22	Sep-22	Oct-22	Nov-22	Dec-22
Finding supervisors	3 weeks														
Filling topic evaluation form	2 weeks														
Deciding the research components and the scope	7 weeks														
Preparation of datasets	17 weeks														
Preparation of project charter and cover sheets	2 weeks														
Creating a repository and initial projects	1 week														
Preparation of project proposal document	4 weeks														
Preparation for the proposal presentation	1 week														
Building the conversational AI	28 weeks														
Building the main frontend															
Preparation of status document	1 week														
Preparation for Progress presentation I	1 week														
Preparation of research paper	7 weeks														
Intergration of all components	8 weeks														
Integration testing	7 weeks														
Preparation of final report	10 weeks														
Deployment	1 week														
Building the website	3 weeks														
Preparation for Progress presentation II	2 weeks														
Status document/Logbook preparation	3 weeks														
Preparation for presentation and viva	7 weeks														
Individual Tasks (Component 3)															
Preparation of the dataset for token mapping	4 weeks														
Developing the SEnCTM Algorithm	15 weeks														
Developing character mappings for Keyboard Interface	17 weeks														
Traning and evaluating the SEnCTM using Word2Vec models	4 weeks														
Developing the Individual Component Frontend	11 weeks														
Overall component testing	7 weeks														

References

[1] Anastasia Zhukova, Felix Hamborg, Bela Gipp, 'ANEA: Automated (Named) Entity Annotation for German Domain-Specific Texts' Available: https://arxiv.org/pdf/2112.06724.pdf

[2] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii, 'BRAT: a Web-based Tool for NLP-Assisted Text Annotation' Available: https://aclanthology.org/E12-2021.pdf

[3] Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, Genevieve Gorrell, 'GATE Teamware: a web-based, collaborative text annotation framework', Available: https://www.jstor.org/stable/42636386

[4] Jie Yang, Yue Zhang, Linwei Li, Xingxuan Li, 'YEDDA: A Lightweight Collaborative Text Span Annotation Tool', Available: https://aclanthology.org/P18-4006.pdf

[5]J.B Dissanayake, Basaka mahima, ISBN: 9789556963656

[6] "Spacy Styleguide",
https://spacy.io/styleguide

[7] "Spacy Data formats · spaCy API Documentation",
https://spacy.io/api/data-formats

[8] "Vector Icons and Stickers - PNG, SVG, EPS, PSD and CSS",
https://www.flaticon.com/

Overall Commercialization Plan

DEMO PACKAGE

Free for 1 Month

10 Intents/Question Categories
2 API Integrations
Bot Analytics Included
Unlimited CDD Improvements

ON PREM PACKAGE

$ 199.99/One Time

2 Free Maintenance.
($9.99 per additional call)
Bot Analytics + CDD

CaaS PACKAGES

STARTER

$ 9.99/Month

20 Intents/Question Categories
2 API Integrations
No Bot Analytics
Unlimited CDD Improvements

PRO

$ 34.99/Month

180 Intents
110 API Integrations
Bot Analytics Included
CDD + Sinhala Entity Annotating

GENIUS

$ 49.99/Month

400 Intents
200 API Integrations
Bot Analytics Included
CDD + ML Packages

Budget Plan

Component Name	Individual Item Price (LKR)	Number of Items	Duration	Total Item Price (LKR)
Domain Name	2148.43/year	1	1 year	2148.43
GCP Instance	10683.84/month	1	6 months	64,103.06
Reference Book: Basaka Mahima by J.B. Dissanayake	1250.00	1	-	1250.00
Research Paper Publication	25000.00	-	-	25000.00
Grand Total	-	-	-	92,501.49

Thank You

Any Questions