End-to-End Natural Language Understanding Pipeline for Bangla Conversational Agents

Fahim Shahriar Khan[1] , Mueeze Al Mushabbir[1] , Mohammad Sabik Irbaz[2] , MD Abdullah Al Nasim[2]

Department of Computer Science and Engineering, Islamic University of Technology[1], Machine Learning Team, Pioneer Alpha Ltd.[2]
khanfahimshahriar0@gmail.com, almushabbir@iut-dhaka.edu, sabikirbaz@iut-dhaka.edu, nasim.abdullah@ieee.org

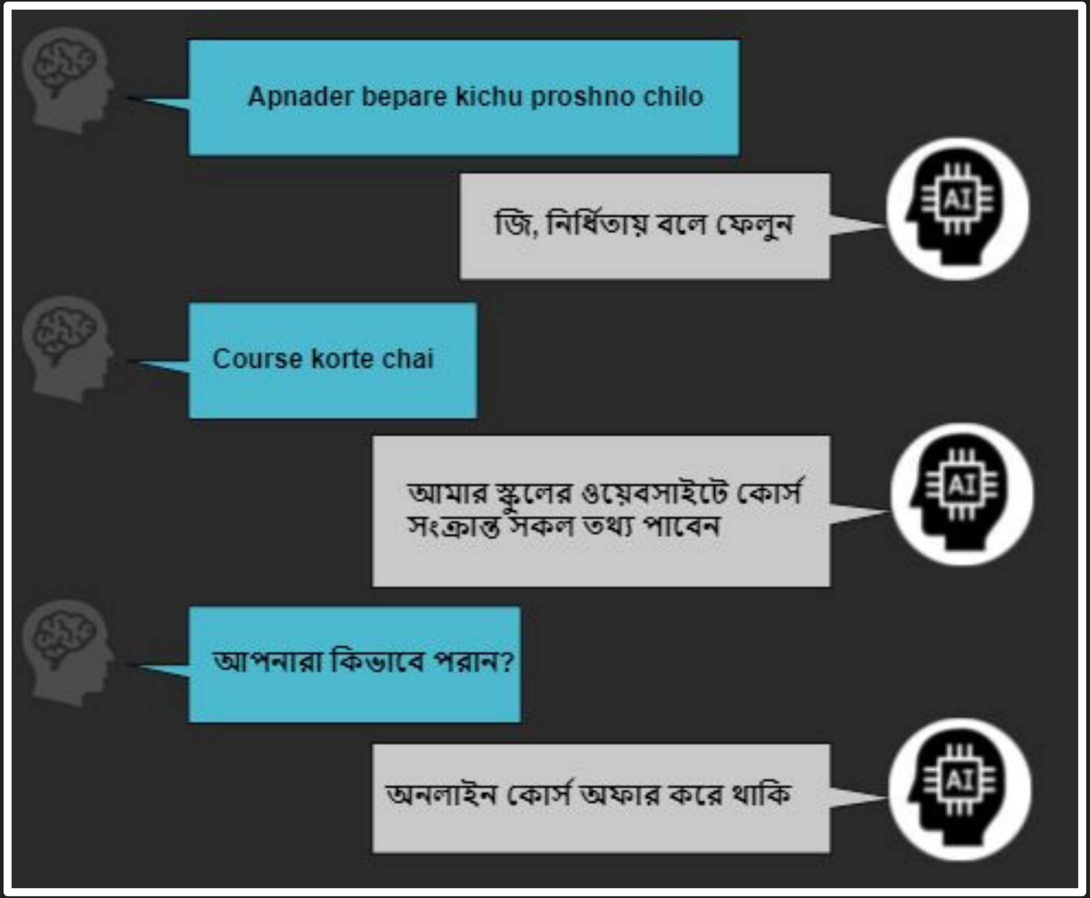
Paper ID : 341

20th IEEE International Conference on Machine Learning and Applications, 2021

Introduction

- Conversational AI Agents aim to provide virtual assistant like services in the form of a dialog system using natural languages.
- Existing chatbot systems usually do not have enough support for low-resource languages, like Bangla.
- We aim to build an end-to-end (from creating a dataset to evaluating components and pipelines) Bangla Chatbot - to be used as a virtual assistant in the business environments.

Example



Apnader bepare kichu proshno chilo
জি, নির্ধিতায় বলে ফেলুন
Course korte chai
আমার স্কুলের ওয়েবসাইটে কোর্স সংক্রান্ত সকল তথ্য পাবেন
আপনারা কিভাবে পরান?
অনলাইন কোর্স অফার করে থাকি
AI

Motivation

Support for existing dialog systems in low-resources languages like Bangla, or Bangla Transliteration is scarce.

With the increasing need of customer services in the virtual world, businesses look for automated service providing solutions.

Comparative analysis among various components and pipelines with proper reasoning are difficult to figure out.

Problem Statement

Using low-resource, scarce and skewed Bangla dataset for intent recognition and entity extraction, we need to build an end-to-end NLU pipeline for chatbots which can receive messages and send responses seamlessly as a Business Assistant.

Research Challenges

1 Skewed Low-Resource Data

Available datasets for languages like Bangla are low-resource, low-quality skewed datasets

2 Dealing with Bangla Transliteration in English

[describe transliteration]

3 Technical Analysis of Components and Pipelines

Understanding the technical properties of each component and pipeline, and their comparative performances with proper reasonings are difficult to figure out.

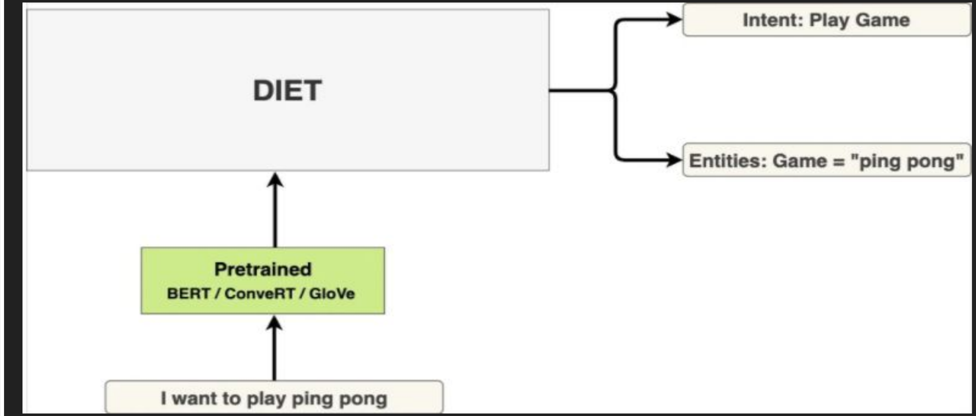
Related works

RASA

Related works

DIET Classifier

- Dual Intent Entity Transformer (DIET)
- Multi-task transformer architecture
- used for both *Intent Classification* and *Entity Extraction*
- Provides Modularity
  - used with various pre-trained embeddings like BERT, GloVe, etc.
- Comparable Performance against large-scale pre-trained language models
  - Faster and Better than BERT

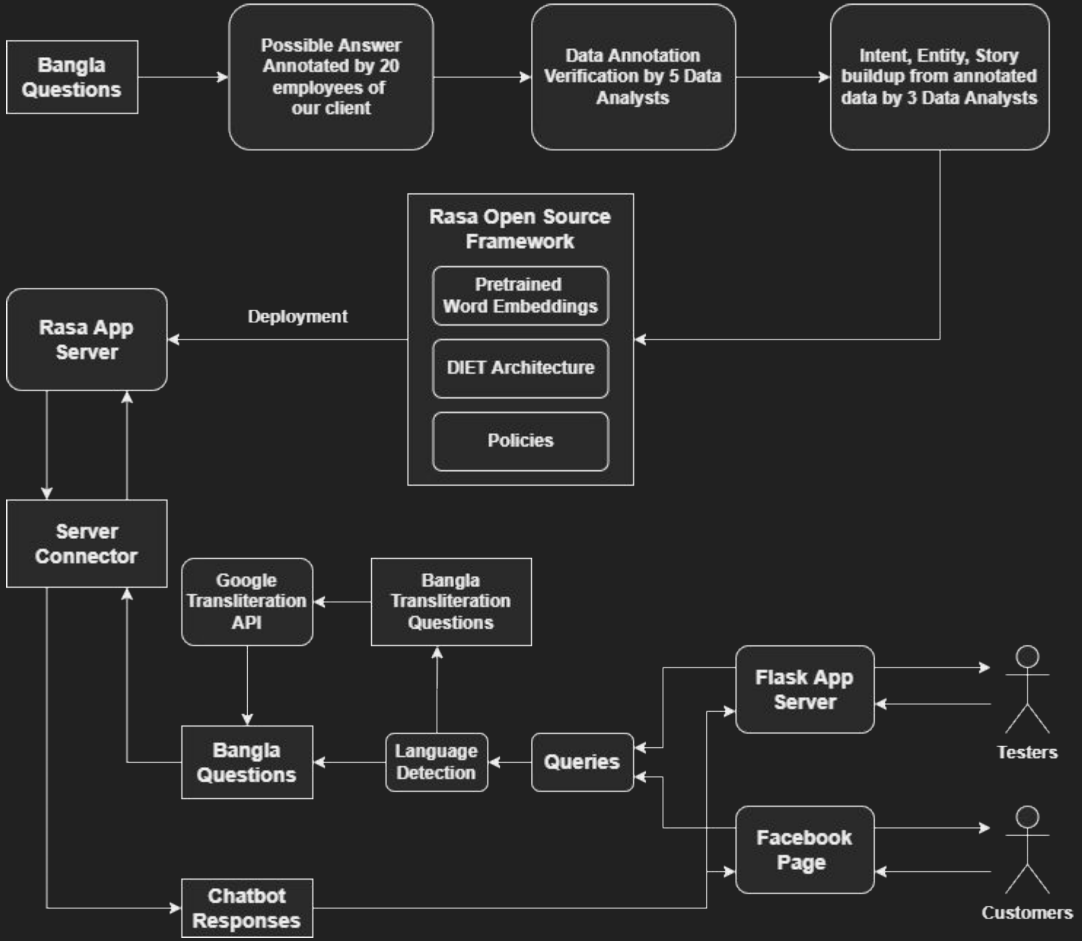


https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/

Related works

[Vietnamese Paper]

Machine Learning Life Cycle



Dataset Preparation

Dataset overview

- Rasa open source architecture expects the dataset to be partitioned into a 3 separate yml files.
  - nlu.yml
  - domain.yml
  - stories.yml

- Our chatbot needs to deal with FAQs in Bangla and Bangla Transliterations in English.

- FAQs were collected from our client’s interaction history with customers from different social media platforms.

- They were annotated by 20 employees of our client.

Parts of the Dataset

1. *nlu.yml*
    - The FAQs gathered from our client is labelled into intents and entities.
    - In our custom dataset there are 45 intents and 9 entities.
    - Intents are classes to which each FAQ belongs to and entities are subjects in them.
    - There are a total of 250+ samples.
    - This file is used to train the NLU module which is responsible for intent classification and entity extraction.

```yaml
- intent: type_of_services
  examples: |
    - আপনারা কী টাইপ [সার্ভিস](entity_service) দিয়ে থাকেন
      ?
    - আপনাদের কি কি [প্রোডাক্ট ](entity_service) আছে?
    - আপনাদের কি কি [সার্ভিস](entity_service) আছে?
    - আপনারা কি [সার্ভিস](entity_service) নিয়ে কাজ করেন?
    - আমাকে আপনাদের সেবা(entity_service) সম্পর্কে বলুন।
    - আপনাদের দেওয়া [সেবা](entity_service) সম্পর্কে জানতে
      চাই?

- intent: type_of_work
  examples: |
    - ভাই, আপনাদের কাজ কি?
    - আপনারা কি নিয়ে কাজ করছেন?
    - আপনাদের প্ল্যাটফর্ম থেকে কি সুবিধা পেতে পারি?
    - আপনারা কি কি জিনিস নিয়ে কাজ করেন?
    - কি কি জিনিস নিয়ে কাজ করেন আপনারা?
    - ভাই, আপনাদের প্লাটফর্মে কি কি কাজ হয়?
    - কোন কোন বিষয়ে আপনারা সাহায্য করতে পারেন?
    - আপনাদের প্লাটফর্মে কি কি কাজ হয় জানতে চাই

- intent: need_help
  examples: |
    - আপনাদের সম্পর্কে কিছু  প্রশ্ন ছিল?
    - আমাকে একটু সাহায্য করবেন?
    - একটা  প্রশ্ন করি?
    - আমার একটি হেল্প দরকার
    - আমার কয়েকটি প্রশ্ন আছে
    - আমাকে একটি সাহায্য করা যাবে কি??
    - আপনাদের প্রোডাক্ট ব্যবহারে সাহায্যের প্রয়োজন।
```

Parts of the Dataset(cont.)

2. *domain.yml*
    - This file contains all the corresponding responses of each FAQ.
    - The responses are classified into 117 different response types.
    - There are over 150+ responses arranged in our custom dataset.
    - This file also requires all the intents and entities in the nlu.yml file grouped together.
    - This file is used to train the Core module used for dialogue management.

```yaml
responses:
  utter_greet:
  - text: "হ্যালো"
  - text: "ওহে"

  utter_goodbye:
  - text: "বিদায়"
  - text: "শুভ বিদায়"

  utter_type_of_services:
  - text: "আমরা মুলত XYZ বিষয়ে সার্ভিস দিয়ে থাকি। আমাদের সার্ভিস নিয়ে আরো বিস্তারিত জানতে আমাদের ওয়েবসাইট ভিসিট করুন।"
  - text: "স্যার/ম্যাম আমাদের x সংখ্যক ক্যাটাগরিতে x সংখ্যক প্রোডাক্ট/সার্ভিস আছে। এগুলো সম্পর্কে বিস্তারিত জানতে আমাদের ওয়েবসাইট ভিজিট করুন কিংবা চাইলে আমাদের কাস্টমার কেয়ার প্রতিনিধির সাথেও কথা বলতে পারেন।"

  utter_need_help:
  - text: "জ্বী স্যার/ম্যাডাম বলুন আমরা কীভাবে আপনাকে হেল্প করতে পারি?"
  - text: "জ্বী বলুন"
  - text: "আমরা আছি আপনাদের সেবায়"
  - text: "জ্বী স্যার/ম্যাডাম অবশ্যই"
  - text: "জ্বী । আমাকে প্রশ্ন করুন, আমি সবসময় চেষ্টা করব আপনার প্রয়োজনীয় উত্তর দিয়ে আপনাকে"
  - text: "জ্বী । নির্দ্বিধায় বলে ফেলুন। "
  - text: "জ্বী স্যার/ম্যাডাম বলুন আমরা কীভাবে আপনাকে হেল্প করতে পারি?"
  - text: "জ্বী স্যার/ম্যাডাম বলুন আমরা কি ইনফো দিয়ে আপনাকে হেল্প করতে পারি?"
```

Parts of the Dataset(cont.)

3. *stories.yml*
    - This file contains 139 stories each mimicking a conversation.
    - Each story consists of an intent defined in the nlu.yml followed by a response defined in the domain.yml.
    - The intent in the story represents a class of query that may come from a user.
    - Each action defines the response our chatbot should give to that input intent.
    - This file is also used to train the Core module used for dialogue management

```yaml
- story: 17.4 website
  steps:
    - intent: website
    - action: utter_website
    - intent: goodbye
    - action: utter_goodbye

- story: 18.3 mail_address
  steps:
    - intent: mail_address
    - action: utter_mail_address

- story: 18.4 mail_address
  steps:
    - intent: greet
    - action: utter_greet
    - intent: mail_address
    - action: utter_mail_address

- story: 19.3 year_of_starting
  steps:
    - intent: year_of_starting
    - action: utter_year_of_starting

- story: 19.4 year_of_starting
  steps:
    - intent: greet
    - action: utter_greet
    - intent: year_of_starting
    - action: utter_year_of_starting
```

Machine Learning Modeling

Rasa is made up of two separate decoupled units

- NLU module: is used to classify the intent of a given sentence and extract the entities
- Core module: is responsible for the dialogue management of the system

So how does the entire model work?

- From the input sentence the intents and entities are identified
- The intent and entities along with the previous state are used to update the current state of the conversation
- Policies use the output of the tracker to select an appropriate response from the domain file

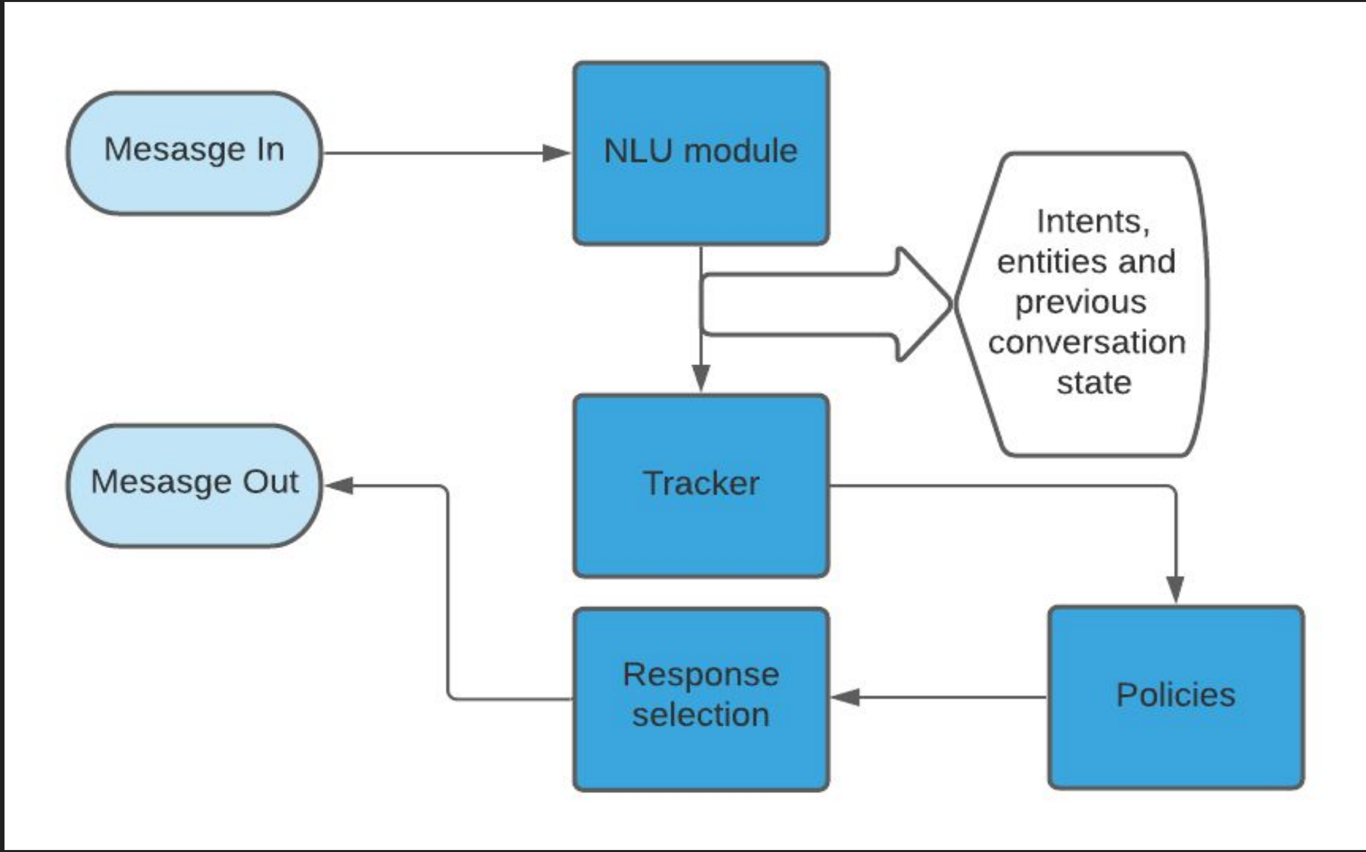


Fig no: How an output message is generated from an input message

The NLU module

- It is formed of a pipeline consisting of a series of steps executed consecutively
- The input text is tokenized
- Sparse featurizers like Lexical Syntactic Featurizer, Count Vector Featurizer extract sparse features
- Using these features the intent and entity predictions are made using the DIETClassifer
- The DIETClassifer takes in the features as input and outputs the intents and entities

```
language: bn

## Pipline 1 = vanilla pipeline
 pipeline:
   - name: WhitespaceTokenizer
   - name: RegexFeaturizer
   - name: LexicalSyntacticFeaturizer
   - name: CountVectorsFeaturizer
   - name: CountVectorsFeaturizer
     analyzer: char_wb
     min_ngram: 1
     max_ngram: 4
   - name: DIETClassifier
     epochs: 200

```

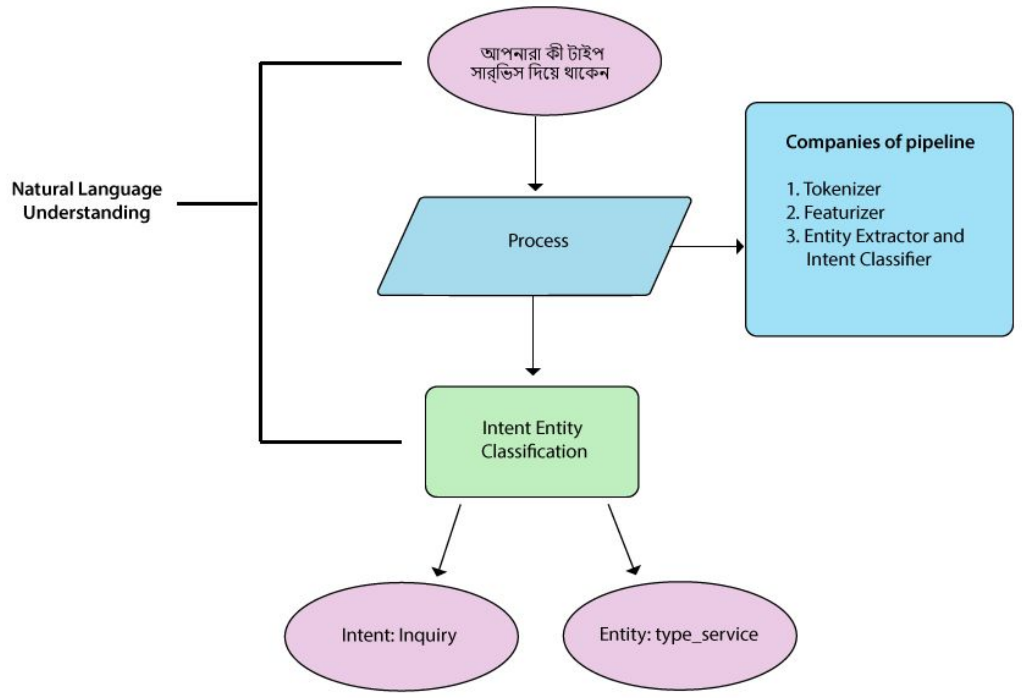


Fig no: How an NLU model works

How can we customize the NLU module?

- We can add language specific tokenizers for Bangla.
- We can leverage pre-trained dense features on Bangla corpus.
- Concatenating pre-trained dense features along with sparse features highly improves performance.
- FastText and Spacy both provide pre-trained dense features for Bangla.
- BERT provides a language agnostic pre-trained dense feature as well.

Different customized NLU pipeline

```yaml
pipeline:
- name: WhitespaceTokenizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: ftfeat.FastTextFeaturizer
  cache_dir: 'F:/Bot/fastText'
  file: 'cc.bn.300.bin'
- name: DIETClassifier
  epochs: 100
```

Fig no: pipeline with dense bangla fastText features

```yaml
pipeline:
  - name: custom_tokenizer.BanglaTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 200
```

Fig no: pipeline with custom bangla tokenizer

Different customized NLU pipeline(contd.)

```yaml
pipeline:
- name: SpacyNLP
  model: "bn_core_news_sm"
- name: SpacyTokenizer
- name: SpacyEntityExtractor
- name: SpacyFeaturizer
  pooling: mean
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
- name: FallbackClassifier
  threshold: 0.2
  ambiguity_threshold: 0.1
```

Fig no: pipeline with dense bangla spacy features

```yaml
pipeline:
    - name: "LanguageModelTokenizer"
    - name: LanguageModelFeaturizer
      model_name: "bert"
      model_weights: "rasa/LaBSE"
      cache_dir: null
    - name: RegexFeaturizer
    - name: LexicalSyntacticFeaturizer
    - name: CountVectorsFeaturizer
      analyzer: char_wb
      min_ngram: 1
      max_ngram: 4
    - name: DIETClassifier
      epochs: 500
   - name: FallbackClassifier
     threshold: 0.3
     ambiguity_threshold: 0.1
```

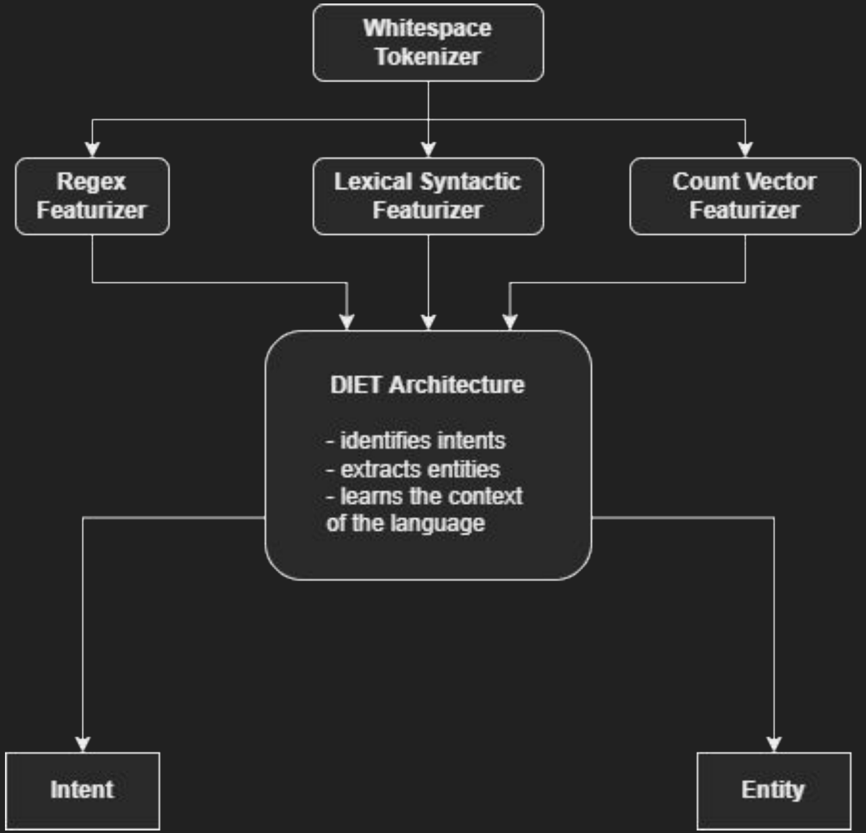
Fig no: pipeline with dense language agnostic Bert features

The Core module

- It includes a tracker and some Policies.
- The tracker keeps track of the conversation state and updates it.
- Policies are of two types: 1) ML based 2) Rule based
- The policies we used for response selection are
    - TED Policy: Transformer based policy to predict the next action to an input text
    - Memoization Policy: A rule-based policy that checks if the current conversation matches our defined stories and responds accordingly
    - Rule Policy: Used to implement fallback response when a out of scope input is fed to the model

Ablation Study

Pipeline P1



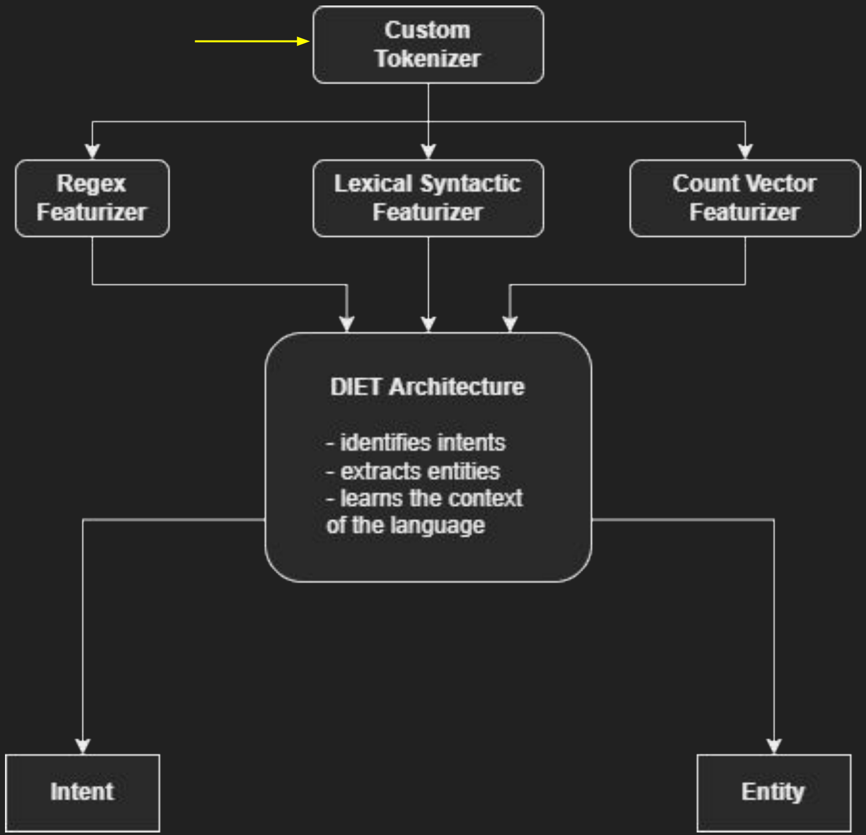
Add

Replace

Delete

NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75

Pipeline P2



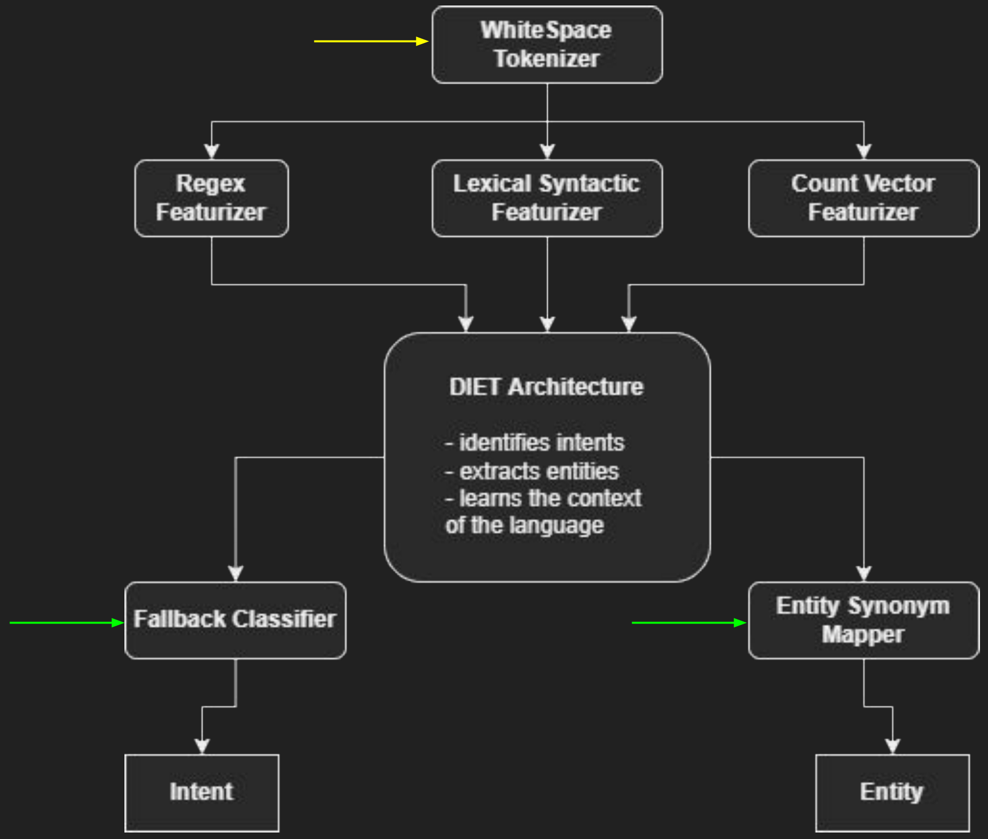
Add

Replace

Delete

NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82

Pipeline P3



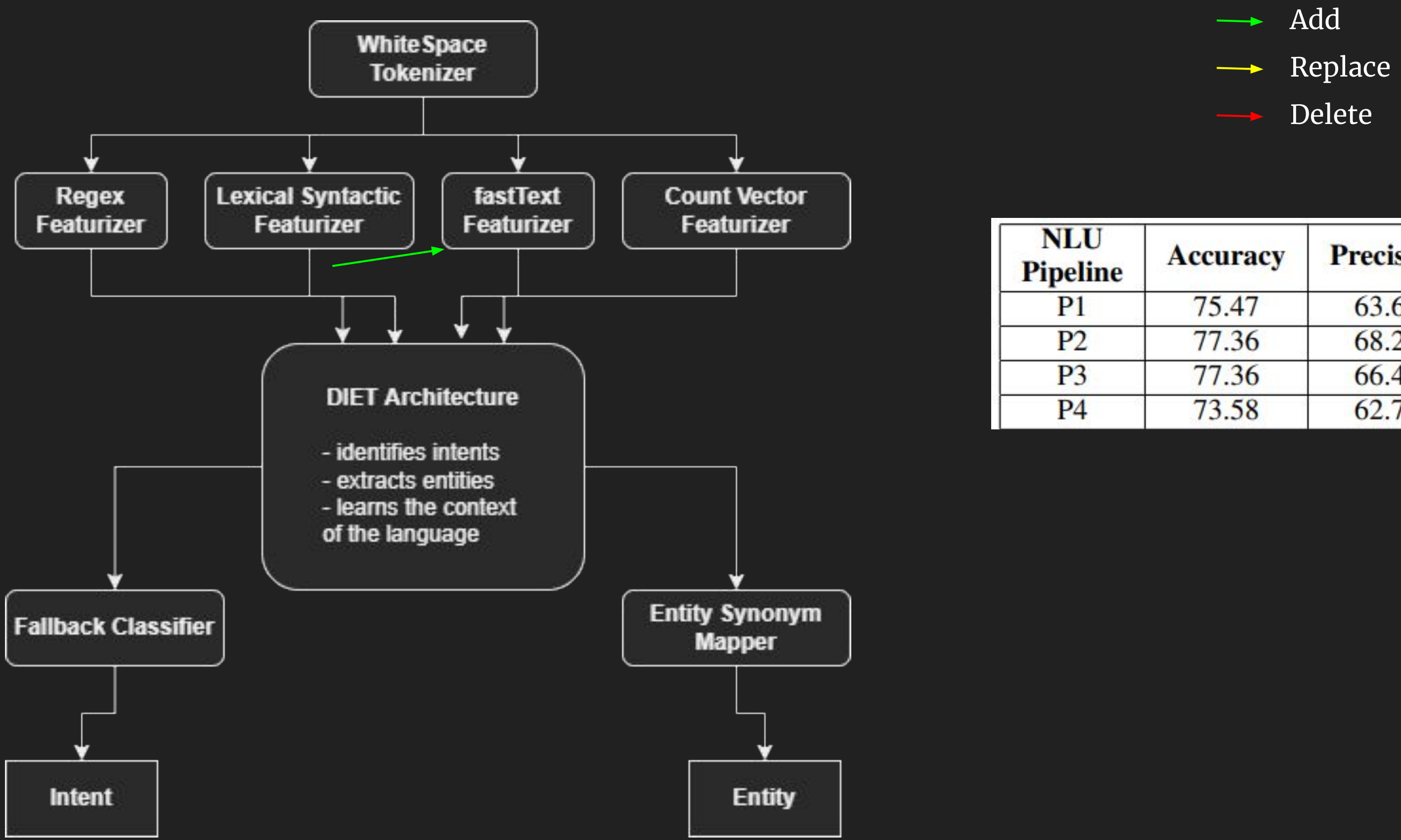
Add

Replace

Delete

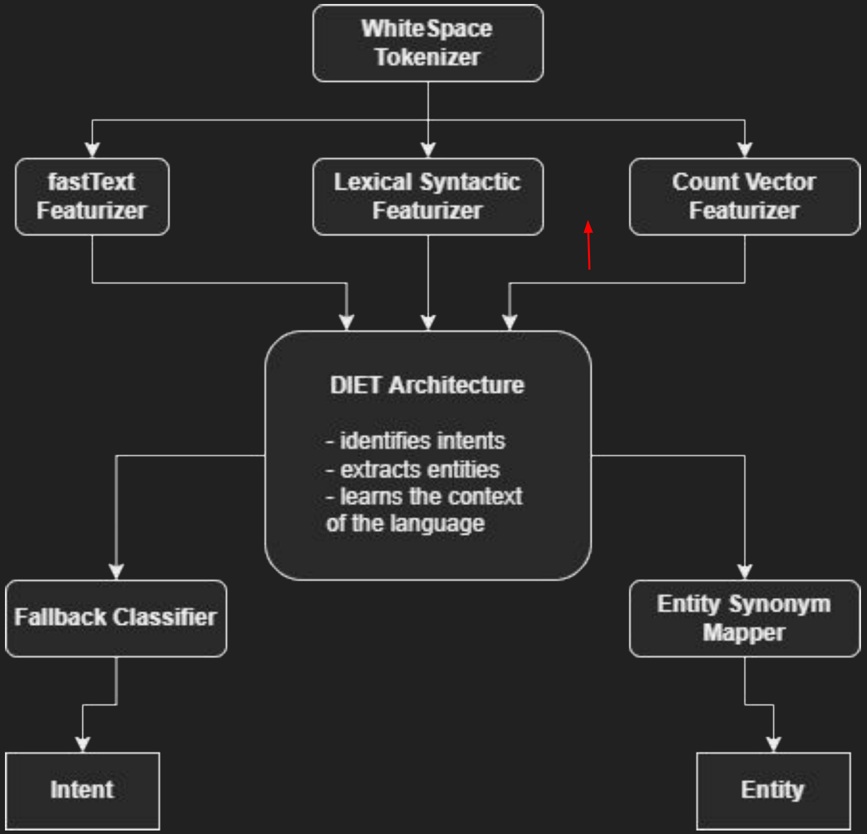
NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48

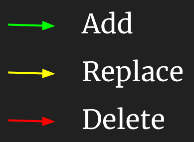
Pipeline P4



NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48
P4	73.58	62.74	73.58	66.49

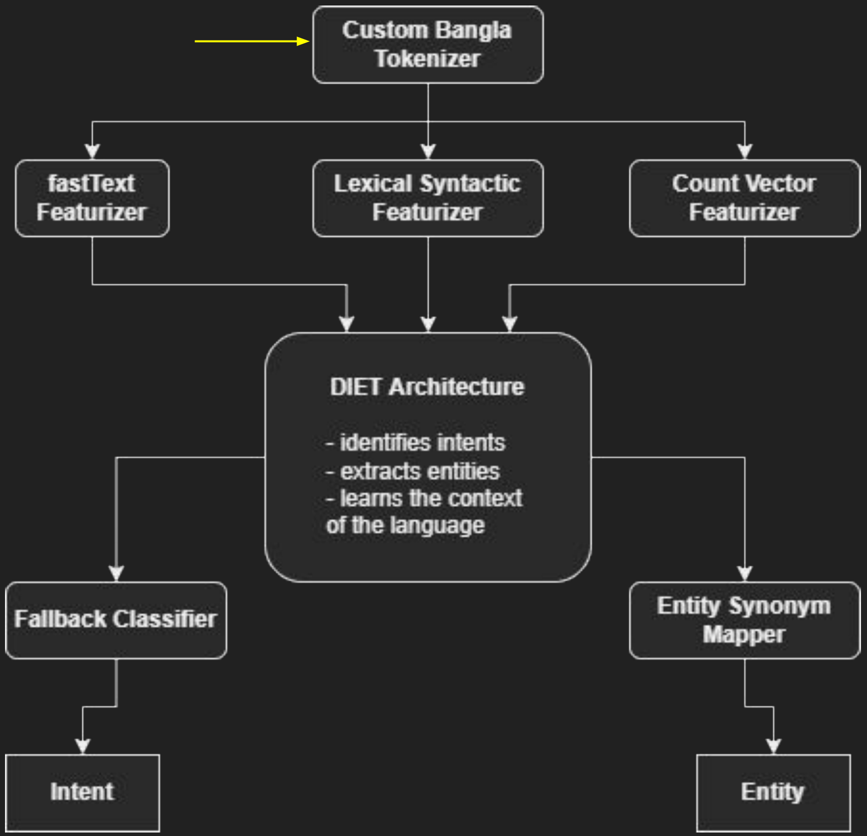
Pipeline P5





NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48
P4	73.58	62.74	73.58	66.49
P5	79.25	71.38	79.25	73.46

Pipeline P6



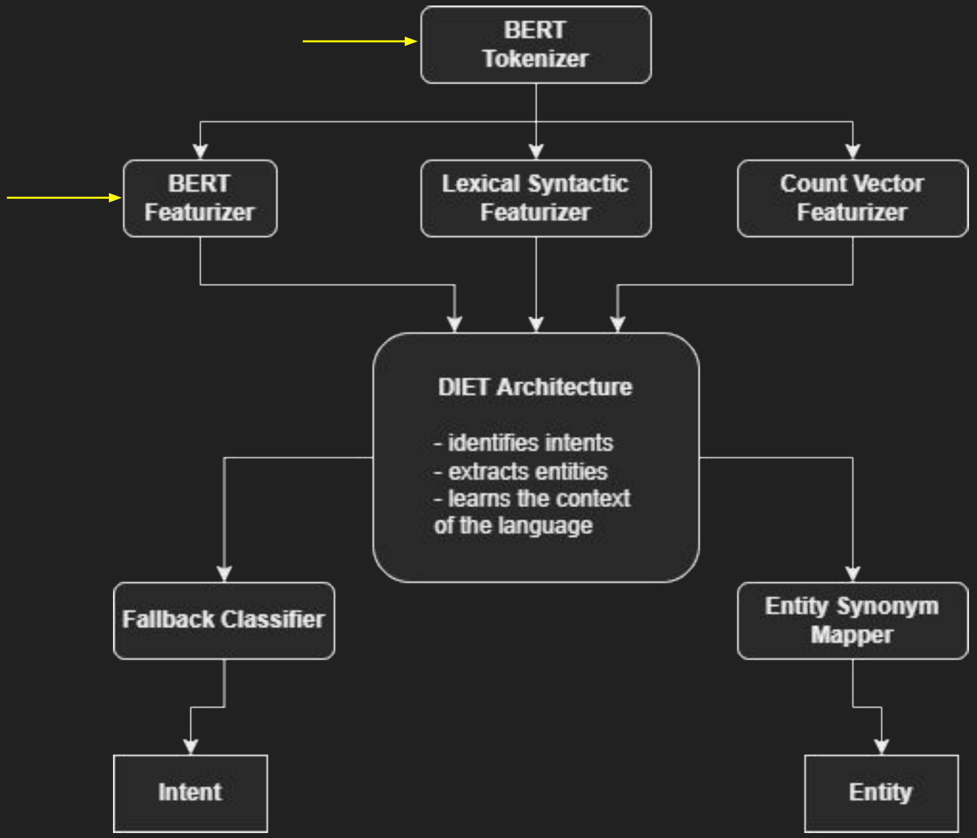
Add

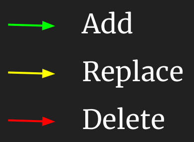
Replace

Delete

NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48
P4	73.58	62.74	73.58	66.49
P5	79.25	71.38	79.25	73.46
P6	81.13	73.27	81.13	75.32

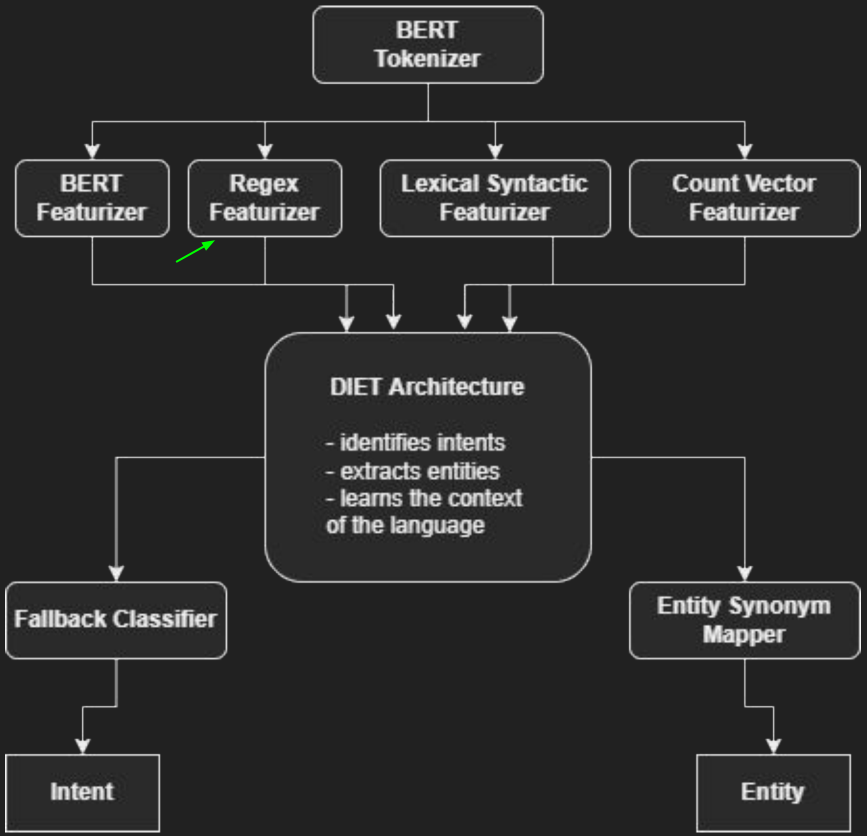
Pipeline P7





NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48
P4	73.58	62.74	73.58	66.49
P5	79.25	71.38	79.25	73.46
P6	81.13	73.27	81.13	75.32
P7	79.25	75.16	79.25	75.47

Pipeline P8



Add

Replace

Delete

NLU Pipeline	Accuracy	Precision	Recall	F1-Score
P1	75.47	63.65	75.47	67.75
P2	77.36	68.24	77.36	70.82
P3	77.36	66.45	77.36	70.48
P4	73.58	62.74	73.58	66.49
P5	79.25	71.38	79.25	73.46
P6	81.13	73.27	81.13	75.32
P7	79.25	75.16	79.25	75.47
P8	**83.02**	**80.82**	**83.02**	**80**

Experimental Setup

- We split the data into 80-20 train/test split

- THE NLU MODULE
  - Learning rate: 0.5
  - Optimizer: Adam
  - Epochs: 500
  - Minimum ngram for Count Vector Featurizer(sparse featurizer): 1
  - Maximum ngram for Count Vector Featurizer(sparse featurizer): 4
  - Fallback Classifier threshold: 0.3

- THE CORE MODULE
  - Max history: 5
  - Epochs: 200

Conclusion and Future Work

- Extend and improve the quality of our dataset.
- Use state of the art custom components like language specific dense featurizers for example: BERT embeddings trained on Bangla corpus.
- Change the model in a way as to handle conversations in several languages.

End-to-End Interaction

References