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)

B.Sc. (Hons) Degree in Information Technology specializing in Data Science

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

September 2022

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)

The dissertation was submitted in partial fulfillment of the requirements for the B.Sc. Special Honors degree in Information Technology

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

September 2022

DECLARATION, COPYRIGHT STATEMENT AND THE STATEMENT OF THE SUPERVISORS

I declare that this is my own work, and this dissertation does not incorporate any material previously submitted for a degree or diploma in any other university or Institute of higher learning without acknowledgement and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and redistribute my dissertation in whole or part in future works (such as articles or books).

Name	Student ID	Signature
Dissanayake D.M.I.M.	IT19069432	[illegible]

The above candidate is conducting research for the undergraduate dissertation under my supervision.

Name of the supervisor: Dr. Lakmini Abeywardhana
Signature of the supervisor: Date: 09/09/2022

Name of the co-supervisor: Ms. Dinuka Wijendra
Signature of the co-supervisor: Date: 09/09/2022

ABSTRACT

With the advancement of machine learning, it is often ambiguous to humans how state-of-the-art machine learning models deliver decisions and predictions despite their superiority when compared to statistical and rule-based machine learning models that are interpretable by design. Machine learning model explainability and interpretability is an emerging research interest among machine learning and artificial intelligence researchers, and this domain is often known as explainable artificial intelligence. Explainable artificial intelligence is one of the main objectives of responsible artificial intelligence, and it promotes the idea of explaining how complex and opaque machine learning models make decisions to establish trustworthiness through deriving human-interpretable explanations. Explainable Artificial Intelligence is crucial for distinct reasons, including identifying model biases, extending the model reliability, and finetuning or debugging the trained machine learning models. This research component focuses on developing an explainable artificial intelligence technique called Dual Interpretable Model-Agnostic Explanations to explain the predictions delivered by the Dual Intent Entity Transformer classifier: a transformer-based deep learning intent classification and entity recognition model utilized in Rasa conversational AI assistants. The explainable artificial intelligence technique discussed in this thesis focuses on generating human-interpretable explanations for a given text query and a trained Dual Intent Entity Transformer classifier by introducing the dual feature importance score: a combined feature importance score that blends both global and local feature importance scores.

Keywords: Explainable Artificial Intelligence, Natural Language Processing, Dual Intent Entity Transformer, Feature Importance, Dual Interpretable Model-Agnostic Explanations, Conversational AI

ACKNOWLEDGEMENT

I express my deepest gratitude to our team supervisor Dr. Lakmini Abeywardhana, and co-supervisor, Ms. Dinuka Wijendra, for their support, guidance, and feedback. This endeavour would not have been possible without their knowledge and expertise. I also appreciate the Sri Lanka Institute of Information Technology for hosting an extensive collection of publicly accessible documents and student and course guides on their official website, which I used to design the conversational AI dataset utilized during the research.

I am also grateful to the Sri Lanka Institute of Information Technology (SLIIT) research project team (CDAP) and lecturers who conducted invaluable research project (RP) module sessions for assisting in devising a well-prepared research timeline, document templates and guidelines. In addition, I should also mention the invaluable feedback given by the research progress evaluation panel throughout the research duration.

Finally, I am also grateful to everyone whom I might not have mentioned by their name who assisted me in conducting the research at any stage.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
BLEU	Bilingual Evaluation Understudy
CLI	Command Line Interface
DFI	Dual Feature Importance
DIET	Dual Intent Entity Transformer
DIME	Dual Interpretable Model-Agnostic Explanations
DNN	Deep Neural Network
ELI5	Explain Like I’m 5
GDPR	General Data Protection Regulation
GFI	Global Feature Importance
GUI	Graphical User Interface
I/O	Input/Output
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LFI	Local Feature Importance
LIME	Local Interpretable Model-Agnostic Explanations
LUIS	Language Understanding
MAGIX	Model Agnostic Globally Interpretable Explanations
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
NPM	Node Package Manager
OS	Operating System
PDF	Portable Document Format
PDP	Partial Dependency Plot
PyPI	Python Package Index

RP	Research Project
SHAP	Shapley Additive Explanations
SLIIT	Sri Lanka Institute of Information Technology
UI	User Interface
WSGI	Web Server Gateway Interface
XAI	Explainable Artificial Intelligence
YAML	Yet Another Markup Language

1. INTRODUCTION

1.1 Background & Literature survey

Applications from diverse domains and scales have adopted Artificial Intelligence (AI) and comprise state-of-the-art machine learning (ML) models. However, with rapid advancements in AI and ML, machine learning models can fall under white-box and black-box models. Typically, black-box ML models are superior to white-box ML models in terms of performance and are far more complex in terms of architecture, making it impossible for humans to explain precisely how these models arrive at a specific decision. In contrast, a white-box ML model is an intrinsically human-interpretable machine learning model that is transparent by design [1].

Conversational AIs or dialogue engines are an advanced implementation of chatbots and comprise advanced machine learning and deep learning models to perform natural language processing (NLP), natural language generation (NLG), and dialogue management. A conversational AI is foremost when compared to traditional chatbots with rule-based, pattern-matching, or other statistical machine learning models and is notably precise [2]. NLP is a subfield in AI for processing natural language data and is widely utilized in modern conversational AI assistants for tasks such as intent classification and entity recognition [2], [3]. Intent classification falls under predictive analytics and can assist conversational AI assistants in classifying the user intent of a specific natural language query under one of the pre-defined intents (also known as classes). Most modern conversational AI development platforms comprise advanced deep learning models such as transformers for the intent classification task, and the exact process of the classification remains non-human-interpretable due to the opaque nature of these models [3], [4].

The utilization of advanced machine learning and deep learning models within conversational AI assistants and other AI-centric applications entirely based on the model performance often raises the question of the reliability and trustworthiness of its outcomes. References [5] and [6] reveal that deep neural network (DNN) based text classifiers are manipulatable and easily fooled by slightly changes done to the inputs.

Although advanced black-box models and techniques can deliver promising results, their fairness, reliability, and level of privacy are questionable due to the lack of transparency [1], [7]. Data protection policies and regulations, such as European Union's General Data Protection Regulation (GDPR) (Article 13-15 and Article 22), mention the importance of transparency and accentuate the right for explanations for decisions made solely based on automated decision-making approaches and opaque algorithms that an average person cannot interpret [8], [9].

Responsible AI proposes a governance framework that ensures the reliability and fairness of AI-centric applications, and it promotes five principles: (1) Transparency, (2) Fairness, (3) non-maleficence, (4) Responsibility, and (5) Privacy [10]. According to [7], government organizations and other professional bodies have disseminated over 600 AI-related policies and guidelines that fully or partially promote the above principles. However, vendors enforce these policies and regulations in significantly diverse ways, making it formidable to discern the level of reliability and accountability offered with AI solutions. The fact that data-driven AI-centric approaches are problematic regarding biases in data and lack of transparency of the algorithms provokes AI-centric organizations and applications to utilize AI with a higher degree of responsibility and accountability [7].

Explainable AI (XAI) is still a young and emerging subfield of AI that focuses on implementing elements of the transparency principle in the set of responsible AI principles concerning the opaqueness of AI algorithms. XAI utilizes various explainability techniques to explain to an average person how a machine learning or deep learning model opaque in nature arrives at its outcomes. Expanding on the domain of NLP, traditional NLP techniques often rely on inherently explainable algorithms such as rules-based approaches and decision tree classifiers. Advanced NLP techniques have decreased human interpretability over time due to their complicated nature, known as the trade-off between model interpretability and performance. In general, less complicated models are human interpretable, while complex models require interpretations that comprise valid explanations generated to provide an acceptable justification for how the model arrives at its predictions. Word embeddings, encoder-decoder architecture, and other deep learning-based approaches utilized in NLP are a few examples of black-box techniques susceptible to the trade-off between model interpretability and performance [1], [11], [12]. Interpretability and explainability are broadly used terms in the XAI domain that are considered interchangeable but can often be confusing since they have two distinct definitions [13]. In XAI, an interpretation is the process of mapping abstract concepts to a domain that an average person can comprehend, while an explanation is a feature-level justification for a specific decision taken by a black-box model [11], [14]. Explainability focuses on justifying the outcomes arrived at by opaque ML models, often by identifying the features that contributed the most toward a given black-box model outcome.

Intrinsic and post hoc approaches are the two widely used XAI techniques. The intrinsic XAI approach refers to white-box ML models that are interpretable by design, while the post hoc approach refers to generating explanations for already-trained black-box ML models. Moreover, the post hoc XAI approaches can be model-specific, model-agnostic, local interpretable, or global-interpretable, as depicted in Figure 1.1.1 and Figure 1.1.2 [1], [11]. Model-specific XAI techniques only support explaining a specific family of ML models, while model-agnostic XAI approaches are applicable for various ML model types and architectures. Model-agnostic XAI techniques allow differing model architectures to generate model explanations regardless of their diverse internal structure. Recent XAI research in the NLP domain has introduced widely known model-agnostic XAI techniques such as Local Interpretable Model-agnostic Explanations (LIME) [15] and Shapley Additive Explanations (SHAP) [16] that have proven their value in contrast to the model-specific approaches in terms of compatibility and portability.

In NLP terminology, a data instance is a sentence, paragraph, or question that consists of related words or phrases. Training or a validation text dataset (also known as a text corpus in NLP terminology) is an ample collection of data instances. NLP-based ML or deep learning models go through the training and validation text corpora during the training and evaluation stages. Local interpretable explanation generation

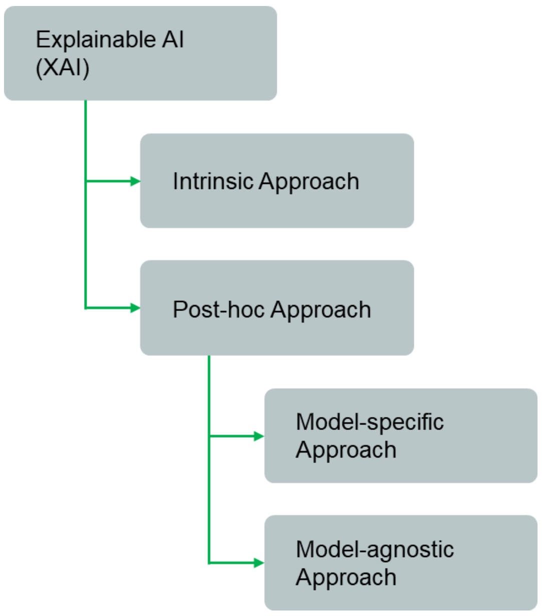


Figure 1.1.1. Categories of Explainable AI approaches

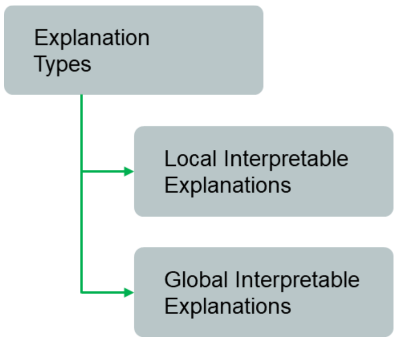

Explanation Types
Local Interpretable Explanations
Global Interpretable Explanations

Figure 1.1.2. Types of explanations

stands for generating explanations around a single data instance. i.e., Local-interpretable XAI techniques offer insights into how a specified model behaves around a single data instance, discarding how the model behaves for other data instances in the corpus. In contrast, global interpretable XAI techniques offer high-level insights considering all the data instances in a specified text corpus.

Most XAI techniques have utilized feature importance as the primary method of computing and delivering explanations, among other techniques, including surrogate modelling, example-driven explanation generation, provenance-based explanation generation, and declarative induction. High utilization of feature importance in XAI is recurrent because features serve as building blocks for ML models, which are comparably easier to explain. XAI techniques based on feature importance investigate the importance of unique features to detect the most influential features to the model outcome. With relevance to NLP, unique features can be either the features generated manually using feature engineering methods or lexical features such as distinct tokens and n-grams directly taken from the input [1].

Feature importance calculation can vary depending on the local or global scope considered. LIME [15], a locally interpretable XAI approach, uses data augmentation to generate additional data instances (known as input perturbations) close to the original data instance based on distance matrices such as cosine similarity and trains an interpretable local surrogate model [1], [15]. Shapley values are another local feature importance (LFI) calculation technique that learns the contribution of each feature toward the model outcome based on concepts found in cooperative game theory [16], [17]. Feature importance in the global setting considers all data instances in a specified corpus to calculate the feature importance score of individual tokens. i.e., In global interpretable explanation generation, the model behaviour around all data points is considered [18]. Permutation feature importance and partial dependency plot-based (PDP) feature importance are widely utilized techniques to calculate global feature importance (GFI) in XAI [19], [20].

Visualizing explanations is another main aspect of XAI. Current XAI approaches widely use saliency visualizations to present explanations, among other visualization techniques, including template-based visualizations, raw (natural language) visualizations, and raw declarative representations, which are perceivable to an average person on varying levels. Saliency heatmaps and saliency highlighting are two widely used saliency visualization techniques in XAI in the NLP domain [1]. LIME and SHAP utilize saliency highlighting to visualize text classification explanations [15], [16]. Figure 1.1.3 holds a sample LIME saliency visualization.

Evaluating XAI approaches is essential to quantify their effectiveness. Even though XAI is still an emerging topic, recent XAI research has shed light on a few XAI

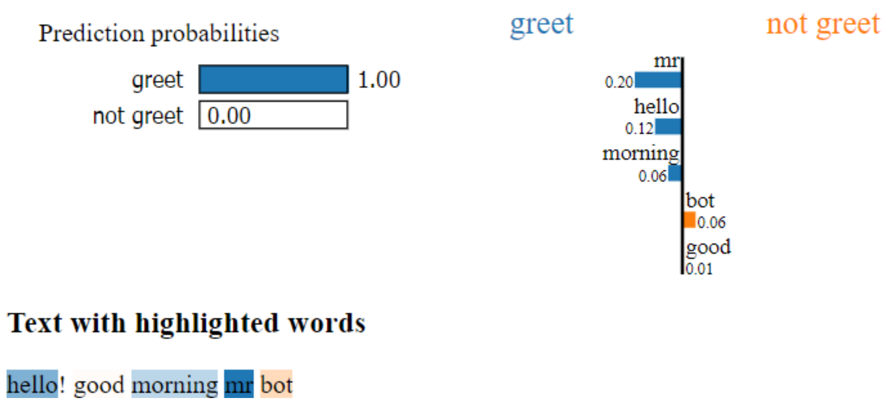


Figure 1.1.3. A sample LIME saliency visualization

evaluation techniques, namely, (1) comparison to ground truth and (2) human-centric evaluation. Evaluating based on the sole outcome of a novel XAI technique without proper evaluation metrics or comparing a new XAI technique with an existing XAI technique such as LIME is considered an invalid or informal evaluation standard [1]. Perplexity, Bilingual Evaluation Understudy Score (BLEU), precision, recall, and F1-score are a few evaluation metrics that fall under the ground truth comparison-based evaluation [21], [22]. Human-centric evaluation is more straightforward and often involves a cluster of humans rating the explanations generated by the proposed XAI technique. [23], [24]. However, the exact evaluation approach depends entirely on the evaluated factor, such as fidelity (the quality of generated explanations) and comprehensibility (the ability of a target group to understand the explanations or visualizations) [1], [25]. According to one of the studies done by Robnik-Sikonja and Bohanec [26], XAI methods have four characteristics, namely, (1). Expressive power, (2) Translucency, (3) Portability, and (4) Algorithmic-complexity, and the same study mentions nine characteristics of explanations generated by an XAI method, namely, (1) Accuracy, (2) Fidelity, (3) Consistency, (4) Stability, (5) Comprehensibility, (6) Certainty, (7) Degree of Importance, (8) Novelty, and (9) Representativeness. These characteristics can assist in designing better XAI methods that can deliver high-quality explanations [19].

References [11] and [12] reveal how blindly trusting black-box machine learning models has led to a pressing issue, why human interpretability for black-box models is essential, and how XAI can bridge the gap between interpretability and model performance. Reference [27] has clearly stated the importance of explainability to reinforce decision-making outputs and support humanity's requirement for scientific understanding. Conversational AI and advanced chatbot development platforms and tools can incorporate XAI to deliver insights into debugging ML models utilized in their natural language understanding (NLU) pipeline and drastically increase the reliability of various NLU models, including intent classification and entity annotation ML models. Conversational AI vendors will be able to deliver exceptionally reliable solutions with not just performance scores of the ML models utilized but with valid explanations to reinforce their performance claims.

This research component focuses on devising a feature importance-based model-agnostic novel XAI technique and integrating it as a component of Kolloqe: a no-code conversational AI development platform delivered as the primary research output. The author of this research component evaluates the XAI technique introduced using human-centric evaluation strategies to quantify the effectiveness of explanations and generated visualizations by assessing the (1) reliability of data instance-level explanations for non-experts, (2) reliability of data instance-level explanations for ML experts to perform feature selection, and (3) comprehensibility of the explanation visualizations. Dual Interpretable Model Agnostic Explanations (DIME) XAI Python package[1] is the main open-source contribution of this research component. DIME-XAI package allows chatbot developers and NLP researchers to enable explainability in Rasa Conversational AI projects without writing a single line of code.

1.2 Research Gap

Despite XAI being an emerging field relatively young, a few studies have focused on explaining black-box models concerning the NLP domain, and most studies have presented explanations using token-level contributions to justify the model decisions [1]. Recently published literature surveys and research [1], [11], [14], [18], and [28] were immensely helpful in getting an idea about the existing research gap through exploring the current state of XAI in the NLP domain. XAI research such as LIME

[1] https://pypi.org/project/dime-xai/

[15], SHAP [16] and XAI libraries such as Explain Like I'm Five (ELI5) [29] have discussed diverse approaches to implement explainability for both NLP and non-NLP ML models. This section defines the research gap under explainability in NLP and explainability in chatbot development platforms.

1.2.1 Explainability for NLP

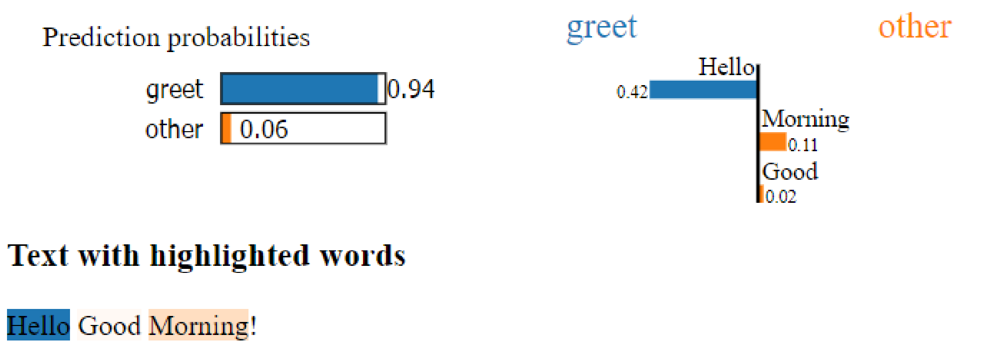
LIME has discussed generating local interpretable explanations in a model-agnostic manner, taking a trained model and relevant set of classes as arguments to the LIME text explainer and training a local interpretable surrogate model [15]. LIME offers a Python implementation[2] with built-in support for sci-kit learn-based classification models, and on top of that, it supports any classification model that can output class probabilities for a given data instance, which makes LIME model-agnostic, as its name suggests. In addition to text classifiers, LIME has extensive support for diverse ML model categories, including image classifiers and classifiers trained on tabular data (both numerical and categorical). However, the paper explains that identifying global faithfulness remains a challenge in complex black-box models and suggests that the explanations should be at least locally faithful. In short, the local fidelity of LIME explanations does not necessarily guarantee global faithfulness. Another noticeable issue with LIME is the instability of the delivered explanations [30]. Figure 1.2.1.1 depicts two LIME explanations for the same data instance. Although the feature score ranking of the first explanation (a) and the second explanation (b) remains intact, individual feature importance scores are inconsistent. Reference [30] explains LIME explanation instability in detail by taking two close data points and their LIME explanations.

SHAP paper discusses a slightly different approach to explaining individual predictions. SHAP estimates local feature contributions toward a specific model prediction inspired by shapely values found in cooperative game theory [17] and LIME. The authors introduce a few implementations of SHAP[3], including KernalSHAP, TreeSHAP, and DeepSHAP. KernalSHAP is an estimation approach for

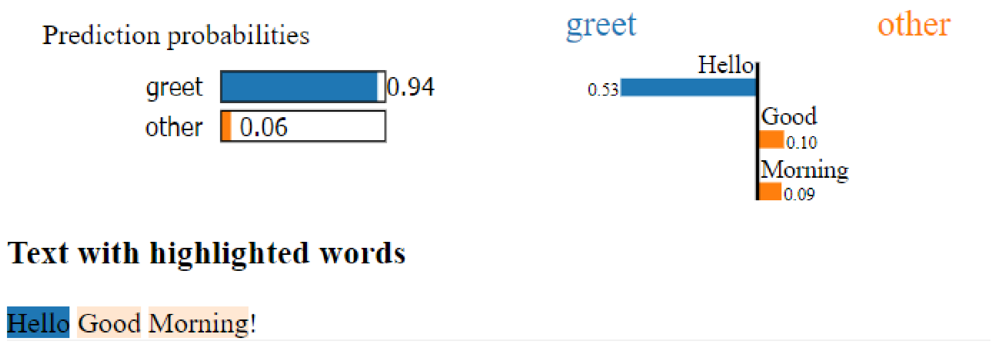
[2] https://github.com/marcotcr/lime
[3] https://github.com/slundberg/shap

Shapley values since directly calculating Shapley values is computationally intensive. TreeSHAP is the estimation method that the authors have introduced for tree-based classifiers, and DeepSHAP is for explaining deep learning models such as TensorFlow and Keras. SHAP also can calculate GFI by aggregating the absolute SHAP values of individual features. Since KernalSHAP is somewhat slow, utilizing SHAP to compute feature importance globally is considered impractical [16], [19].

For the sampling step (input perturbation), neither LIME nor KernalSHAP considers feature dependence, and it can result in generating unrealistic data samples. However, TreeSHAP is an exception [20]. Another issue with LIME and SHAP explainable AI techniques is that they are both vulnerable to adversarial attacks since



(a) LIME explanations for the first run (Scores: [('Hello', -0.4179474911423747), ('Morning', 0.10790955365652435), ('Good', 0.020088634633055855)])



(b) LIME explanations for the second run (Scores: [('Hello', -0.5265343656147796), ('Good', 0.10275867505678518), ('Morning', 0.09401509281285964)])

Figure 1.2.1.1. Two different runs, *(a)* and *(b)*, delivered inconsistent LIME explanations for the same data instance.

they rely on input perturbations. i.e., Both LIME and SHAP explanations are consciously manipulatable by an interested third party by moderating the data instance perturbation process. Therefore, model explanations delivered by LIME and SHAP can arguably be less reliable [19], [31].

The python XAI library ELI5[4] calculates local explanations using LIME. ELI5 also provides global model explanations based on the permutation feature importance technique by replacing words (also known as features in machine learning terminology and as tokens in NLP terminology) with random words (noise), which is closely related to the XAI approach introduced in this research component [29]. Model Agnostic Globally Interpretable Explanations (MAGIX) [32] is another XAI technique that utilizes local explanations derived using LIME to construct global explanations with the help of genetic algorithms.

Although some researchers have attempted to calculate the GFI based on individual LFI scores, none of the referred research has mentioned deriving LFI using global-level feature importance. Table 1.2.1.1 presents the most relevant, well-implemented, and widely utilized XAI-NLP research and the XAI techniques in contrast to the work done in this research component [33], [34], [35]. However, this research focuses on a different XAI approach that is notably different from the other approaches and proposes utilizing GFI as the feature selection criterion for generating LFI scores to ensure the global faithfulness of the explanations. The XAI approach introduced in this research also extensively minimizes adversarial attack vulnerabilities due to the absence of input perturbation.

1.2.2 Explainability for chatbot development platforms

Apart from generalized XAI approaches for NLP, none of the referred research has investigated integrating XAI techniques with Conversational AI assistants and chatbot development platforms. Even the most recent surveys focused on exploring recent

[4] https://github.com/eli5-org/eli5

Table 1.2.1.1. Comparison of XAI-NLP research in contrast to this research

Research Name	**Intrinsic /Post-hoc**	**XAI Scope (Local/ Global)**	**Model-specificity**	**Visualization Technique**	**Feature contribution calculation**
LIME	Post hoc	Local	Model Agnostic	Saliency highlighting, Raw scores	Input perturbation + Local linear surrogate model + Ridge Regression
SHAP	Post hoc	Local or Global	Model Agnostic	Saliency highlighting, Raw scores	LIME, DeepLIFT, and other approaches with SHAP values with kernel approximation.
Deep LIFT	Post hoc	Local	Model Specific	Sequence visualizations	Back-propagating feature contributions.
Self-Explain	Self-explaining	Local or Global	Model Specific	Raw scores. (Paper does not elaborate on an exact visualizing technique)	Regularization with explanation specific losses and phrase-based concept instead of words.
DIME *(This Research Component)*	Post hoc	Local **and** Global combined	Model Agnostic	Global and Dual Feature importance plots, Raw scores	Global feature importance-based feature selection to calculate local feature importance (introduced as **Dual Feature Importance**)

advancements in natural language understanding services, including conversational AI assistants and chatbots, have not mentioned explainability as a prominent aspect to consider or a future direction to study [28], [36]. Reference [37] and other similar research have also failed to accentuate the importance of having built-in support for explainability as a crucial factor when selecting a chatbot development platform or a framework.

Zooming in on chatbot development platforms and frameworks, none of the widely utilized chatbot building tools, including Google DialogFlow[5], Amazon Lex[6], Microsoft Language Understanding (LUIS)[7], Microsoft Bot Framework[8], and Rasa Open Source[9], offers built-in support for explainability-based model validation and

[5] Documentation of DialogFlow CX and ES are available and can be accessed online at https://cloud.google.com/dialogflow/cx/docs and https://cloud.google.com/dialogflow/es/docs
[6] https://docs.aws.amazon.com/lex/index.html
[7] https://learn.microsoft.com/en-us/azure/cognitive-services/luis/
[8] https://learn.microsoft.com/en-us/azure/bot-service/index-bf-sdk
[9] https://rasa.com/docs/rasa/

debugging. The specified platforms are cloud-based except for Rasa Open Source, which supports hosting locally, and most interestingly, Rasa provides extended control over the chatbot development tasks, including the NLU pipeline and ML components. However, even Rasa does not support explainability and relies on performance scores given by testing approaches such as hold-out validation and cross-validation to evaluate ML model performance [2], [38]. The absence of built-in support for XAI in widely utilized AI-enabled chatbot development frameworks and platforms is a significant research gap. This research component strives to emphasize the importance of XAI in modern chatbot building platforms by providing a solid XAI implementation for Rasa-based chatbot development.

1.2.3 Other related work

Reference [27] has proposed a conversational XAI approach based on the argument that the interpretability tools must be bidirectional rather than monolithic. The proposed solution takes the interpretability goals of the users into account through conversing and delivers natural language interpretations according to the questions presented by users. The technique is not directly related to the work done in this research component since implementing a bidirectional interpretability tool is not one of the primary goals. However, most explanations obtained through conversing mentioned in the study will be available to users via static visualizations generated by the XAI technique introduced in this research.

Reference [39] has introduced explainability for delivering constantly-refined recommendations through conversing in a chatbot-based recommendation engine. The explanations given by the chatbot are natural language (raw) explanations that justify the recommendation engine decisions. Although this technique is well-suited for

Table 1.2.2.1. Chatbot development platforms and their support for XAI

Chatbot Development Platform/ Framework	**Support for Explainability**	**Support for ML Model Evaluation**	**Model Evaluation Technique**
DialogFlow	No built-in support	No built-in support	-
Amazon Lex	No built-in support	No built-in support	-
Microsoft LUIS	No built-in support	No built-in support	-
Rasa Open Source	No built-in support	Yes	Hold-out and Cross-validation

conversational recommendation engines, it is not the most appropriate XAI approach for general conversational AI assistants and chatbots, where providing explanations to the end-users is not the priority. The specified research has not explored how XAI can assist in debugging and validating machine learning models in chatbots, which is the area of concern in this research component.

Reference [40] promotes the idea of a conversational XAI agent based on the argument that the delivery of explanations should be similar to human-to-human conversations. The proposed solution comprises an NLU component for processing the user requests and selects the best-suited explainability technique from various XAI approaches. References [41], [42], [43], [44] propose similar XAI techniques and concepts built around conversational XAI agents. Although the proposed solution is insightful and interactive, conversational XAI agents are not a priority in this research component. Instead, this research component presents a novel model agnostic XAI approach that delivers human interpretable explanations for intent classification models utilized in conversational AI assistants and chatbots and focuses on integrating it with a chatbot development platform to enable model debugging and validation.

Reference [45] introduces ChatrEx: an in-app task-oriented chatbot that can converse underlying steps of a task and explain the reason for task failures. However, in contrast to ChatrEx, this research component focuses on standard chatbots and conversational AIs rather than task-oriented chatbots and offers model explanations to debug the ML model utilized in chatbots instead of delivering model explanations directly through conversations.

2. RESEARCH PROBLEM

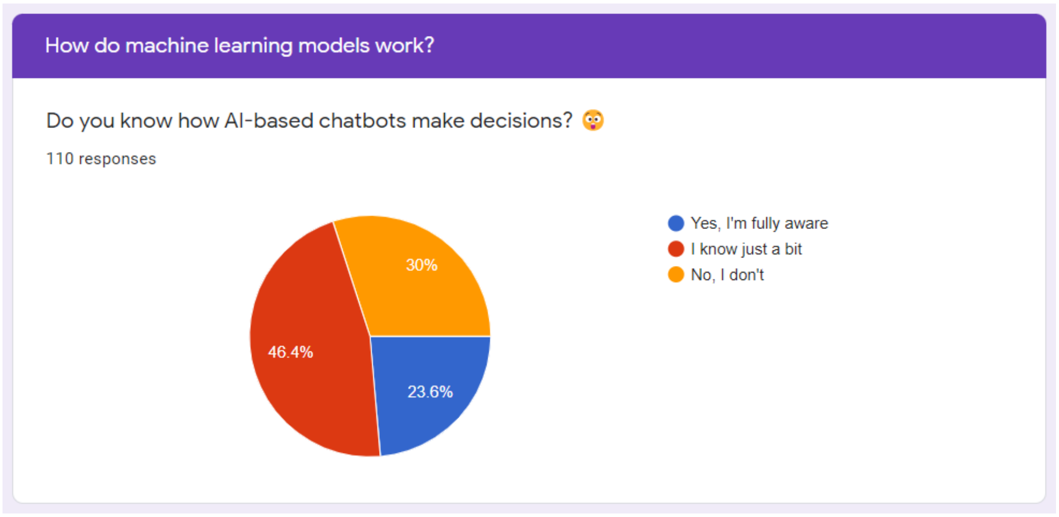
As previously explained, algorithms and machine learning models are getting more complex frequently. Widely utilized applications such as chatbots have employed machine learning and NLP due to high performance and accuracy compared to traditional approaches. These complex algorithms and advanced machine learning techniques, such as deep learning, are opaque by design, and humans fail to comprehend how these models work internally to make decisions [11]. As explained in section 1.2, most currently available local interpretable XAI approaches are not globally faithful. Reference [1] states that the authors have only found four research papers in the global explanation category. Many text explainers in the XAI domain focus on generating local model explanations, and only a few explainers support generating global model explanations by aggregating local model explanations. However, the accuracy of the aggregated model explanations is often distrustful since the global faithfulness of the individual local model explanations is not guaranteed. None of the referred studies has considered generating model explanations that are locally and globally faithful, mainly due to global explanation calculation being computationally intensive [1]. This research component concentrates on finding the local model explanations based on globally important features and enhancing the efficiency of the GFI calculation, making the explanations generated by the introduced XAI approach globally and locally faithful.

Another conspicuous issue is that input perturbation-based XAI techniques are prone to adversarial attacks. Widely utilized XAI text explainers, including LIME and SHAP, employ input perturbation for sampling, which makes the explanations generated by these methods unreliable [31]. The inconsistent and manipulatable nature of these XAI techniques poses a direct threat to the reliability of the model explanations. This research component concentrates on eliminating the input perturbation layer to make the delivered model explanations more reliable and immune to adversarial attacks.

It is arguably easy to integrate an existing XAI technique with ML models built from scratch using well-known machine learning frameworks such as TensorFlow[10], PyTorch[11], or Scikit-learn[12] because the developers are fully aware of the ML model architecture, and the ML frameworks provide a comprehensive API to interact with the models. i.e., The developers have the required tools and documentation to enable XAI in models built with well-known ML frameworks from scratch. Most XAI libraries provide built-in support for models built with well-known ML frameworks and are well-documented. For example, LIME has built-in support for Scikit-learn models, and DeepLIFT implementation is fully compatible with TensorFlow and Keras. However, although many NLP-based XAI explainers exist, they cannot generate explanations for all ML model types. It is often impossible to find model explanations for application-specific ML models without comprehensive technical and ML knowledge, even with model-agnostic explainers such as LIME [15] and SHAP [16]. The ML model inside the chatbot framework Rasa is an example of such a model. The models trained and stored within Rasa contain a cluster of machine learning models, NLP pipeline components, and metadata. Rasa NLU is responsible for extracting the model, loading the pipeline components, and processing textual data. Generating model explanations for intent classifiers such as the Dual Intent Entity Transformer (DIET) classifier is tedious since Rasa does not provide a proper toolset or an API to interact with specific ML models utilized in the NLU pipeline. Extracting prediction probabilities from DIET on demand is also not directly supported, which is required by many model-agnostic text explainers [2], [3], [15]. However, it is more acceptable if there is a way to interpret these application-specific ML models by chatbot developers to trust the predictions given by the models employed within these widely utilized frameworks instead of having to manually attach an XAI library or only rely on performance scores that do not evaluate the reliability of the model decisions at all.

[10] https://pypi.org/project/tensorflow/
[11] https://pypi.org/project/pytorch/
[12] https://pypi.org/project/scikit-learn/

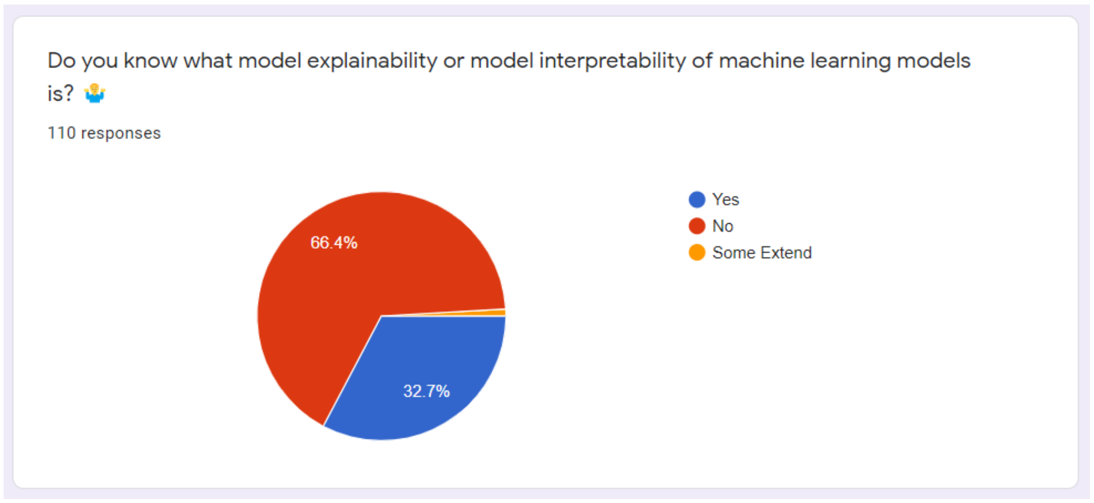
Moreover, a recent survey that studied the current trends in ML model interpretability and explainability confirms that the majority prefers explanations for the machine learning model predictions. The population of interest in the survey was 110 Undergraduates of the Faculty of Computing of Sri Lanka Institute of Information Technology, selected based on the technical nature of the survey questions. Refer to Appendix A: Survey Form to observe the complete survey questions and responses. Responses in Figure 2.1 clearly show that more than three-fourths of the population has little to no understanding of the NLU techniques used within chatbot frameworks and the intent classification process. Although 32.7% of the population has claimed that they are aware of the terms model interpretability and explainability (Figure 2.2), Figure 2.3 clearly illustrates that only 26.4% have used text explainer tools such as LIME and SHAP. 66.4% of the total population is unaware of model explainability and interpretability.


How do machine learning models work?

Do you know how AI-based chatbots make decisions?

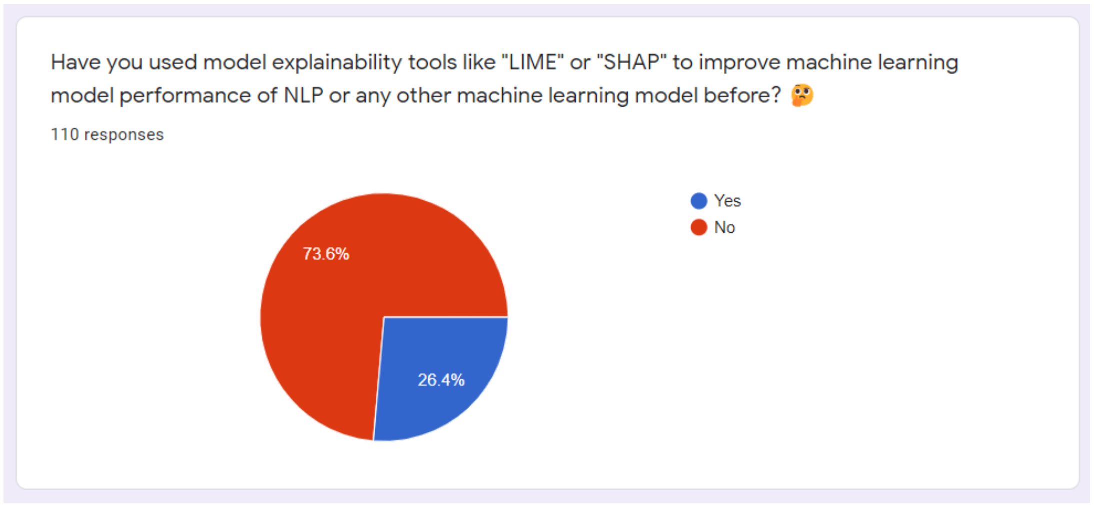
110 responses

Figure 2.1. Summary of survey responses received for the question "Do you know how AI-based chatbots make decisions?"


Do you know what model explainability or model interpretability of machine learning models is?

110 responses

Figure 2.2. Summary of the responses received for the survey question "Do you know what model explainability or model interpretability of machine learning models is?"


Have you used model explainability tools like "LIME" or "SHAP" to improve machine learning model performance of NLP or any other machine learning model before?

110 responses

Figure 2.3. Summary of the responses received for the survey question "Have you used model explainability tools”

3. RESEARCH OBJECTIVES

3.1 Main Objectives

The main objective of this individual research component is to implement DIME, a novel model-agnostic locally and globally faithful XAI approach, to deliver local model explanations that comprise only globally influential features. The DIME implementation concentrates on computing the GFI of the tokens present in a specified text data instance to sift only the most significant features efficiently to calculate the LFI.

3.2 Specific Objectives

The specific objectives of this research component are as follows.

1. Collect required Sinhala-English code-switched textual data for training a Rasa conversational AI to develop, integrate, and test DIME explanations.

2. Develop a novel XAI approach to calculate the GFI for individual tokens of a specified text data instance and utilize that as the feature selection step to calculate LFI.

3. Sift only the significant tokens based on the GFI scores, calculate the LFI of the selected features, and introduce the final score as the Dual Feature Importance (DFI) Score.

4. Compare and contrast different cost metrics for feature importance calculation and integrate the most suitable cost metric with the DIME core codebase.

5. Implement the DIME Command Line Interface (CLI), a fully Rasa-supported version of DIME that enables explanation generation in the DIET classifier utilized in the Rasa framework in CLI environments. Develop a visualization technique that works in CLI environments to display the explanations with the

contribution percentages towards the predicted intent in a human-interpretable manner.

6. Implement the DIME developer console, a fully Rasa-supported Graphical User Interface (GUI) version of DIME that enables explanation generation in the DIET classifier utilized in the Rasa framework in GUI environments. Develop a visualization technique that works in GUI environments to display the explanations with the contribution percentages towards the predicted intent in a human-interpretable manner.

7. Build and release DIME as a modular open-source Python package that can explain any Rasa chatbot model. Test the published Python package functionality on various operating system (OS) platforms, including Windows, Mac OS, and Linux.

8. Design test cases and conduct unit testing, integration testing, system testing, performance testing, acceptance testing, performance testing, and security testing on the DIME package parallel to development (test-driven development).

9. Integrate the DIME core and server packages with Kolloqe, the chatbot development framework introduced as the overall research outcome, seamlessly, and test the integrated functionality.

10. Build and host an official documentation website for the published DIME-XAI Python package and integrate the documentation website content with the Kolloqe documentation website: the documentation website of the integrated solution.

4. METHODOLOGY

4.1 Data Collection

Overall research required two datasets in total. The datasets are domain-specific and comprise Sinhala-English code-switched text data. The two datasets are notably different, and subsections 4.1.1 and 4.1.2 clearly explain these differences. Both datasets utilized data augmentation techniques mainly to overcome the low-resource nature of Sinhala text data gathered and captured as many code-switched phrases and distinct writing patterns as possible to reduce the biases. All research group members generated four versions of the samples in both datasets according to different code-switching styles. Duplicate data removal assisted in ensuring the quality of the data collected.

This individual research component only utilized the domain-specific dataset mentioned in subsection 4.1.2 to train a Rasa conversational AI model since developing DIME only required a fully trained Rasa model and its training dataset.

4.1.1 General dataset for machine learning model training

The first domain-specific dataset for ML model training and NLP text pre-processing tool development contains scraped Sinhala-English code-switched textual data from websites related to SLIIT[13] and news articles[14]. In addition to that, the dataset also comprises publicly available documents, such as Portable Document Format (PDF) and Microsoft Word documents, taken from the same sources since they are both public and official. The authors utilized data augmentation techniques to overcome the low-resource issue of the collected text corpus. The overall dataset comprises 720 paragraphs after data augmentation and cleaning, and it is publicly

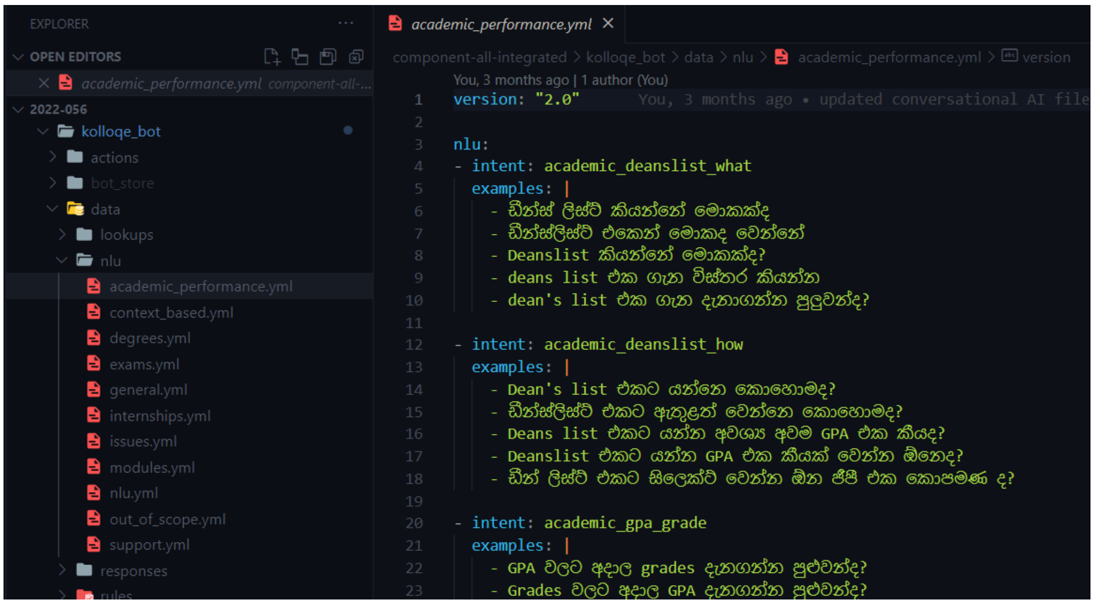
[13] SLIIT related websites referred are https://support.sliit.lk/, https://sliitinternational.lk/, https://www.sliit.lk/, https://www.cahm.lk/, https://library.sliit.lk/, and http://sliit.lk/blog

[14] News articles related to SLIIT and educational domain were extracted from https://www.sundaytimes.lk/, http://www.dailynews.lk, and http://www.sundayobserver.lk/,

available as a public GitHub repository where interested parties can clone and utilize it for future research and other related academic work[15].

4.1.2 Domain-specific dataset for conversational AI training

The second domain-specific dataset for training the Rasa-based conversational AI contains handcrafted Sinhala-English code-switched textual data based on the first dataset. The authors designed the dataset carefully as a training dataset for the intent classification task of conversational AI according to the guidelines provided by Rasa[16]. As in the previous case, the second dataset also utilized data augmentation techniques to overcome the low-resource issue and capture various sentence patterns. There were around seventy-eight intents (78 classes) and one thousand and seven hundred examples (1700 data points), and each class initially contained a minimum of ten question examples as a standard. The authors refined the original chatbot training dataset and created a new training dataset that comprises seventy-one intents (71 classes) and six hundred and thirty-five examples (635 data points). Note that the chatbot developers can later adjust the number of question examples per intent to increase the overall performance of conversational AI at any moment. The chatbot



```
version: "2.0"      You, 3 months ago • updated conversational AI file

nlu:
- intent: academic_deanslist_what
  examples: |
    - ඩීන්ස් ලිස්ට් කියන්නේ මොකක්ද
    - ඩීන්ස්ලිස්ට් එකෙන් මොකද වෙන්නේ
    - Deanslist කියන්නේ මොකක්ද?
    - deans list එක ගැන විස්තර කියන්න
    - dean's list එක ගැන දැනාගන්න පුලුවන්ද?

- intent: academic_deanslist_how
  examples: |
    - Dean's list එකට යන්නෙ කොහොමද?
    - ඩීන්ස්ලිස්ට් එකට ඇතුළත් වෙන්නෙ කොහොමද?
    - Deans list එකට යන්න අවශ්‍ය අවම GPA එක කීයද?
    - Deanslist එකට යන්න GPA එක කීයක් වෙන්න ඕනෙද?
    - ඩීන් ලිස්ට් එකට සිලෙක්ට් වෙන්න ඕන ජීපී එක කොපමණ ද?

- intent: academic_gpa_grade
  examples: |
    - GPA වලට අදාල grades දැනගන්න පුළුවන්ද?
    - Grades වලට අදාල GPA දැනගන්න පුළුවන්ද?
```

Figure 4.1.2.1. A part of the prepared Rasa conversational AI training dataset

[15] The dataset is publicly available under MIT license at https://github.com/kolloqe/datasets
[16] Rasa dataset formatting guidelines are available at https://rasa.com/docs/rasa/training-data-format/

training dataset also resides in the same public GitHub repository mentioned in section 4.1.1, where interested parties can clone and utilize it for future research, other related academic work, or domain-specific chatbot training.

4.2 Functional and Non-functional Requirements

The requirements-gathering phase mainly focused on studying existing research on XAI concepts for NLP and the necessity of an XAI component for machine learning-based products, especially chatbot frameworks such as Rasa. After identifying the current requirement for an XAI component to interpret machine learning models used in modern conversational AI frameworks, the authors conducted the survey mentioned in section 2, which delivered the following insights regarding this research component.

1. The percentage of users who are aware of the decision-making process in chatbots.
2. The proportion of users who are aware of explainable AI and its importance.

Based on the survey responses and analyzed data, the author of this research component derived the following functional and non-functional requirements expected from the DIME XAI approach.

4.2.1 Functional requirements

The functional requirements of the XAI approach, DIME, are as follows.

1. The proposed XAI approach DIME should logically calculate the GFI for any given fully trained DIET classifier in any Rasa chatbot.
2. DIME should utilize GFI scores as a feature selection technique and only select globally valuable features to calculate LFI.
3. DIME should support any ML text classification model that outputs confidence scores.
4. DIME should be able to drastically reduce computational time by allowing users to limit the number of words included in a DIME explanation.

5. DIME should visualize generated model explanations in a human-interpretable manner.
6. DIME should be seamlessly combinable with any Rasa conversational AI assistant project to generate explanations for non-technical users.
7. The generated DIME explanations should be easily sharable and visualizable.

4.2.2 Non-functional requirements

The non-functional requirements of DIME are as follows.

1. DIME should perform feature importance calculations efficiently.
2. DIME should provide reliable and consistent model explanations.
3. DIME should provide simple and easy-to-interpret model explanation visualizations.
4. DIME implementation for Rasa should be modular.

The overall outcome of this research component has successfully addressed all functional and non-functional requirements stated above, and the upcoming sections elaborate on how exactly the DIME implementation addresses these requirements and accomplishes the objectives.

4.3 Individual Component Architecture and Overview

The implementation of the DIME XAI approach developed in this research component comprises several sub-components, as depicted in Figure 4.3.1, the high-level architectural diagram of DIME. The gist of the DIME Python package has five logical sub-packages: CLI, Server, Shared, Utils, and Core. The CLI package contains the DIME CLI explainer logic, which enables users to generate and visualize explanations all in the CLI itself. The Server package is responsible for serving a production-ready DIME developer console GUI and comprises the server and frontend build files. The front end is fully React-based. The Shared package contains constants and DIME-specific custom exception classes that the other packages require. The Utils package contains I/O (Input/ Output) and other widely utilized utility functions. The

Core package is the most noteworthy and comprises the core logic of calculating GFI and LFI.

The developers can request DIME explanations by inputting a textual data instance to either the DIME developer console GUI or via the DIME CLI interface. The Core package of DIME captures the input data instance and tokenizes it using a Whitespace Tokenizer: an NLP tool that splits a given text instance by the whitespace characters. DIME calculates the GFI for only the unique words in the tokenized word list and ranks the words based on the descending order of GFI scores. The developers can also pre-configure DIME to select only a fixed number of the most important words (introduced as the *Ranking Length*). Words with zero or lower feature importance at the global level get pruned, which means that DIME does not include them in the LFI calculation. DIME then calculates the LFI for only the sifted words, generates an explanation JSON (JavaScript Object Notation) object, and returns it. DIME CLI and GUI interfaces can visualize the importance scores as easily comprehensible bar charts and raw scores based on the content of explanation objects returned. The developer console allows developers to visualize, download, and upload explanation files efficiently, making it easy to share explanation objects without regenerating them recursively. The following sections of the dissertation clearly explain the explanation generation mechanism utilized in DIME.

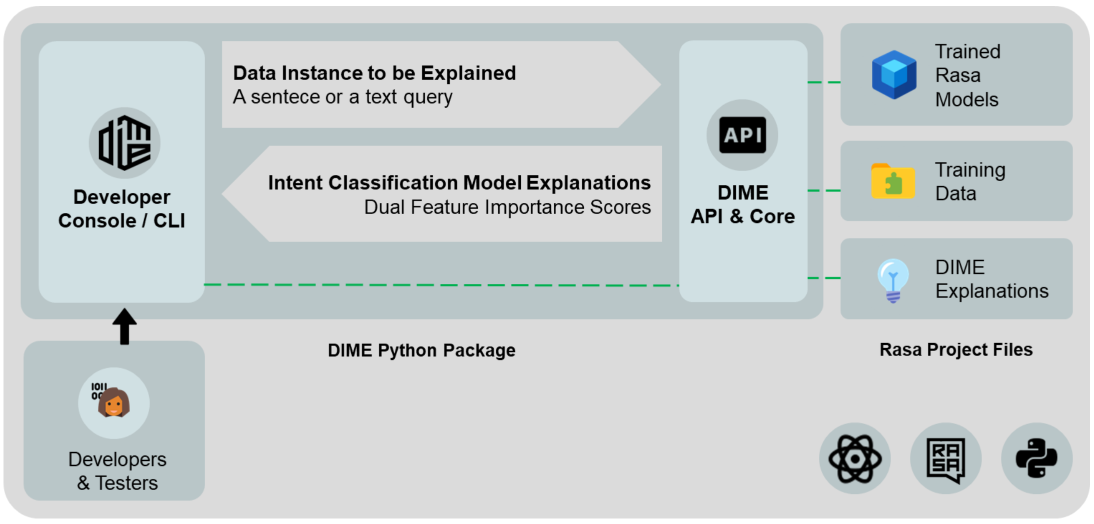


Figure 4.3.1. High-level architectural diagram of DIME

Figure 4.3.2 depicts how DIME fits into the overall system architecture after integrating all research components. The DIME XAI integrated into Kolloqe allows developers to debug ML models built using the developer console of the Kolloqe chatbot development framework and increases the trustworthiness and reliability of the framework.

4.4 Generating Model Explanations

The DIME XAI approach primarily relies on feature importance. For any ML or deep learning model that can output prediction probabilities, DIME can calculate and visualize a combined score derived from GFI and LFI (introduced as DFI since DIME combines GFI and LFI) based on the prediction probabilities given by the model, making DIME model-agnostic. Frameworks such as Rasa outputs intent classification probabilities as confidence scores. When DIET: the default intent classifier in Rasa, classifies an intent for a specified user query, DIET calculates and picks the intent type with the highest confidence score as the intent type of the user query. Non-DIET models select the class with the highest prediction probability when classifying a specified text input. DIME calculates the feature importance based on the total confidence drop, and sections 4.4.2, 4.4.3, and 4.4.4 clearly explain how DIME calculates GFI, LFI and DFI by combining the GFI and LFI in detail.

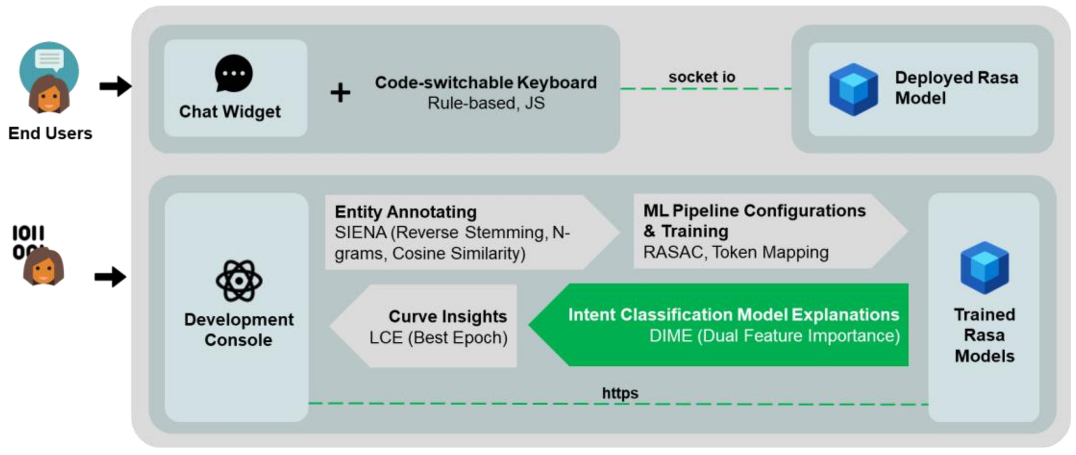
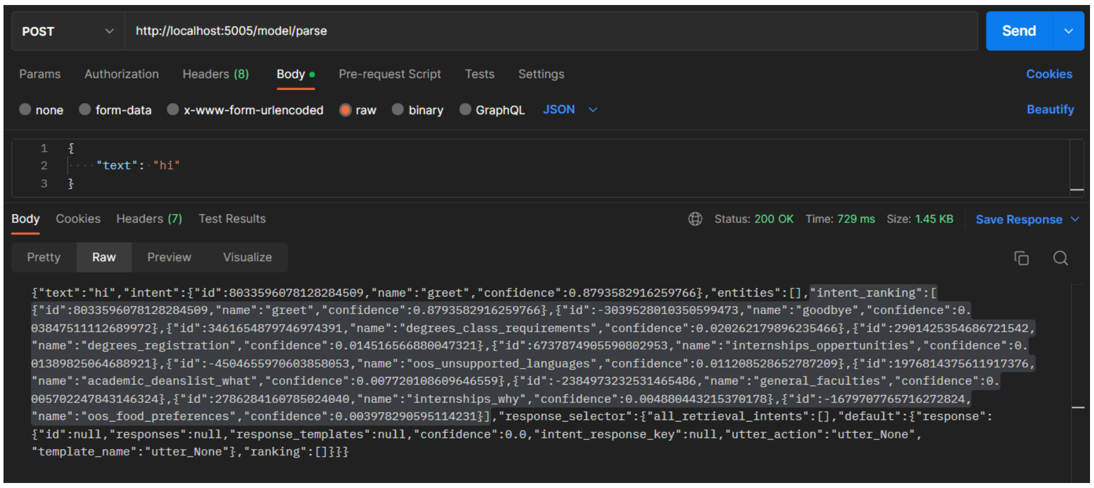


Figure 4.3.2. High-level architectural diagram of Kolloqe, the overall research, and indicated in green is where the DIME XAI component conforms

4.4.1 Training a Rasa conversational AI model

Although the DIME XAI approach is theoretically model-agnostic (section 4.4 clearly explains the model-agnostic nature of DIME), this research component focuses on implementing the DIME approach to explain DIET, the transformer-based deep learning intent classification model introduced and utilized by the Rasa chatbot framework, based on the scope of the research component and the overall research. As depicted in Figure 4.3.1, developing DIME required a Rasa chatbot trained on the training dataset specified in section 4.1.2, with DIET selected as the intent classification model. The source code of the DIET classifier[17] only outputs confidence scores (prediction probabilities) of the first ten intents (classes) with the highest confidence by default. However, DIET needs to output confidence scores for all intent types to implement the DIME approach. Thus, before training the Rasa chatbot, the DIET classifier source code required a few changes to precisely output confidence scores for all intents[18]. The modified DIET classifier was then attached to the Rasa NLU pipeline before training the chatbot model. Figure 4.4.1.1 shows an authentic Rasa output, while Figure 4.4.1.2 shows a Rasa output after the source code



```
http://localhost:5005/model/parse
```

```
{
    "text": "hi"
}
```

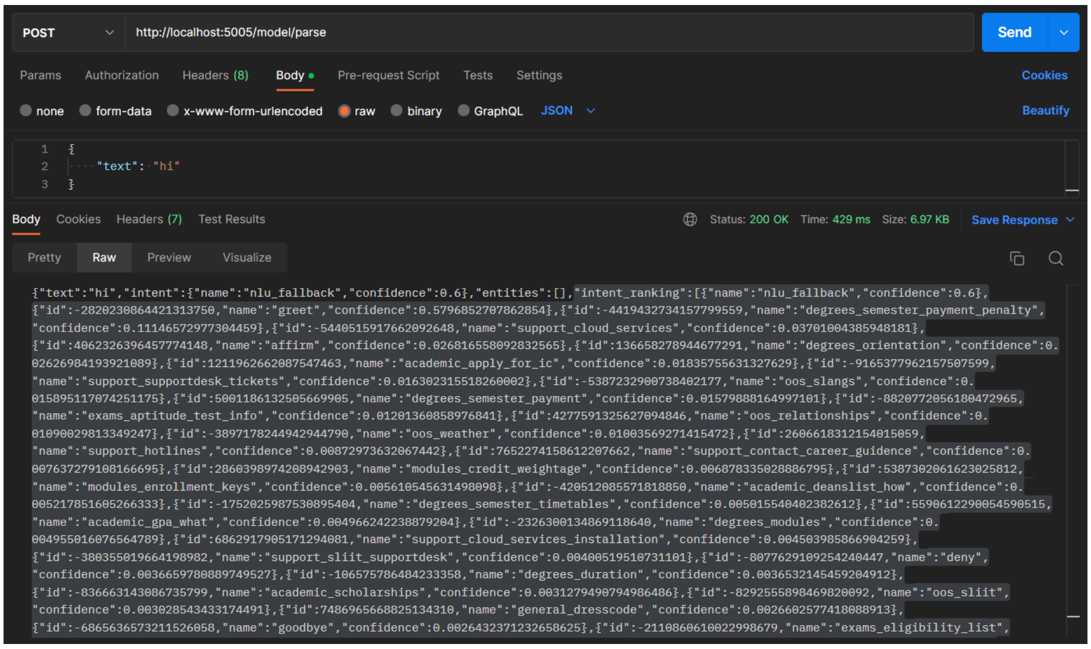
```
Status: 200 OK   Time: 729 ms   Size: 1.45 KB
```

```
{"text":"hi","intent":{"id":8033596078128284509,"name":"greet","confidence":0.8793582916259766},"entities":[],"intent_ranking":[
{"id":8033596078128284509,"name":"greet","confidence":0.8793582916259766},{"id":-3039528010350599473,"name":"goodbye","confidence":0.
03847511112689972},{"id":3461654879746974391,"name":"degrees_class_requirements","confidence":0.020262179896235466},{"id":2901425354686721542,
"name":"degrees_registration","confidence":0.014516566880047321},{"id":6737874905590802953,"name":"internships_oppertunities","confidence":0.
01389825064688921},{"id":-4504655970603858053,"name":"oos_unsupported_languages","confidence":0.011208528652787209},{"id":1976814375611917376,
"name":"academic_deanslist_what","confidence":0.007720108609646559},{"id":-2384973232531465486,"name":"general_faculties","confidence":0.
005702247843146324},{"id":2786284160785024040,"name":"internships_why","confidence":0.004880443215370178},{"id":-1679707765716272824,
"name":"oos_food_preferences","confidence":0.003978290595114231}],"response_selector":{"all_retrieval_intents":[],"default":{"response":
{"id":null,"responses":null,"response_templates":null,"confidence":0.0,"intent_response_key":null,"utter_action":"utter_None",
"template_name":"utter_None"},"ranking":[]}}}
```

Figure 4.4.1.1. Authentic Rasa output with confidence score for only ten intents

[17] https://github.com/RasaHQ/rasa/blob/2.8.x/rasa/nlu/classifiers/diet_classifier.py
[18] Rasa is licensed under Apache 2.0, making it possible to edit the source code without any explicit authorization from Rasa HQ. https://github.com/RasaHQ/rasa/blob/2.8.x/LICENSE.txt

modifications. Figure 4.4.1.3 manifests the exact source code changes done to the Rasa DIET classifier.



POST http://localhost:5005/model/parse

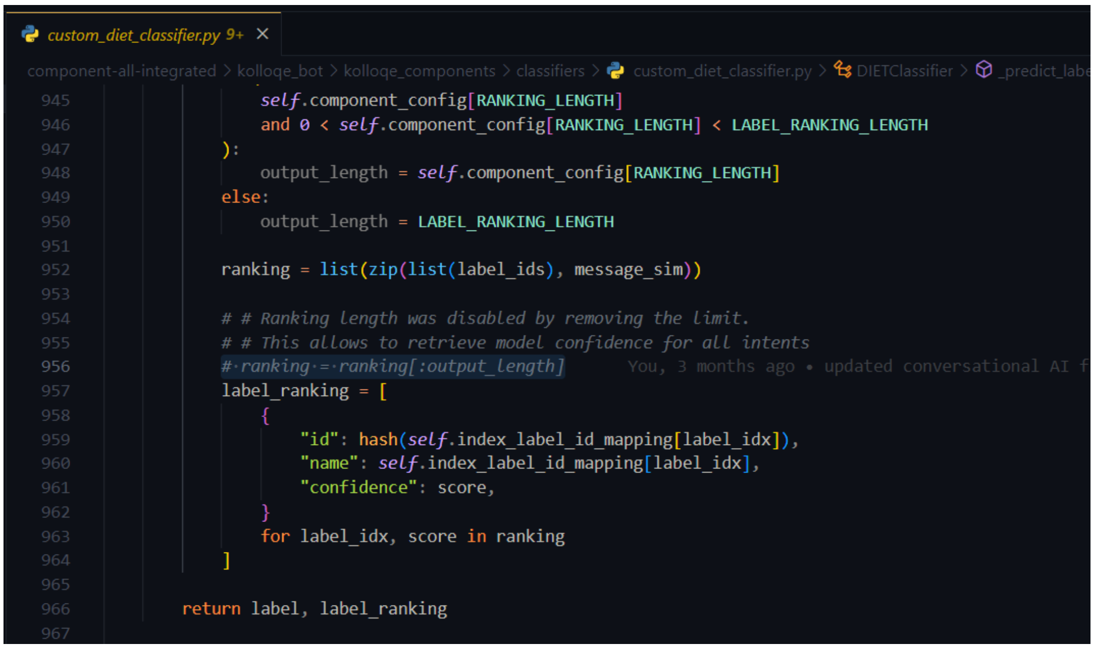
Body (raw, JSON):

```
{
    "text": "hi"
}
```

Status: 200 OK Time: 429 ms Size: 6.97 KB

```
{"text":"hi","intent":{"name":"nlu_fallback","confidence":0.6},"entities":[],"intent_ranking":[{"name":"nlu_fallback","confidence":0.6},
{"id":-2820230864421313750,"name":"greet","confidence":0.5796852707862854},{"id":-4419432734157799559,"name":"degrees_semester_payment_penalty",
"confidence":0.11146572977304459},{"id":-5440515917662092648,"name":"support_cloud_services","confidence":0.03701004385948181},
{"id":4062326396457774148,"name":"affirm","confidence":0.026816558092832565},{"id":136658278944677291,"name":"degrees_orientation","confidence":0.
02626984193921089},{"id":1211962662087547463,"name":"academic_apply_for_ic","confidence":0.01835755631327629},{"id":-9165377962157507599,
"name":"support_supportdesk_tickets","confidence":0.016302315518260002},{"id":-5387232900738402177,"name":"oos_slangs","confidence":0.
015895117074251175},{"id":5001186132505669905,"name":"degrees_semester_payment","confidence":0.01579888164997101},{"id":-8820772056180472965,
"name":"exams_aptitude_test_info","confidence":0.01201360858976841},{"id":4277591325627094846,"name":"oos_relationships","confidence":0.
01090029813349247},{"id":-3897178244942944790,"name":"oos_weather","confidence":0.01003569271415472},{"id":2606618312154015059,
"name":"support_hotlines","confidence":0.00872973632067442},{"id":7652274158612207662,"name":"support_contact_career_guidence","confidence":0.
007637279108166695},{"id":2860398974208942903,"name":"modules_credit_weightage","confidence":0.006878335028886795},{"id":5387302061623025812,
"name":"modules_enrollment_keys","confidence":0.005610545631498098},{"id":-420512085571818850,"name":"academic_deanslist_how","confidence":0.
005217851605266333},{"id":-1752025987530895404,"name":"degrees_semester_timetables","confidence":0.005015540402382612},{"id":5590612290054590515,
"name":"academic_gpa_what","confidence":0.004966242238879204},{"id":-2326300134869118640,"name":"degrees_modules","confidence":0.
004955016076564789},{"id":6862917905171294081,"name":"support_cloud_services_installation","confidence":0.004503985866904259},
{"id":-380355019664198982,"name":"support_sliit_supportdesk","confidence":0.00400519510731101},{"id":-8077629109254240447,"name":"deny",
"confidence":0.0036659780889749527},{"id":-106575786484233358,"name":"degrees_duration","confidence":0.0036532145459204912},
{"id":-836663143086735799,"name":"academic_scholarships","confidence":0.0031279490794986486},{"id":-8292555898469820092,"name":"oos_sliit",
"confidence":0.003028543433174491},{"id":7486965668825134310,"name":"general_dresscode","confidence":0.0026602577418088913},
{"id":-6865636573211526058,"name":"goodbye","confidence":0.0026432371232658625},{"id":-2110860610022998679,"name":"exams_eligibility_list",
```

Figure 4.4.1.2. Rasa output for all 70 intents after DIET source code modifications



custom_diet_classifier.py 9+

component-all-integrated > kolloqe_bot > kolloqe_components > classifiers > custom_diet_classifier.py > DIETClassifier > _predict_labe

```
945                 self.component_config[RANKING_LENGTH]
946                 and 0 < self.component_config[RANKING_LENGTH] < LABEL_RANKING_LENGTH
947             ):
948                 output_length = self.component_config[RANKING_LENGTH]
949             else:
950                 output_length = LABEL_RANKING_LENGTH
951 
952             ranking = list(zip(list(label_ids), message_sim))
953 
954             # # Ranking length was disabled by removing the limit.
955             # # This allows to retrieve model confidence for all intents
956             # ranking = ranking[:output_length]        You, 3 months ago • updated conversational AI f
957             label_ranking = [
958                 {
959                     "id": hash(self.index_label_id_mapping[label_idx]),
960                     "name": self.index_label_id_mapping[label_idx],
961                     "confidence": score,
962                 }
963                 for label_idx, score in ranking
964             ]
965 
966         return label, label_ranking
967 
```

Figure 4.4.1.3. DIET classifier source code changes (line 956 was commented)

4.4.2 Calculating global feature importance

As previously mentioned, DIME calculates the GFI of a specified word by removing the word from the entire training dataset and aggregating the confidence score difference across all data instances. Figure 4.4.2.1 depicts the high-level flow of GFI calculation in DIME. For example, to calculate the GFI of the word *degree*, first, DIME removes it from all data instances in the training dataset of the chatbot model and creates a modified dataset. Then, DIME loads the unmodified dataset, trained Rasa model, and finds the confidence score of the relevant intent type for each data instance in the dataset. Then the algorithm loads the modified dataset, parses individual data instances using the trained Rasa model, and finds the new confidence score for the relevant intent type for each data instance. Finally, the algorithm iterates through the confidence scores of the data instances and constructs an array of confidence score differences between unmodified and modified datasets. Then, DIME assigns the aggregated confidence score differences as the feature importance score of the word *degree* at the global level. This method is known as *permutation feature importance* calculation, and equation (1) clearly states that GFI is the aggregated confidence

$$global\ feature\ importance\ of\ feature\ f = \sum_{i=1}^{n} \Delta c \tag{1}$$

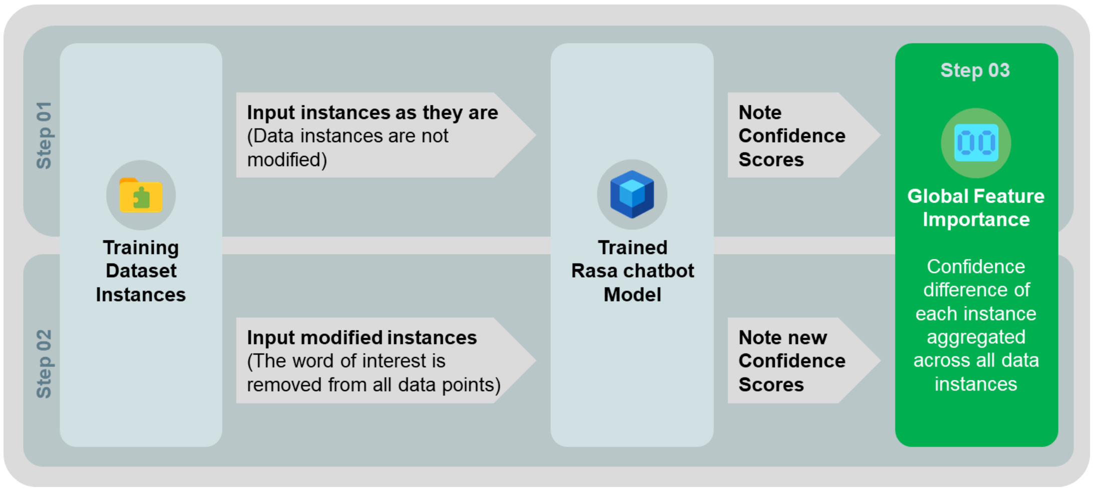

Step 01
Step 02
Training Dataset Instances
Input instances as they are (Data instances are not modified)
Input modified instances (The word of interest is removed from all data points)
Trained Rasa chatbot Model
Note Confidence Scores
Note new Confidence Scores
Step 03
Global Feature Importance
Confidence difference of each instance aggregated across all data instances

Figure 4.4.2.1. The high-level flow of global feature importance calculation in DIME

difference Δc across all data instances, where n is the total number of data instances present in the dataset.

Algorithm 1 demonstrates how DIME calculates the GFI for a given feature f and a training dataset d that consists of a list of training data instances and relevant intent types. The Parse method mentioned in the Algorithm 1 is responsible for retrieving the confidence score for individual data instances of unmodified and modified datasets, $p1$ and $p2$, from the loaded Rasa conversational AI model. Rasa provides a Python API that lets the developers load Rasa NLU models using the Rasa Interpreter for retrieving the confidence scores. Steps 5 to 6 of Algorithm 1 indicate how DIME loops through the dataset, calculate the confidence score difference and aggregates confidence score differences.

The $RemoveFeature$ method defined in Algorithm 1 is responsible for removing a specified feature (word) f from a provided dataset d. Algorithm 2 reveals the $RemoveFeature$ method steps in detail and indicates how DIME removes a specific word from the whole dataset for calculating the GFI. The algorithm iterates through

Algorithm 1 GLOBALFEATUREIMPORTANCE(f, d)

```
1   answer ← 0
2   $p1$ ← PARSE($d$)                         ▷ Unmodified instances confidence scores
3   $p2$ ← PARSE(REMOVEFEATURE($f$, $d$))     ▷ Modified instances confidence scores
4   $n$ ← LENGTH ($d$)                        ▷ Iterate for all data instances
5   for index ∈ {0,…, $n$} do
6           $\Delta c$ ← SUBTRACT($p1$[index], $p2$[index])
7           answer ← ADD(answer, $\Delta c$)
8   return answer
```

Algorithm 2 REMOVEFEATURE(f, d)

```
1   answer ← ⟨ ⟩
2   $n$ ← LENGTH ($d$)                                 ▷ Iterate for all data instances
3   for index ∈ {0,…, $n$} do
4           answer[index] ← REPLACE($d$[index], $f$, ‘’)  ▷ Replace feature with noise or
                                                         remove the feature entirely
5   return answer
```

the entire dataset and replaces all instances of the word from all data instances. Current DIME implementation utilizes a simple string replacement function coupled with regular expressions to achieve that.

4.4.3 Calculating local feature importance

DIME utilizes a similar approach to GFI calculation when calculating LFI. It computes the LFI of a specific feature by inspecting the confidence score given by the Rasa model for the initial data instance and memorizes the predicted intent type. DIME then removes all feature occurrences from the specified text data instance and inspects the confidence score for the same intent type that the model predicted earlier for the initial unmodified data instance. Finally, the algorithm computes the confidence score difference, Δc, by subtracting the confidence score of the modified data instance from the initial data instance confidence for the initially predicted intent type. DIME assigns

$$local\ feature\ importance = \Delta c \tag{2}$$

Δc as the LFI of the specified feature. Figure 4.4.3.1 illustrates the high-level flow of LFI calculation in DIME. Equation (2) clearly states LFI is the confidence score difference, Δc.

Algorithm 3 reveals how DIME calculates LFI for a given feature f and a single text data instance i. The algorithm utilizes $Parse$ and $RemoveFeature$ functions to

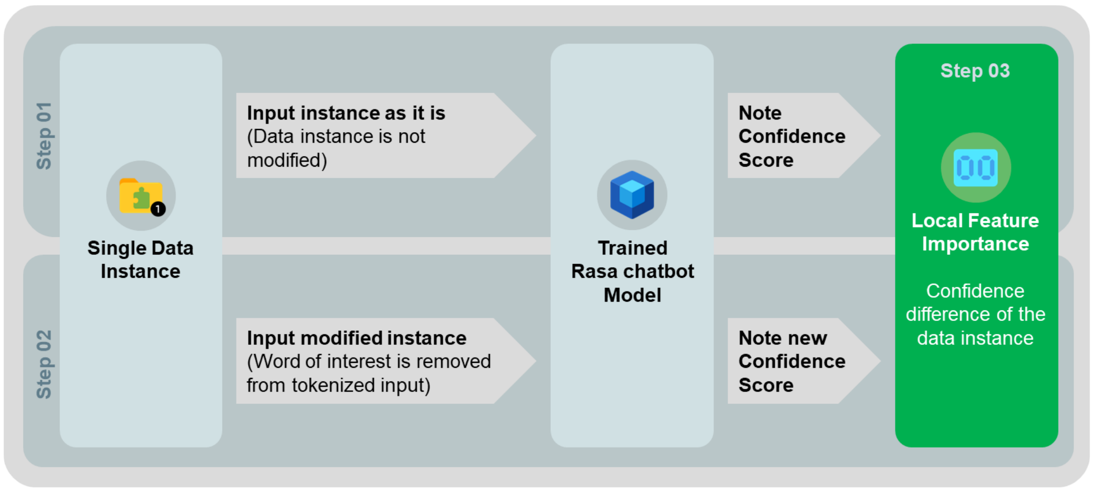

Step 01
Step 02
Single Data Instance
Input instance as it is (Data instance is not modified)
Input modified instance (Word of interest is removed from tokenized input)
Trained Rasa chatbot Model
Note Confidence Score
Note new Confidence Score
Step 03
Local Feature Importance
Confidence difference of the data instance

Figure 4.4.3.1. The high-level flow of local feature importance calculation in DIME

Algorithm 3 LOCALFEATUREIMPORTANCE(f, i)

```
1  $c1 \leftarrow$ PARSE($i$)                          ▷ Unmodified instance confidence score
2  $c2 \leftarrow$ PARSE(REMOVEFEATURE($f$, $i$))      ▷ Modified instance confidence score
3  $\Delta c \leftarrow$ SUBTRACT($c1$, $c2$)          ▷ Confidence score difference
4  **return** $\Delta c$
```

get the Rasa model output for a specified data instance and remove a given feature f from the data instance i, respectively. Then, the algorithm calculates the confidence score difference Δc by subtracting the confidence score of the modified data instance $c2$ from the confidence score of the unmodified data instance $c1$ and assigns that as the LFI of feature f.

4.4.4 Calculating dual feature importance

Sections 4.4.2 and 4.4.3 clarify how DIME calculates GFI and LFI for a specified word. However, the DIME XAI approach follows a specific order when calculating GFI and LFI when generating the combined score introduced as DFI. This section clarifies the overall process of how DIME explains a given text data instance by combining GFI and LFI.

The DIME XAI approach allows developers to input any text data instance and first tokenizes the input to find the distinct features (bag of words). The algorithm then iterates through the feature set, calculates the GFI for each word using the logic stated in Algorithm 1, and orders the GFI scores in descending order. The ordered set of GFIs serves as a feature selection layer and passes only the features with a non-zero GFI to the LFI calculation. The feature selection step allows the algorithm to filter out globally unimportant words and include only the most influential words in the generated explanation, allowing DIME explanations to be globally faithful. As calculating feature importance is compute-intensive, DIME also allows the developers to manually specify the maximum number of features to select from the feature selection step to make the calculations further efficient. DIME introduces the adjustable maximum number of features as $ranking\ length$, and if the selected feature set based on the GFI exceeds the user-defined ranking length r, the algorithm selects only the r number of features with the highest GFI for the LFI calculation.

After selecting the most influential set of features, DIME calculates LFI for each sifted word using Algorithm 2 and introduces it as DFI to indicate that the DIME explanations contain only globally significant features. i.e., DFI is the LFI of only the sifted features based on GFI. Finally, the algorithm converts raw DFI scores into probability values by dividing the DFI of each sifted word by the aggregated DFI across all selected words according to equation (3), where n is the total number of sifted features.

$$DFI\ probability\ of\ feature\ f = \frac{DFI_f}{\sum_{i=0}^{n} DFI_i} \tag{3}$$

Algorithm 4 indicates how DIME calculates DFI for a given data instance d and ranking length r, in simple terms. The algorithm outputs an array of arrays where each element is an array that contains a feature and its raw dual feature importance score, respectively. However, the actual implementation done in Python is slightly different and accepts additional parameters such as case sensitivity.

Algorithm 4 DUALFEATUREIMPORTANCE(d, r)

```
1   $answer \leftarrow \langle\ \rangle$                      ▷ Initialize answer to none. This is an array of arrays
2   $selection \leftarrow \langle\ \rangle$                   ▷ Initialize selected features based on
                                                              GFI to none. This is an array of arrays
3   $bow \leftarrow$ UNIQUE(TOKENIZE ($d$))                  ▷ Bag of words from data instance
4   $n \leftarrow$ LENGTH($bow$)
5   for each $f \in bow$ do                                   ▷ Iterate through all words in bow
6       $score \leftarrow \langle\ \rangle$
7       $gfi \leftarrow$ GLOBALFEATUREIMPORTANCE($f, d$)
8       $score \leftarrow$ ADD($f, gfi$)
9       $selection \leftarrow$ ADD($selection, score$)        ▷ Append feature and GFI as an array
                                                              to the selection array
10  if LENGTH($selection$) $> r$ then                         ▷ Slice the selection array at $r$
11      $selection \leftarrow$ SLICE($selection, r$)
12  for $index \in \{0,\ldots,$ LENGTH($selection$)$\}$ do
13      if $selection[index][1] > 0$ then                     ▷ Select features for LFI calculation
                                                              only if GFI is a non-zero value
14          $score \leftarrow \langle\ \rangle$
15          $lfi \leftarrow$ LOCALFEATUREIMPORTANCE($selection[index][0], d$)
16          $score \leftarrow$ ADD($selection[index][0], lfi$)
17          $answer \leftarrow$ ADD($answer, score$)
18  return $answer$
```

4.5 DIME Python Package Implementation

The author of this individual research component has implemented the DIME XAI approach as a modular Python package that developers can integrate with any Rasa chatbot project. Although DIME is theoretically model-agnostic, as previously mentioned, the current Python implementation mainly focuses on explaining the DIET intent classifier utilized by Rasa-based chatbot models as one of the NLU pipeline components. DIME package source code has five logical sub-packages: CLI, Server, Shared, Utils, and Core, and section 4.3 clearly states the purpose of each sub-package in detail. The author has decided to release the individual research work, i.e., the DIME XAI Python package, as an open-source Python package that anyone can get installed simply by running *pip install dime-xai*[19] on any Python 3.7 or 3.8 supported machine. The DIME Python package adheres to semantic versioning[20], and each stable build of the DIME package is available as a new version at Python Package Index (PyPI). The open-source contribution is available under the Apache 2.0 license. However, the Apache 2.0 license demands individuals to clearly state any changes made to the source code if they wish to republish the package, whether they have done an improvement, bug fix, or any other modification.

The open-source DIME-XAI Python package allows Rasa conversational AI developers to enable explainability in their chatbot development projects by simply installing it at the root of any rasa chatbot project. Table 4.5.1 shows the compatibility between DIME-XAI and Rasa Python packages. Developers should carefully pick the correct DIME package that is fully compatible with the Rasa Python package they use. DIME also provides the customized DIET classifier source code explained in detail in

Table 4.5.1. DIME-XAI and Rasa compatibility matrix

DIME-XAI Developer Console Version		**Rasa Open-Source Version**
MAJOR.MINOR	**PATCH (Released Bug fixes)**	
1.2.x	1.2.1, 1.2.0	2.8.8
1.1.x	1.1.3, 1.1.2, 1.1.1	
1.0.x	1.0.1, 1.0.0	
0.0.x	0.0.4a14, 0.0.4a13, 0.0.4a6, 0.0.3a5, 0.0.3a2	

[19] https://pypi.org/project/dime-xai/
[20] https://semver.org/

section 4.4.1, which the developers should attach to the NLU pipeline of the chatbot to enable DIME XAI explanations.

4.5.1 DIME CLI interface

The primary DIME CLI interface is responsible for parsing input arguments and deciding which interface to invoke out of four sub-interfaces, namely, DIME Init, DIME Explain, DIME Server, and DIME Visualize. This section clearly explains the purpose of each sub-interface and the CLI commands to invoke them. Figure 4.5.1.1 depicts all sub-interfaces that DIME CLI provides along with their purpose.

The main DIME CLI entry point is the dime command, which means that typing dime in the CLI after installing the DIME-XAI package will invoke the main DIME CLI interface. Table 4.5.1.1 comprises the optional arguments allowed by the main DIME CLI interface.

Table 4.5.1.1. Main DIME-XAI CLI Interface arguments

CLI Command	**Purpose**
Main Command	
dime	Main entry point of main DIME CLI Interface.
Optional Arguments	
-v or --version	Outputs DIME XAI package version details to CLI.
-h or --help	Provides instructions on how to use dime CLI.

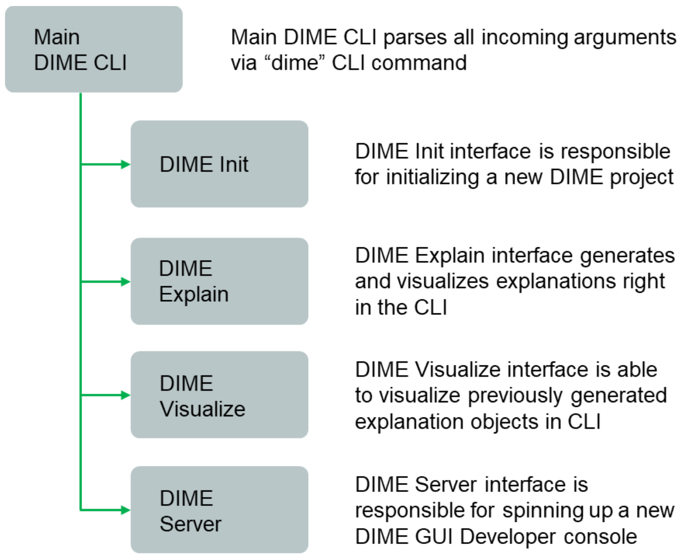

Main DIME CLI
Main DIME CLI parses all incoming arguments via “dime” CLI command
DIME Init
DIME Init interface is responsible for initializing a new DIME project
DIME Explain
DIME Explain interface generates and visualizes explanations right in the CLI
DIME Visualize
DIME Visualize interface is able to visualize previously generated explanation objects in CLI
DIME Server
DIME Server interface is responsible for spinning up a new DIME GUI Developer console

Figure 4.5.1.1. Interfaces offered through main DIME CLI

Table 4.5.1.2 through Table 4.5.1.5 comprises all DIME CLI sub-interfaces, positional arguments, optional arguments, and the exact purpose they serve.

Table 4.5.1.2. DIME Init sub-interface arguments

CLI Command	**Purpose**
Sub-interface Command	
dime init	Initializes a new DIME project in a specified directory.
Optional Arguments	
--debug	Sets the logging level to debug mode from info mode.
--quiet	Initializes a starter dime project without prompting the user for a project location.

Table 4.5.1.3. DIME explain sub-interface arguments

CLI Command	**Purpose**
Sub-interface Command	
dime explain	Runs DIME CLI explainer, a terminal-based explainer tool.
Optional Arguments	
-i or --instance	Specify data instance to explain.
-m or --metric	Specify metric to use for calculating feature importance.
-o or --output	Specify the output mode of the DIME CLI explainer.
--case or --no-case	Preserve or ignore the case sensitivity of the data instance provided. If no case is enabled, data instance is converted to lower-case.
-r or –request-id	If specified, DIME CLI Explainer appends the explanation object to a special explanation process queue, so the requester is able to retrieve the explanation from the queue remotely. (Utilized in DIME Server). Must follow --quiet.
--debug	Sets the logging level to debug mode from info mode.
--quiet	Initializes a starter dime project without prompting the user for a project location

Table 4.5.1.4. DIME server sub-interface arguments

CLI Command	**Purpose**
Sub-interface Command	
dime server	Run DIME server, a web-based visualization tool for DIME.
Optional Arguments	
-p or --port	Specify the DIME server port. The default port is 6066
--debug	Sets the logging level to debug mode from info mode.

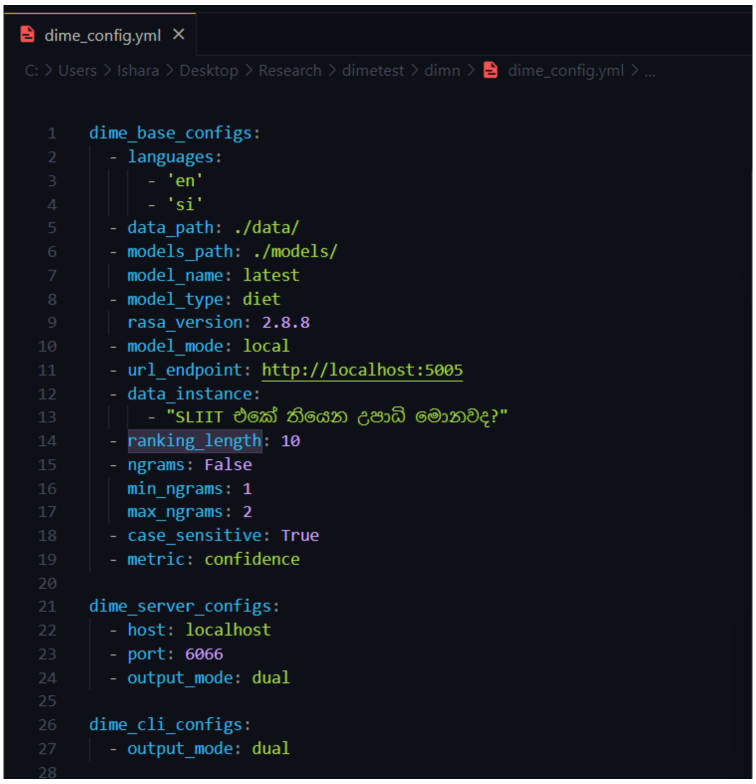
Table 4.5.1.5. DIME visualize sub-interface arguments

CLI Command	Purpose
Sub-interface Command	
dime visualize	Runs DIME CLI visualizer, a terminal-based visualization tool for visualizing already generated DIME explanation objects.
Optional Arguments	
-e or --explanation	Specify the explanation file name for generating visualizations. The file must reside on the dime_explanations directory.
-l or --limit	Limits the number of words in the global explanation visualizations. Not effective for dual feature importance visualizations.
--debug	Sets the logging level to debug mode from info mode.

DIME also includes a Yet Another Markup Language (YAML) configuration file called *dime_config.yml*. Instead of passing configurations as arguments via the CLI, developers can specify all the configurations in the configuration file and run DIME CLI without explicitly specifying any optional arguments. However, the existence of the default configuration file is a must. If dime_config.yml is missing from the project root, DIME CLI interface refuses to run. Developers also can override the configurations mentioned in the configuration file using the CLI. For example, when running the command dime explain *--no-case*, the case_sensitive state mentioned in the *dime_config.yml* is overridden by DIME CLI. Figure 4.5.1.2 depicts the content of the dime_config.yml file with default dime configurations.

4.5.2 DIME project structure

As stated in Table 4.5.1.2, running the command dime init creates a new DIME project. Figure 4.5.2.1 shows the directory structure of a DIME project. If a Rasa project does not exist where *dime init* is running, DIME creates two directories, *data* and *models*, with a default dataset and a trained Rasa model. However, if a Rasa project exists in the directory where *dime init* is running, DIME skips creating the two folders and initializes only the files and directories required by DIME: *dime_cache*, *dime_explanations*, *dime_components*, and *dime_config.yml*. The DIME Init interface utilizes the dime_cache directory to store temporary files and directories when creating a new DIME project and automatically removes them after the initialization is over. dime_components folder contains custom Rasa NLU pipeline components required to enable DIME explanations for Rasa chatbot projects. Currently, the folder only contains the modified DIET classifier source code since it is the only NLU pipeline component required to enable DIME-XAI for any Rasa project.


dime_config.yml
C: > Users > Ishara > Desktop > Research > dimetest > dimn > dime_config.yml > ...

```yaml
dime_base_configs:
  - languages:
      - 'en'
      - 'si'
  - data_path: ./data/
  - models_path: ./models/
    model_name: latest
  - model_type: diet
    rasa_version: 2.8.8
  - model_mode: local
  - url_endpoint: http://localhost:5005
  - data_instance:
      - "SLIIT එකේ තියෙන උපාධි මොනවද?"
  - ranking_length: 10
  - ngrams: False
    min_ngrams: 1
    max_ngrams: 2
  - case_sensitive: True
  - metric: confidence

dime_server_configs:
  - host: localhost
  - port: 6066
  - output_mode: dual

dime_cli_configs:
  - output_mode: dual
```

Figure 4.5.1.2. DIME Configuration file *dime_config.yml* with default configurations

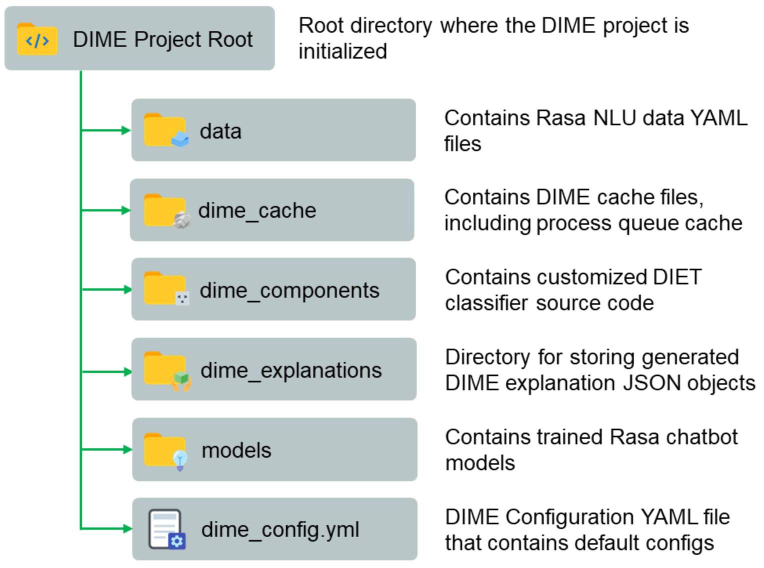

DIME Project Root
Root directory where the DIME project is initialized
data
Contains Rasa NLU data YAML files
dime_cache
Contains DIME cache files, including process queue cache
dime_components
Contains customized DIET classifier source code
dime_explanations
Directory for storing generated DIME explanation JSON objects
models
Contains trained Rasa chatbot models
dime_config.yml
DIME Configuration YAML file that contains default configs

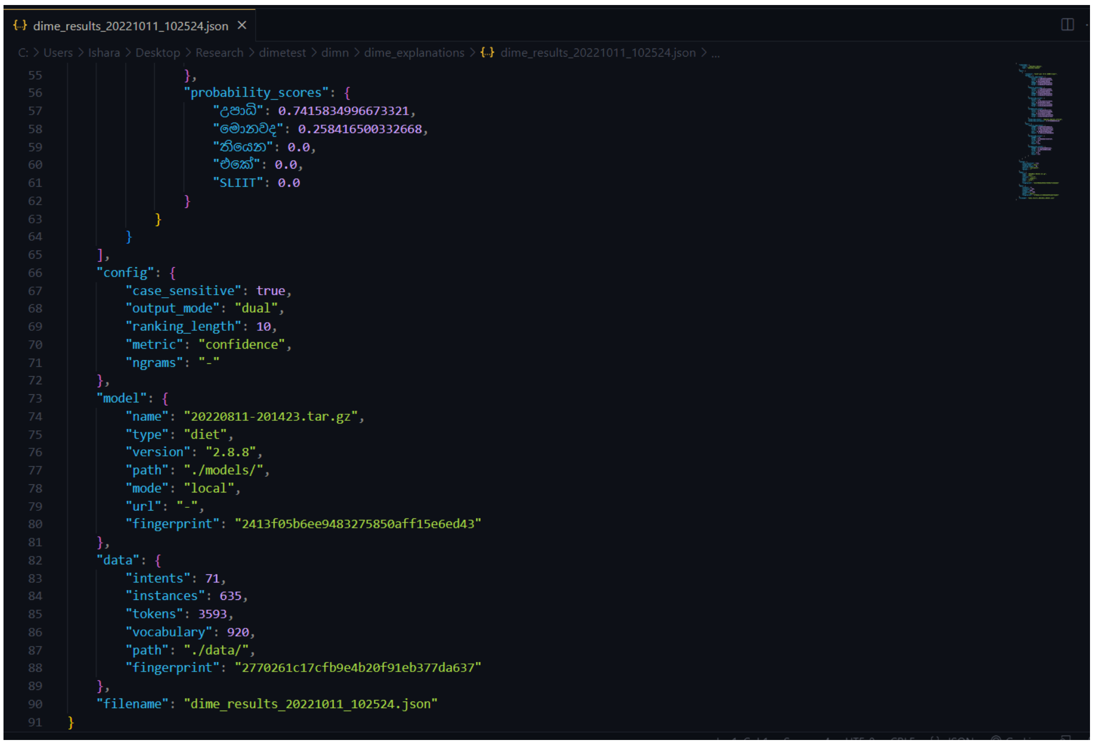
Figure 4.5.2.1. DIME Project Structure

4.5.3 Dime configuration file

As depicted in Figure 4.5.1.2, developers can tweak default DIME configurations using the dime_config.yml file. Table 4.5.3.1 contains all customizable DIME configurations, their default values, and valid options.

4.5.4 DIME explanation objects

As previously explained in section 4.3, DIME creates an explanation object after calculating the DFI that consists of information about the explanation job. An explanation object is a JSON object that contains details about the Rasa model and NLU data utilized, DIME configurations used to generate the explanation, start and end timestamps of the explanation job, raw GFI and DFI scores obtained, and GFI and DFI probabilities. Figure 4.5.4.1 depicts a part of an explanation object. The DIME XAI package automatically persists the explanation objects after the explanation job terminates and utilizes the generated explanation object to render visualizations in both CLI and GUI. DIME delivers portability by allowing users to visualize any valid explanation object right away on any machine where DIME is up and running.


dime_results_20221011_102524.json
C: > Users > Ishara > Desktop > Research > dimetest > dimn > dime_explanations > dime_results_20221011_102524.json > ...

```
                },
                "probability_scores": {
                    "උපාධි": 0.7415834996673321,
                    "මොනවද": 0.258416500332668,
                    "තියෙන": 0.0,
                    "එකේ": 0.0,
                    "SLIIT": 0.0
                }
            }
        }
    ],
    "config": {
        "case_sensitive": true,
        "output_mode": "dual",
        "ranking_length": 10,
        "metric": "confidence",
        "ngrams": "-"
    },
    "model": {
        "name": "20220811-201423.tar.gz",
        "type": "diet",
        "version": "2.8.8",
        "path": "./models/",
        "mode": "local",
        "url": "-",
        "fingerprint": "2413f05b6ee9483275850aff15e6ed43"
    },
    "data": {
        "intents": 71,
        "instances": 635,
        "tokens": 3593,
        "vocabulary": 920,
        "path": "./data/",
        "fingerprint": "2770261c17cfb9e4b20f91eb377da637"
    },
    "filename": "dime_results_20221011_102524.json"
}
```

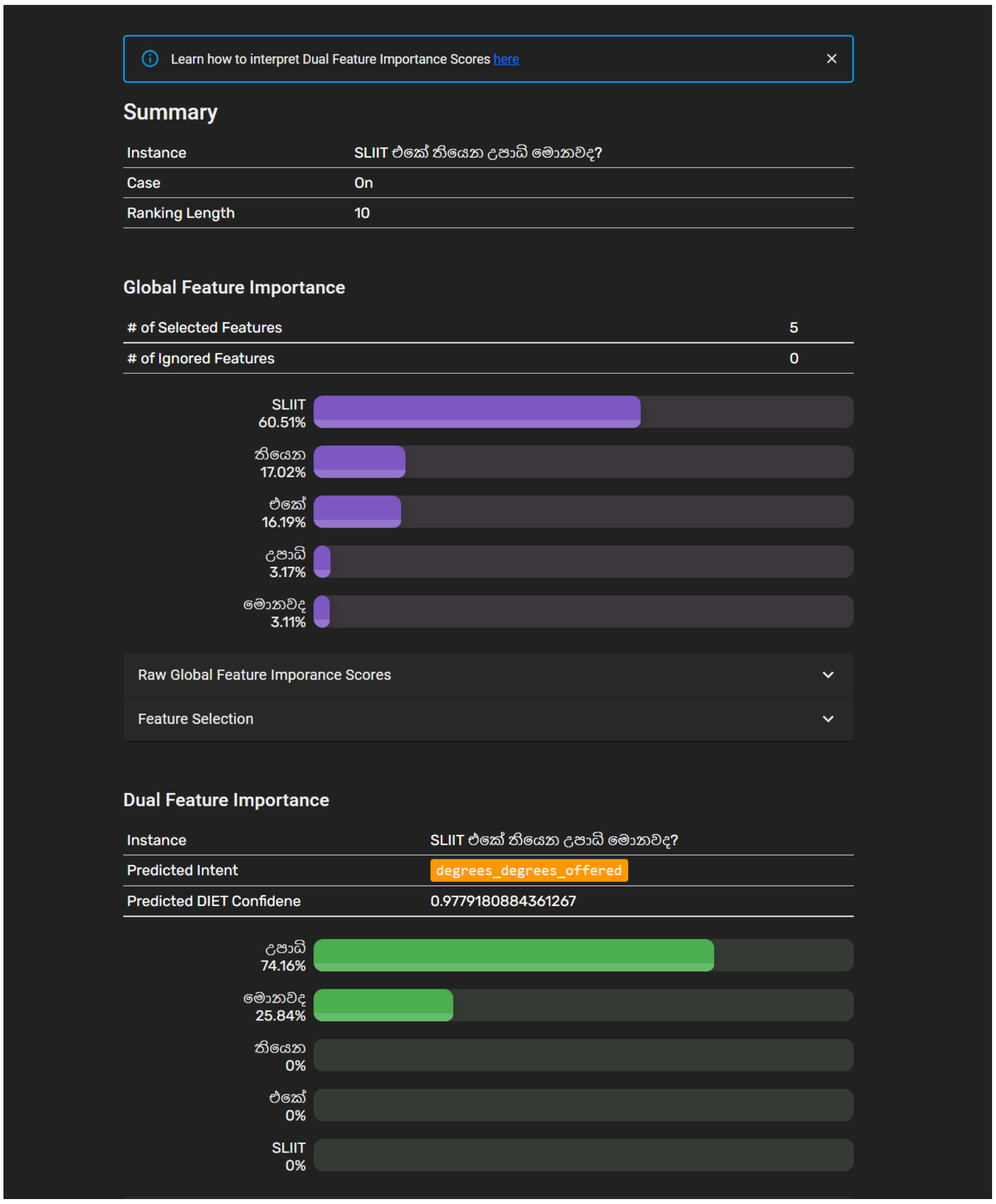
Figure 4.5.4.1. A part of a DIME explanation JSON object

Table 4.5.3.1. DIME configurations, default values, and valid options

Configuration	Value		Description
Base configurations			
languages	*Default value*	[en, si]	Informs DIME about languages present in the training dataset
	Valid values	[en, si], si, en	
data_path	*Default value*	./data	Specifies the Rasa NLU data directory. If a separate dataset is used for XAI, the directory path must be altered
	Valid values	a valid directory path	
models_path	*Default value*	./models	Directory path where Rasa models persist
	Valid values	a valid directory path	
model_name	*Default value*	latest	A valid Rasa model name. Specifying as latest will read the latest rasa model based on the timestamp
	Valid values	a valid rasa model name	
model_type	*Default value*	diet	Model architecture. Custom models are not supported yet
	Valid values	diet, custom	
rasa_version	*Default value*	2.8.8	Locally available Rasa version. Currently supports only 2.8.8
	Valid values	2.8.8	
model_mode	*Default value*	local	**Support for REST models is experimental**
	Valid values	local, rest	
url_endpoint	*Default value*	http://localhost:5005	URL of the REST model where Rasa is running if REST enabled
	Valid values	a valid bot URL	
data_instance	*Default value*	SLIIT එකේ තියෙන උපාධි මොනවද?	A data instance or a list of data instances to explain
	Valid values	a valid string	
ranking_length	*Default value*	10	Number of maximum features to select. DIME recommends 10
	Valid values	a valid integer	
ngrams	***n-grams* and its sub-keys are experimental and have no effect**		
case_sensitive	*Default value*	True	If set to false, data instances are converted to lower case first
	Valid values	True, False	
metric	*Default value*	confidence	Metric for feature importance calculation. DIME recommends *confidence* as accuracy and f1-score generate results that are mostly sparse
	Valid values	confidence, accuracy, f1-score	
Server configurations			
host	*Default value*	localhost	Host name to host the DIME server
	Valid values	localhost, 0.0.0.0, 127.0.0.1	
port	*Default value*	6066	Port to start the DIME server
	Valid values	a valid port number	
output	*Default value*	dual	If set to global, only GFI is visualized, else both GFI and DFI are visualized (valid for both server and cli visualizations)
	Valid values	dual, global	

4.5.5 DIME visualizations

The DIME XAI approach implemented via Python can render visualizations for generated explanations in both CLI and GUI interfaces, enabling portability for DIME explanations. DIME visualizes the probability scores computed for each sifted feature in the CLI using the Python package termgraph. The GUI visualizations incorporate React components to visualize the DFI probability scores in the DIME server. Figure 4.5.5.1 and Figure 4.5.5.2 depict a DIME GUI and CLI visualization, respectively.



Learn how to interpret Dual Feature Importance Scores here

Summary

Instance	SLIIT එකේ තියෙන උපාධි මොනවද?
Case	On
Ranking Length	10

Global Feature Importance

# of Selected Features	5
# of Ignored Features	0

SLIIT	60.51%
තියෙන	17.02%
එකේ	16.19%
උපාධි	3.17%
මොනවද	3.11%

Raw Global Feature Imporance Scores

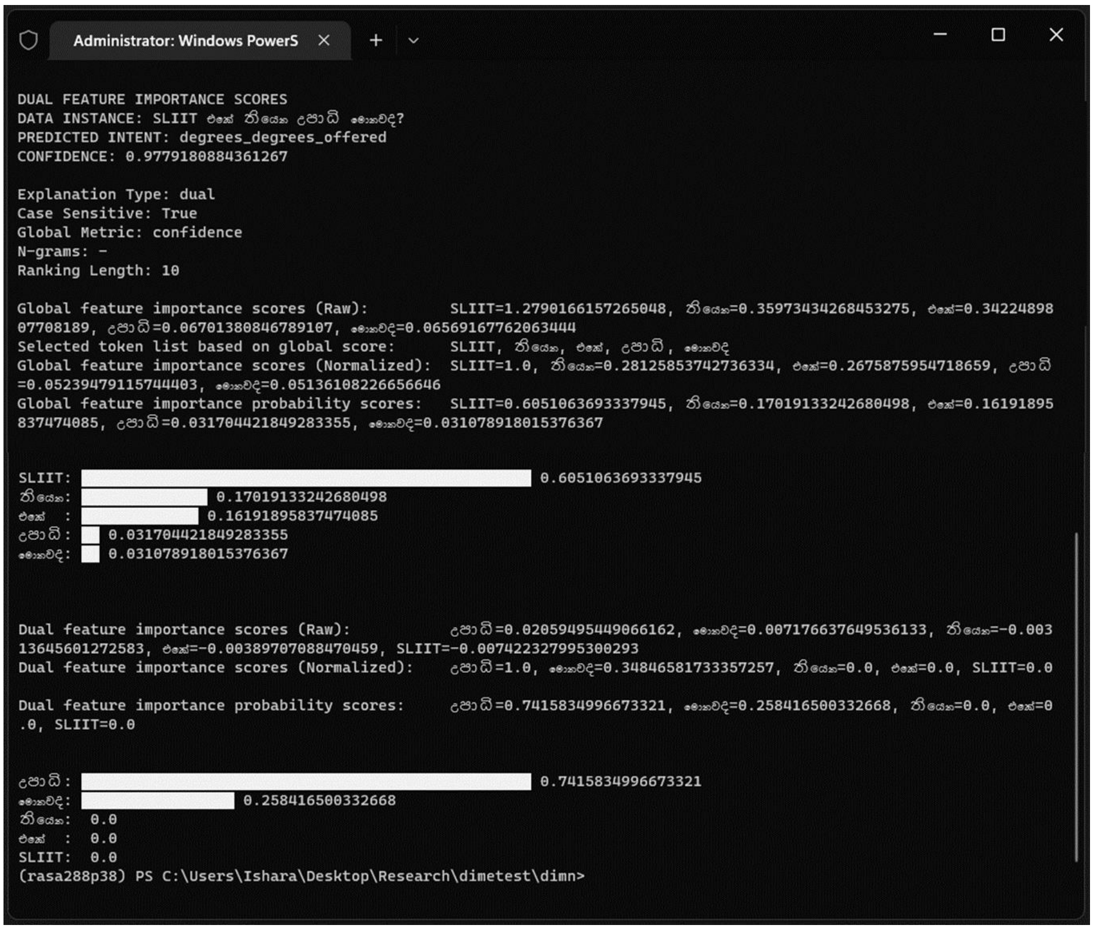
Feature Selection

Dual Feature Importance

Instance	SLIIT එකේ තියෙන උපාධි මොනවද?
Predicted Intent	degrees_degrees_offered
Predicted DIET Confidene	0.9779180884361267

උපාධි	74.16%
මොනවද	25.84%
තියෙන	0%
එකේ	0%
SLIIT	0%

Figure 4.5.5.1. A DIME GUI Visualization



```
DUAL FEATURE IMPORTANCE SCORES
DATA INSTANCE: SLIIT එක් තියෙන උපාධි මොනවද?
PREDICTED INTENT: degrees_degrees_offered
CONFIDENCE: 0.9779180884361267

Explanation Type: dual
Case Sensitive: True
Global Metric: confidence
N-grams: -
Ranking Length: 10

Global feature importance scores (Raw):        SLIIT=1.2790166157265048, තියෙන=0.35973434268453275, එක්=0.34224898
07708189, උපාධි=0.06701380846789107, මොනවද=0.06569167762063444
Selected token list based on global score:     SLIIT, තියෙන, එක්, උපාධි, මොනවද
Global feature importance scores (Normalized):  SLIIT=1.0, තියෙන=0.28125853742736334, එක්=0.2675875954718659, උපාධි
=0.05239479115744403, මොනවද=0.05136108226656646
Global feature importance probability scores:   SLIIT=0.6051063693337945, තියෙන=0.17019133242680498, එක්=0.16191895
837474085, උපාධි=0.031704421849283355, මොනවද=0.031078918015376367


SLIIT: ██████████████████████████████████████████████████ 0.6051063693337945
තියෙන: ██████████████ 0.17019133242680498
එක්  : █████████████ 0.16191895837474085
උපාධි: ██ 0.031704421849283355
මොනවද: ██ 0.031078918015376367



Dual feature importance scores (Raw):          උපාධි=0.02059495449066162, මොනවද=0.007176637649536133, තියෙන=-0.003
13645601272583, එක්=-0.00389707088470459, SLIIT=-0.007422327995300293
Dual feature importance scores (Normalized):    උපාධි=1.0, මොනවද=0.34846581733357257, තියෙන=0.0, එක්=0.0, SLIIT=0.0

Dual feature importance probability scores:     උපාධි=0.7415834996673321, මොනවද=0.258416500332668, තියෙන=0.0, එක්=0
.0, SLIIT=0.0


උපාධි: ██████████████████████████████████████████████████ 0.7415834996673321
මොනවද: █████████████████ 0.258416500332668
තියෙන:  0.0
එක්  :  0.0
SLIIT:  0.0
(rasa288p38) PS C:\Users\Ishara\Desktop\Research\dimetest\dimn>
```

Figure 4.5.5.2. A DIME CLI Visualization

4.5.6 DIME server graphical user interface

DIME server offers an easy-to-use GUI for developers in addition to the DIME CLI interface. The front end is a React app with three interconnected primary user interfaces (UIs), (1) DIME dashboard, (2) DIME Explanations UI, and (3) Models UI. The dashboard interface summarizes the number of models and explanations available and provides required documentation links. Developers can provide a data instance and click the explain button to start an explanation job from the dashboard UI. Once an explanation job finishes, the DIME server renders visualizations at the bottom of the dashboard page and also appends the explanation to the explanation UI for later use. DIME explainer in the backend can handle multiple explanation jobs and uses a process queue to achieve that. Each explanation job consists of a unique request ID, and if users interrupt an ongoing explanation job by clicking the abort button, the same request ID assists the backend in terminating the correct explanation process. DIME shows an already-generated explanations list in the explanations UI, where developers can visualize, download, or delete individual explanations. Users also can easily upload valid explanation JSON objects generated in different projects to quickly render visualizations without having access to the original project, which makes DIME explanations portable. DIME server offers the Peak feature to glance at externally generated explanation objects, similar to the Upload feature, but without permanently uploading them to the server. On the other end, the DIME server supports toggling between dark-mode and light-mode themes to reduce eye strain and glare in low-light environments. See Appendix B: DIME Server User Interfaces for the complete list of DIME server UIs.

4.6 Tools and Technologies

4.6.1 DIME package implementation

The author of this individual research component has utilized Python 3.8[21] as the backend programming language to implement the core logic of the algorithm, the DIME CLI interface, and its sub-interfaces. The Rasa open-source package assisted in training the required chatbot models and testing the DIME explainer functionality. The NumPy Python package assisted in handling numerical operations such as clipping feature importance values at zero and calculating probability scores. The CLI explanation visualizations utilize *termgraph*[22] Python package to generate CLI bar charts.

4.6.2 DIME server implementation

The DIME server and the API are single Flask-based server that allows both local and production-level deployments. The production deployment of the DIME server relies on a Web Server Gateway Interface (WSGI) server, *waitress*[23]. The author has selected the *waitress* server primarily due to its extensive support for Windows and

[21] DIME implementation supports both Python 3.7 and 3.8. This version restriction is mainly caused by the Rasa 2.8.8 which does not support newer Python versions.
[22] https://pypi.org/project/termgraph/
[23] https://pypi.org/project/waitress/

Linux operating systems provided out-of-the-box. Developers can effectively run DIME server in debug mode by running the command *dime server* with the *--debug* flag, which disables the *waitress* production deployment mode and executes the flask server directly.

4.6.3 DIME server front end development

The front end of the DIME server utilizes React as the front-end development framework primarily due to component reusability. The front end is a single page app that uses Node 16.15.0 and utilizes material-ui, axios, uuid, prop-types, react-router, and framer-motion Node Package Manager (NPM) based packages. The entire frontend resides within the flask backend server explained in the section 4.6.2, and does not require a separate deployment.

4.6.4 Python development environment

The author used PyCharm and Visual Studio Code as the primary Integrated Development Environments (IDEs) and Anaconda Python distribution to efficiently manage the Python virtual environments. The DIME package also utilizes *.env* files to hold required environmental variables and credentials and uses *requirements.txt* files to state required Python packages for future reference and deployment.

4.6.5 Source code, version, and release management

The overall research utilizes GitLab as the source code management platform, and the implementation of the individual research component relies on GitHub for the version and release management of the DIME open-source package. Stable releases of the DIME-XAI Python package reside on PyPI, where any developer can download and install DIME-XAI as any other Python package. All releases of DIME-XAI adhere to semantic versioning, making it easy to identify major, minor, and patch releases and any breaking changes.

Table 4.6.5.1 summarizes the tools and technologies stated in all sub-sections of section 4.6.

Table 4.6.5.1. Summary of Tools and Technologies utilized

Task	Tools and technologies utilized
Algorithm Implementation and DIME Python package development	Python 3.8, NumPy, Termgraph, Python Dotenv, Ruamel YAML, tqdm, Rasa 2.8.8, PyCharm, Visual Studio Code, Anaconda Distribution
DIME server frontend implementation	Flask, Flask-Cors, psutil, waitress, Bootstrap, React JS, Axios, UUID4, Material-UI, framer-motion, prop-types, react-router
Process queue implementation	SQLite3
Chatbot development	Rasa 2.8.8, Rasa-SDK 2.8.0
Source code and version management	GitHub, GitLab, semantic-versioning, PyPI
Testing	pytest, Snyk vulnerability assessment tool, Docker

4.7 Commercialization Aspect of the Product

Although the individual research component outcome holds a vast commercial value, the author published the research component outcome as an open-source XAI Python package due to the following reasons.

1. DIME-XAI package is more beneficial as a free and open-source tool than a commercial product to the ML and NLP communities since the number of XAI research freely available is considerably low.

2. Since DIME XAI is a novel approach, there is more potential to identify its flaws, limitations, and drawbacks when a vast audience has access to the research and its initial implementation.

3. The main objective of releasing DIME-XAI as an open-source tool is to accelerate the utilization of XAI in production-ready AI-centric applications to make them trustworthy and reliable.

4. Interested individuals can integrate DIME-XAI into their solutions to explain Rasa-based chatbot models, further develop the tool and add support beyond Rasa DIET models, or utilize the DIME approach in their research, which makes the open-source release of DIME invaluable.

However, the DIME approach adds commercial value to the Kolloqe chatbot development platform, the overall research solution, by enabling explainability for the chatbot models trained using Kolloqe. Kolloqe has the competitive advantage of containing the ability to ensure the trustworthiness and reliability of the chatbot models when compared with currently available widely utilized chatbot development frameworks, such as Google DialogFlow and Rasa, where XAI is not available at all. The following facts explain how DIME increases the overall commercial value of Kolloqe in detail.

1. The DIME XAI approach enables out-of-the-box support for chatbot model reliability verification for models trained using the Kolloqe developer console.

2. The DIME XAI approach provides the Kolloqe developer console with a remarkable and rare competitive advantage over the other chatbot development frameworks and platforms by enabling explainability in a production-grade chatbot development platform for the first time.

3. The DIME approach delivers a solid technique to debug and validate ML models and identify training data issues efficiently.

4. Explanations provide an additional validation step before rolling out a production-level deployment based solely on performance scores such as accuracy or f1-score. Section 1.1 clearly states how relying entirely on a performance score is inadequate for ML models.

5. TESTING, IMPLEMENTATION RESULTS & DISCUSSION

5.1 Results

The individual research component yielded a modular, user-friendly, secure, and efficient Python package that enables explainable AI for Rasa 2.8.8 chatbot projects. The package provides CLI and GUI interfaces for the convenience of developers with different expertise. It is available as an open-source Python package at PyPI, which is free, and the source code is publicly available on GitHub. The DIME Python package lets users generate explanations as insights for model validation and debugging and establishes reliability and trustworthiness through consistent feature importance-based model explanations. By using the DIME package, users can perform the following tasks.

1. Users can generate explanations for any Rasa 2.8.8 chatbot model. Users can specify the desired model if a Rasa project contains multiple chatbot models.

2. DIME offers both CLI and GUI feature importance visualizations. The interface selection is entirely up to the users and their level of expertise.

3. DIME visualizes explanations as individual feature importance probabilities and reveals the most influential features for the intent classification task. Users can intuitively identify models with high biases based on the DIME explanations, even if the accuracy is remarkably high.

4. The DIME package preserves explanation results as JSON objects users can easily share. Sharable explanations allow users to eliminate the need to regenerate already-generated explanations. Users can visualize the explanation objects at any DIME console outside the original project.

Section 5.2.5 elaborates on why DIME explanations are intuitive and how DIME improves the reliability of ML model predictions using an appropriate experimental setup.

5.2 Research Findings

This section consists of various testing approaches employed by the author to test different aspects of the DIME XAI implementation and a comprehensive study of the findings.

5.2.1 Feature importance calculation metric selection

Initially, the algorithms for feature importance calculation had accuracy, f1-score, and confidence score as candidate metrics for computing the GFI and LFI. Although the implementation of DIME supports all three metrics, the author tested each metric and inspected the feature importance output for both GFI and LFI to learn the most suitable metric. Table 5.1.1.1 through Table 5.1.1.3 contain three test cases for the same data instance under varying metrics and their respective outcomes. Based on the results of both tests, it is evident that accuracy and f1-score are not suitable for

Table 5.1.1.1. Metric selection test case for accuracy

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: Global Feature Importance Calculation					
Test case ID: 001			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Obtained GFIs should be comparable and sortable.					
Prerequisite: Data instance, NLU data, and a Rasa model should be passed					
Test Steps: Step 1: Add a new data instance to the *dime_config.yml* file Step 2: Open a terminal and activate the conda environment Step 3: Go to the DIME project root in the terminal Step 4: Run *dime explain -m accuracy* in the terminal and inspect the output					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
001	**Input:** SLIIT එකේ තියෙන උපාධි මොනවද? **Metric:** Accuracy	The output GFI sores must be less sparse, mostly non-zero, and sortable	මොනවද: 0.0046 තියෙන: 0.0012 SLIIT: 0.0 උපාධි: 0.0 එකේ: 0.0	fail	GFIs are mostly sparse.

calculating feature importance as most calculated feature importance scores are equal to zero (0). These two metrics produce highly sparse feature importance scores, negatively affecting the feature selection step since it is impossible to define a logical order of features without non-zero GFI scores. Comparably confidence as a metric produces more sensitive feature importance scores, allowing easy feature selection based on their descending order of importance.

Table 5.1.1.2. Metric selection test case for f1-score

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: Global Feature Importance Calculation					
Test case ID: 002			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Obtained GFIs should be comparable and sortable.					
Prerequisite: Data instance, NLU data, and a Rasa model should be passed					
Test Steps: Step 1: Add a new data instance to the *dime_config.yml* file Step 2: Open a terminal and activate the conda environment Step 3: Go to the DIME project root in the terminal Step 4: Run *dime explain -m f1-score* in the terminal and inspect the output					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
002	**Input:** SLIIT එකේ තියෙන උපාධි මොනවද? **Metric:** f1-score	The output GFI sores must be less sparse, mostly non-zero, and sortable	මොනවද: 0.0036 තියෙන: 0.0006 SLIIT: 0.0 උපාධි: 0.0 එකේ: 0.0	fail	GFIs are mostly sparse.

Table 5.1.1.3. Metric selection test case for confidence score

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: Global Feature Importance Calculation					
Test case ID: 003			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Obtained GFIs should be comparable and sortable.					
Prerequisite: Data instance, NLU data, and a Rasa model should be passed					
Test Steps: Step 1: Add a new data instance to the *dime_config.yml* file Step 2: Open a terminal and activate the conda environment Step 3: Go to the DIME project root in the terminal Step 4: Run *dime explain -m f1-score* in the terminal and inspect the output					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
003	**Input:** SLIIT එකේ තියෙන උපාධි මොනවද? **Metric:** confidence	The output GFI sores must be less sparse, mostly non-zero, and sortable	මොනවද: 4.3731 තියෙන: 1.5212 SLIIT: 0.4101 උපාධි: 0.1526 එකේ: 0.1416	pass	GFIs are not sparse and sortable.

Equivalent tests executed for five distinct data instances verified that only the confidence score computes sensitive enough GFI scores.

5.2.2 Algorithm execution time evaluation

The total algorithm execution time for DIME is the summation of the time taken to calculate the initial confidence of the unmodified dataset, GFI compute time for each token and the DFI calculation time for all sifted tokens. Table 5.1.2.1 presents the execution time for GFI calculation for the unmodified dataset, token-wise GFI calculation, and DFI calculation observed over five distinct data instances. Based on the observed execution times, the author gained the following insights.

Table 5.1.2.1. DIME algorithm execution time evaluation

Data Instance	**Number of Distinct Tokens**	**Execution time observed (seconds)**			
		Initial global confidence	**Token-wise GFI**	**DFI**	**Total Execution Time**
SLIIT එකේ තියෙන උපාධි මොනවද?	5	22.515625	SLIIT: 18 එකේ: 17 තියෙන: 17 උපාධි: 17 මොනවද: 18	0.182187	109.697812
ඩීන්ස් ලිස්ට් කියන්නේ මොකක්ද	4	23.546875	ඩීන්ස්: 17 ලිස්ට්: 18 කියන්නේ: 17 මොකක්ද: 17	0.134154	92.681029
GPA වලට අදාල grades දැනගන්න පුළුවන්ද?	6	24.25	GPA: 18 වලට: 18 අදාල: 18 grades: 18 දැනගන්න: 18 පුළුවන්ද: 18	0.196699	132.446699
ස්ලිට් එකේ scholarships දෙනවද?	4	22.53125	ස්ලිට්: 18 එකේ: 18 Scholarships: 17 දෙනවද: 17	0.088495	92.619745
බොහොම ස්තුතියි, සුබ දවසක්!	4	22.421875	බොහොම: 18 ස්තුතියි: 18 සුබ: 18 දවසක්: 18	0.066278	94.488153
Column Total	23	115.265625	406	0.667813	521.933438
Column Average	4.6 (4 or 5)	23.053125	81.2 (Instance-wise) or 17.652173 (Token-wise)	0.133563	104.3866876

1. Overall, for a training dataset with approximately 635 examples and a data instance comprising 4 or 5 words, the algorithm can conclude the execution in about 104.39 seconds (1.74 minutes).

2. For a training dataset with approximately 635 examples, the algorithm takes about 23.05 seconds to calculate the initial global confidence score.

3. For a training dataset with approximately 635 examples, DIME calculates GFI roughly in 17.65 seconds for a single token. If the data instance contains around 4 or 5 words, the GFI calculation takes about 81.2 seconds (1.35 minutes).

4. For a training dataset with approximately 635 examples and a data instance with 4 or 5 words, the algorithm takes roughly 0.13 seconds to calculate the DFI.

If the number of examples in the training dataset remains unchanged, the initial global confidence calculation takes a constant time and only GFI and DFI calculation time depend on the number of unique words present in the data instance.

It is worth noticing that GFI calculation is more compute intensive than DFI calculation, and the overall algorithm execution time is proportional to the number of examples in the dataset and the number of distinct tokens in the data instance.

5.2.3 Integration testing

The DIME XAI package has undergone a series of integration tests after integrating the sub-packages of DIME into a standalone Python package. Test cases created for each integration testing task are available from Table 5.1.3.1 through Table 5.1.3.8, and the test cases covered all primary tasks performed in the DIME CLI and GUI. Note that test cases 004 to 010 comprises smoke tests as they assessed the most critical functionalities of the integrated package.

5.2.4 Security testing

Snyk is a well-known vulnerability scanning and security testing tool that provides a web-based GUI to integrate public GitHub repositories free of charge. Snyk is popular among the open-source community, and Docker is a popular platform that has integrated Snyk as the primary container vulnerability scanning tool. The public GitHub repository of the open-source DIME package also utilizes Snyk and constantly looks for vulnerabilities and fixes them quickly. The most recent vulnerability that Snyk identified was a path traversal vulnerability in the DIME server caused by serving untrustworthy explanations download requests without verifying the path beforehand. Figure 5.1.4.1 contains the Snyk code analysis page for DIME-XAI.

Table 5.1.3.1. Integration test case for DIME project initialization

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME project initialization					
Test case ID: 004			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Create a new DIME project using DIME CLI and verify necessary files and directories are created.					
Prerequisite: DIME Python package must be installed in the virtual environment					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Create a new folder as the new DIME project root Step 3: Run *dime inti* command Step 4: Specify a sub-folder location to create a new DIME project. (Press enter to select the current directory) Step 5: Wait until the project is successfully created and observe the created files.					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
004	*dime init* command	DIME directories and files created in the new folder where the project was initialized	Observed that all folders and dime_config.yml file is correctly created.	pass	This was tested on both Windows and Linux. For Linux-based testing, Docker was used.

Table 5.1.3.2. Integration test case for DIME project initialization at rasa root

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME Project Initialization					
Test case ID: 005			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Create a new DIME project using DIME CLI on directory with a Rasa project and verify necessary files and directories are created and overlapping files are ignored.					
Prerequisite: DIME Python package must be installed in the virtual environment					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to an existing Rasa project directory in the terminal Step 3: Run *dime inti* command Step 4: Specify a sub-folder location to create a new DIME project. (Press enter to select the current directory) Step 5: Wait until the project is successfully created and observe the created files.					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
005	*dime init* command	DIME directories and files created in the directory with an existing Rasa project, without overlapping the *data* and *models* directories	Observed that all folders and dime_config.yml file is correctly created without replacing the existing Rasa NLU data or models.	pass	This was tested on both Windows and Linux.

Table 5.1.3.3. Integration test case for DIME CLI explainer

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME CLI Explainer					
Test case ID: 006			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): Medium					
Test Description: Trigger DIME explainer from CLI interface and observe if configurations are loaded as expected from the *dime_config.yml* file.					
Prerequisite: DIME Python package must be installed in the virtual environment and a trained Rasa model, and the dataset must be available in the project directory					
Test Steps: Step 1: Open *dime_config.yml* and verify that a data instance is specified, and the metric is confidence. Step 2: Open a terminal and activate the conda environment Step 3: Open the project directory in the terminal Step 4: Run *dime explain* command Step 5: Wait until explainer runs successfully and CLI visualizations are rendered.					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
006	*dime explain* command	DIME CLI Explainer must load configurations and data instance from *dime_config.yml*, generate the explanation, and visualize in the CLI.	Observed that the configurations and the data instance was loaded correctly. The CLI explainer generated the explanation and rendered it to the CLI without any issue.	pass	This was again tested with the –*instance* flag to see if the data instance gets overridden. The result was a success.

Table 5.1.3.4. Integration test case for DIME CLI visualizer

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME CLI Visualizer					
Test case ID: 007			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): Medium					
Test Description: Trigger DIME visualizer from CLI interface and observe if a valid explanation JSON object gets rendered into the CLI.					
Prerequisite: DIME Python package must be installed in the virtual environment and an already generated explanation JSON file must reside in the project directory					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to the project directory in the terminal Step 3: Run *dime visualize* command with --*explanation* flag with the correct explanation filename. (*dime_results_20220917_210016.json*) Step 4: Wait until explainer runs successfully and CLI visualizations are rendered.					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
007	*dime visualize -e filename* command	DIME CLI Visualizer must load the specified explanation object, verify the content, and visualize it in the CLI.	Observed that the specified explanation file content is correctly read and rendered in the CLI as expected.	pass	Tested with multiple explanation objects.

Table 5.1.3.5. Integration test case for DIME Server

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME Server Deployment					
Test case ID: 008			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): Medium					
Test Description: Spin up the DIME server and test the server initialization in production and debug modes.					
Prerequisite: DIME Python package must be installed in the virtual environment and a valid *dime_config.yml* file must reside in the project directory.					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to the project directory in the terminal Step 3: Run *dime server* command with --*debug* to turn on debugging Step 4: Wait until the DIME server starts in the debug mode and open up the URL in a web browser and confirm that the server is up and running.					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
008	*dime server --debug* command	DIME Server must initialize in the debug mode and serve the DIME developer console on port 6066	Observed that the server loads as expected in port 6066. The developer console was opened and confirmed that the server hosts the DIME developer console without an issue.	pass	The same steps were repeated without the --*debug* flag to test the production mode with *waitress*.

Table 5.1.3.6. Integration test case for DIME Server explanations

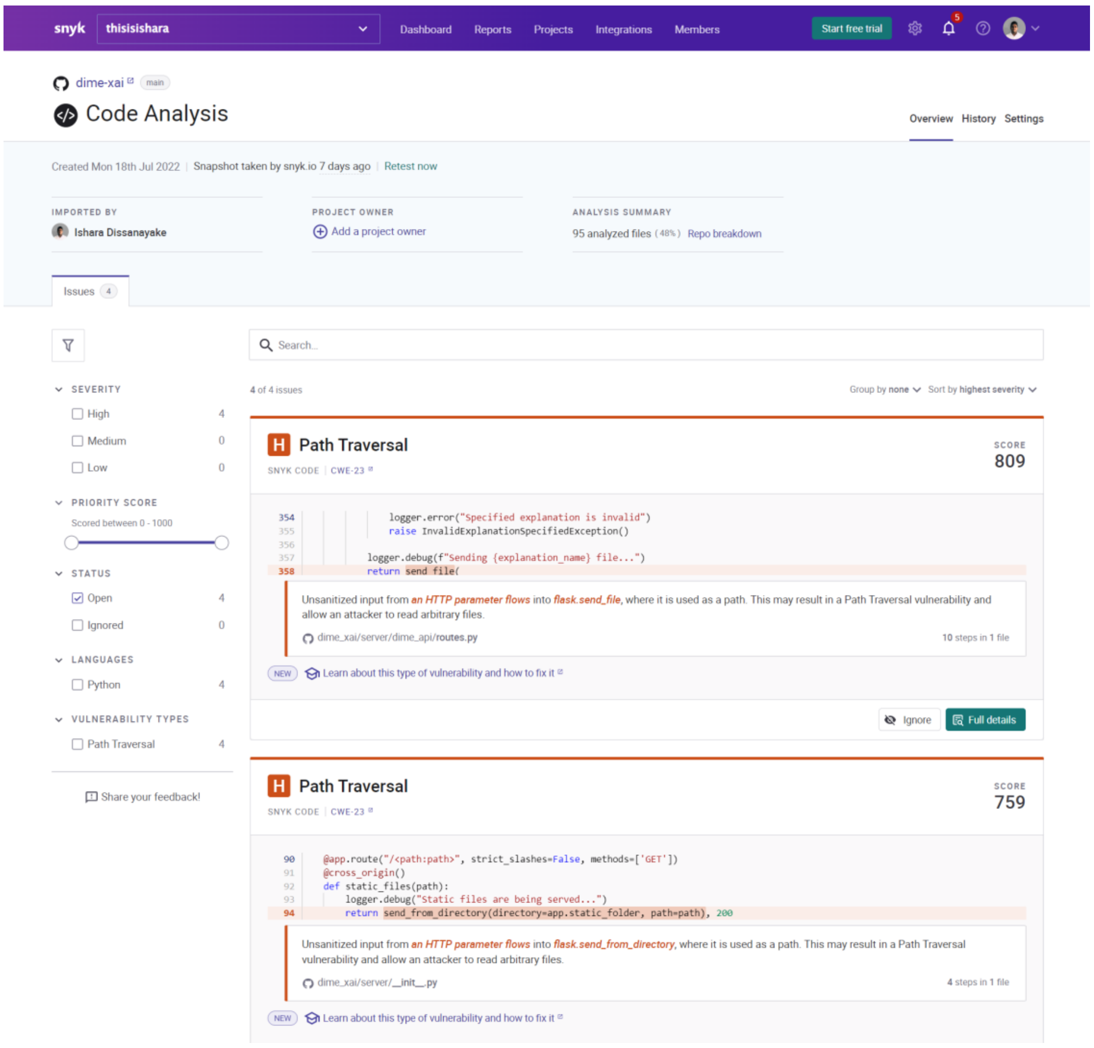
Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME Server Explanation Generation					
Test case ID: 009			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Spin up the DIME server and test the server initialization in production and debug modes to test explanation generation.					
Prerequisite: DIME Python package must be installed in the virtual environment and a valid *dime_config.yml* file must reside in the project directory.					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to the project directory in the terminal Step 3: Run *dime server* command with --*debug* to turn on debugging Step 4: Wait until the DIME server starts in the debug mode and open the URL in a web browser and confirm that the server is up and running. Step 5: Go to the dashboard page and change the data instance Step 6: Click on *explain* button to start an explanation job Step 7: Insect terminal and verify the request is processing Step 8: Wait till the job finishes and inspect if the generated visualization appears at the bottom of the dashboard page Step 9: Go to the *explanations* page and verify the latest explanation object is available (based on the timestamp) Step 10: Open the explanation and verify that it is correctly rendering Step 11: Go to the dashboard page and verify that the explanations count on the top status bar has increased or not					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
009	Run *dime server --debug* Input any data instance in the dashboard	Explanation job must run, and the request id must be attached. After the job completes, result must render, and the statistics and the explanations list should be automatically updated.	Observed that the server runs the job with a valid request id. The results were rendered, statics were updated, and the explanation list contained the latest object.	pass	This was repeatedly done in the production environment and for various input data instances.

Table 5.1.3.7. Integration test case for DIME Server visualizations

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME Server Visualization					
Test case ID: 010			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): High					
Test Description: Spin up the DIME server and test the server initialization in production and debug modes to test explanation visualization.					
Prerequisite: DIME Python package must be installed in the virtual environment and a valid *dime_config.yml* file must reside in the project directory.					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to the project directory in the terminal Step 3: Run *dime server* command with --*debug* to turn on debugging Step 4: Wait until the DIME server starts in the debug mode and open up the URL in a web browser and confirm that the server is up and running. Step 5: Navigate to the *explanations* tab in the sidebar Step 6: Under explanations list, click on *visualize* button under any explanation Step 7: Inspect the explanation visualization overlay					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
010	Run *dime server --debug* command	Explanation must be rendered when clicking on the *visualize* button and an overlay should slide from the right side of the screen. Both GFI and DFI must be correctly rendered.	Observed that the overlay appears as expected and GFI and LFI visualizations are correctly rendered.	pass	The same steps were repeated for multiple explanations.

Table 5.1.3.8. Integration test case for DIME Server peak feature

Project ID: 2022-056-IT1906932					
Project Name: DIME-XAI					
Project Function: DIME Server Explanation Peak					
Test case ID: 011			**Test case designed by:** **ID No:** IT19069432 **Name:** Dissanayake D.M.I.M.		
Test Priority (High/Medium/Low): Medium					
Test Description: Spin up the DIME server and test the server initialization in production and debug modes and verify the *peak* feature is working as expected.					
Prerequisite: DIME Python package must be installed in the virtual environment and a valid *dime_config.yml* file must reside in the project directory.					
Test Steps: Step 1: Open a terminal and activate the conda environment Step 2: Go to the project directory in the terminal Step 3: Run *dime server* command with --*debug* to turn on debugging Step 4: Wait until the DIME server starts in the debug mode and open up the URL in a web browser and confirm that the server is up and running. Step 5: Navigate to the *explanations* tab in the sidebar Step 6: Click on *peak* button on the top-left corner Step 7: Pick a previously downloaded explanation object from the file open dialog Step 8: Verify if the visualizations are rendering and not permanently persisting					
Test ID	Test Inputs	Expected Outputs	Actual Output	Result (Pass/Fail)	Comments
011	*dime server --debug* command	The picked explanation object must be visualized right away, but should not upload it to the server, thus the explanation count, and the list must remain unchanged.	Observed that the selected object was correctly rendered when peaked. The object was not uploaded, and explanations the count and the list remained unchanged.	pass	The same steps were repeated for multiple explanation objects.



Code Analysis

Created Mon 18th Jul 2022 | Snapshot taken by snyk.io 7 days ago | Retest now

IMPORTED BY: Ishara Dissanayake
PROJECT OWNER: Add a project owner
ANALYSIS SUMMARY: 95 analyzed files (48%) Repo breakdown

SEVERITY	
High	4
Medium	0
Low	0
PRIORITY SCORE	
Scored between 0 - 1000	
STATUS	
✓ Open	4
Ignored	0
LANGUAGES	
Python	4
VULNERABILITY TYPES	
Path Traversal	4

4 of 4 issues — Group by none — Sort by highest severity

H Path Traversal — SNYK CODE | CWE-23 — SCORE 809

```
                logger.error("Specified explanation is invalid")
                raise InvalidExplanationSpecifiedException()

            logger.debug(f"Sending {explanation_name} file...")
            return send_file(
```

Unsanitized input from *an HTTP parameter flows* into *flask.send_file*, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

dime_xai/server/dime_api/routes.py — 10 steps in 1 file

NEW Learn about this type of vulnerability and how to fix it

H Path Traversal — SNYK CODE | CWE-23 — SCORE 759

```
@app.route("/<path:path>", strict_slashes=False, methods=['GET'])
@cross_origin()
def static_files(path):
    logger.debug("Static files are being served...")
    return send_from_directory(directory=app.static_folder, path=path), 200
```

Unsanitized input from *an HTTP parameter flows* into *flask.send_from_directory*, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

dime_xai/server/__init__.py — 4 steps in 1 file

NEW Learn about this type of vulnerability and how to fix it

Figure 5.1.4.1. DIME XAI Security testing with Snyk

5.2.5 The impact of DIME explanations

The author of the individual research component conducted a carefully designed experiment to evaluate the effect of DIME explanations. The experiment environment consisted of two carefully crafted Rasa chatbot models, Bot-1 and Bot-2. Bot-1 consists of a highly biased dataset, and Bot-2 employs a slightly better dataset with low bias. Both datasets contain 64 examples, and the author has separated 14 examples as the testing set. Each dataset contained four intent categories: greet, degrees, degree_requirements, and nlu_fallback. Bot-1 and Bot-2 have trained for 15 epochs under the same Rasa NLU pipeline configurations. At the end of the training process, Bot-1 demonstrated 92.86% validation accuracy, and Bot-2 delivered a validation accuracy of 85.71%. See Appendix C: DIME Experiment Learning Curves for learning curves of Bot-1 and Bot-2 for testing and validation accuracy. Table 5.2.5.1 summarizes the Bot-1 and Bot-2 datasets utilized, configuration specified, and accuracy observed.

The author randomly selected five testers with at least a basic ML knowledge for the experiment and first asked each user to determine the best chatbot model among Bot-1 and Bot-2 suitable for deployment, considering the accuracy scores based on their level of understanding. The testers then queried Bot-1 and Bot-2 and asked a degree-related query, *University එකේ තියෙන ඩිග්‍රීස් මොනවද?* (*What are the degrees offered at the University*), that should fall under the *degree* intent to verify if the bots accurately respond. As the next step of the experiment, testers determined the most meaningful word from words in the specified query when categorizing it as a *degree*-related intent solely based on their intuition. In the final phase of the experiment, the author provided the testers with two DIME GUIs for Bot-1 and Bot-2, asked them to generate explanations for the same data instance they used to query the bots, and provided the testers with a second opportunity to determine the best chatbot model based on the DIME explanations to observe the impact.

Table 5.2.5.1. Summarized DIME experiment setup details

Bot #	**Dataset Size**	**Dataset Bias**	**DIET epochs**	**Validation Accuracy**
Bot-1	Total: 64 Train: 50 Test: 14 Total intent categories: 4	High bias with many similar examples	15	92.86%
Bot-2	Total: 64 Train: 50 Test: 14 Total intent categories: 4	Low bias with varying patterns of examples	15	85.71%

Table 5.2.5.2 concludes the observations of the experiment. All testers selected Bot-1 as the best chatbot model due to its high accuracy and solely based on manual testing, with no knowledge of DIME explanations. However, after exposing them to DIME explanations for Bot-1 and Bot-2, testers detected the biased model as Bot-1 and declared Bot-2 as the best model. Five out of five testers justified the reason for altering their selection, mentioning that Bot-1 did not consider the word *ඩිග්‍රීස්* as the most important word for the classification task. Instead, Bot-1 revealed *University* as the most important word for classifying the query *University එකේ තියෙන ඩිග්‍රීස් මොනවද?* as a *degree*-related question. Three out of five testers stated that accuracy is misleading compared to explanations. The experiment results demonstrate how the most influential word selected by the testers and Bot-2 are the same. Based on the observations, it is evident that users prioritize logical model explanations over performance scores, and DIME explanations can further enhance the model validation process.

Appendix D: DIME Experiment Setup contains the chatbot widgets and the user interfaces designed specifically for the experiment. The complete DIME experiment setup and observations are available as a GitHub repository[24], and any individual can replicate the experiment described in this section at any time. The *README.md* file

Table 5.2.5.2. DIME experiment observations

Tester #	**Best Model without DIME**	**Most Important Word based on the Intuition of the Tester**	**Most Important Word given by DIME Explanations**	**Best Model after exposed to Explanations**	**Are explanations helpful for model debugging**
Tester 1	Bot-1	ඩිග්‍රීස්	Bot-1: University Bot-2: ඩිග්‍රීස්	Bot-2	Yes
Tester 2	Bot-1	ඩිග්‍රීස්	Bot-1: University Bot-2: ඩිග්‍රීස්	Bot-2	Yes
Tester 3	Bot-1	ඩිග්‍රීස්	Bot-1: University Bot-2: ඩිග්‍රීස්	Bot-2	Yes
Tester 4	Bot-1	ඩිග්‍රීස්	Bot-1: University Bot-2: ඩිග්‍රීස්	Bot-2	Yes
Tester 5	Bot-1	ඩිග්‍රීස්	Bot-1: University Bot-2: ඩිග්‍රීස්	Bot-2	Yes

[24] https://github.com/thisisishara/DIME-test-setup

given in the GitHub repository contains instructions for properly setting up the experiment environment in detail. To obtain the observations as a report, see Appendix E: DIME Experiment Observations.

5.3 Discussion

Based on the observations mentioned in section 5.2.1, the author retained confidence as the core metric for calculating feature importance in the DIME algorithm due to the less sparse scores obtained and does not advise using accuracy and f1-score for the task. As previously mentioned, many GFI scores with zero importance prevent ordering the features logically in descending order and are problematic when selecting the words of the highest importance. The confidence score is well suited for the task due to the less sparse GFI scores observed.

The author suggested utilizing Shapley values for LFI calculation at the beginning of the research but decided to use confidence drop instead, considering the algorithm efficiency. GFI-based feature selection in DIME has assisted LFI calculation to acquire globally faithful explanations, and DIME local explanations only contain globally influential features, which is the main reason why the author introduces LFI under a new name called DFI. DFI represents LFI calculated only for words with higher GFI.

Unlike input perturbation based XAI approaches, the DIME XAI approach is resistant to adversarial attacks since DIME is not utilizing input perturbations (see section 2). It makes fooling DIME explanations difficult, ensuring the consistency and reliability of the DIME explanations.

Based on the execution time evaluations mentioned in section 5.2.2, an average DIME explanation takes only approximately 2 minutes to generate, which is arguably acceptable in a production-grade explainable AI solution. DIME enables generating model explanations with or without a GUI, and the author has extensively tested DIME on both Windows and Linux operating systems and on Docker. i.e., DIME can run in diverse environments without an issue. Rasa developers can utilize the open-source DIME package and integrate it with any chatbot project that uses Rasa 2.8.8. The Rasa version selected is an actively developed Rasa version by Rasa HQ, which makes

DIME available to a more extensive set of chatbot developers. As mentioned in sections 5.2.3 and 5.2.4, the maintainers of the DIME Python package constantly work on improvements and security testing, making DIME secure and feature rich.

Since the DIME package is open-source and the explanation generation approach is relatively novel, the author has paid careful attention to preserving the maintainability of the source code. All Python-based code adheres to the PEP8 style guide, and each module utilizes Python docstrings to state the purpose of the functions, classes, and methods. DIME also provides comprehensive documentation on DIME CLI, GUI, and Python API. It makes it easy for an interested individual to easily contribute to the project or refer to and understand the source code with minimum effort.

Observations of the experiment mentioned in section 5.2.5 elaborate on how DIME explanations can enhance the model validation process in chatbots and demonstrate the drawbacks of relying on just performance scores. The experiment indicates the importance of integrating an explainable AI technique with AI-centric applications to enhance the trustworthiness and reliability of automated decision-making enabled through overly complex algorithms such as deep learning. The section exemplifies how users tend to give a low priority to higher performance scores and trust explanations when it is clear how machine learning or deep learning models make decisions.

5.4 Individual Contribution to the Overall Research

Section 4.7 clearly states the individual research component contribution towards the overall research outcome. This research component is responsible for enabling explainable AI in the Kolloqe chatbot development platform, taking the ML model validation task a step further. Refer to section 4.7 for a more detailed clarification of the individual research contribution.

6. CONCLUSION

XAI is crucial in interpreting how a black-box ML model delivers outcomes. Regulations such as GDPR have introduced a right to explanation due to the rise of opaque algorithms and autonomous decision-making to enforce trust through reliable, clear, and concise explanations. XAI also assists in debugging ML models to identify possible biases. This research component delivered a comprehensive analysis of existing XAI approaches in NLP and explainability in chatbot frameworks. Section 1 of this dissertation elaborated on how explanations can assist in identifying and debugging model biases not accentuated by performance scores, such as accuracy or f1-score. The introduction of the dissertation also discussed why current XAI approaches are inadequate to achieve locally and globally faithful explanations and the need for a novel XAI approach to acquire globally precise local model explanations.

The author has successfully delivered DIME, a novel XAI approach that offers both locally and globally faithful explanations and implemented a modular, efficient, open-source, and production-grade Python package that developers can integrate with Rasa 2.8.8 chatbot projects. The produced research outcome delivers model explanations for DIET classifiers utilized in Rasa ML pipelines and provides comprehensive visualizations on both CLI and GUI interfaces, supporting the explanation generation process in numerous operating systems and environments. Interested individuals can contribute to the open-source package and utilize it for further research or other related work in the XAI and NLP domains.

Section 5 of the dissertation elaborated on how the author conducted testing and evaluated the introduced XAI approach with ample details and what changed in the initially proposed research approach and why. The experiment carried out by the author demonstrated how users tend to trust the explanations over the performance scores simply due to the perceivable justifications provided by model explanations generated using the introduced XAI technique.

The dissertation also revealed that none of the widely utilized chatbot-building frameworks and platforms relies on the trustworthiness of the model decision or offers valid justifications when selecting, evaluating, or debugging ML models. The research primarily encourages utilizing explainability in AI-centric applications to validate model decisions instead of blindly accepting decisions given by complex automated algorithms or systems. The author demonstrates this by integrating the DIME XAI approach into Kolloqe: the conversational AI development platform delivered by the overall research, to allow developers to identify and deploy only reliable chatbot models.

REFERENCES

[1] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A survey of the state of explainable AI for natural language processing," *arXiv preprint arXiv:2010.00711,* 2020.

[2] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," *arXiv preprint arXiv:1712.05181,* 2017.

[3] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," *arXiv preprint arXiv:2004.09936,* 2020.

[4] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems,* vol. 32, no. 2, pp. 604-624, 2020.

[5] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328,* 2017.

[6] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006,* 2017.

[7] V. Dignum, "Responsible Artificial Intelligence--from Principles to Practice," *arXiv preprint arXiv:2205.10785,* 2022.

[8] P. Regulation, "Regulation (EU) 2016/679 of the European Parliament and of the Council," *Regulation (eu),* vol. 679, p. 2016, 2016.

[9] A. D. Selbst and J. Powles, "Meaningful information and the right to explanation," *International Data Privacy Law,* vol. 7, no. 4, pp. 233-242, 2017, doi: 10.1093/idpl/ipx022.

[10] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of AI ethics guidelines," *Nature Machine Intelligence,* vol. 1, no. 9, pp. 389-399, 2019.

[11] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 2018: IEEE, pp. 0210-0215.

[12] B. Galitsky and S. Goldberg, "Explainable machine learning for chatbots," in *Developing Enterprise Chatbots*: Springer, 2019, pp. 53-83.

[13] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital signal processing,* vol. 73, pp. 1-15, 2018.

[14] S. R. Islam, W. Eberle, S. K. Ghafoor, and M. Ahmed, "Explainable artificial intelligence approaches: A survey," *arXiv preprint arXiv:2101.09429,* 2021.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "" Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135-1144.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems,* vol. 30, 2017.

[17] L. S. Shapley, "17. A value for n-person games," in *Contributions to the Theory of Games (AM-28), Volume II*: Princeton University Press, 2016, pp. 307-318.

[18] P. Gohel, P. Singh, and M. Mohanty, "Explainable AI: current status and future directions," *arXiv preprint arXiv:2107.07045,* 2021.

[19] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[20] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable machine learning–a brief history, state-of-the-art and challenges," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2020: Springer, pp. 417-431.

[21] S. Carton, Q. Mei, and P. Resnick, "Extractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts," *arXiv preprint arXiv:1809.01499,* 2018.

[22] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, "Program induction by rationale generation: Learning to solve and explain algebraic word problems," *arXiv preprint arXiv:1705.04146,* 2017.

[23] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein, "Explainable prediction of medical codes from clinical text," *arXiv preprint arXiv:1802.05695,* 2018.

[24] Y. Dong, Z. Li, M. Rezagholizadeh, and J. C. K. Cheung, "EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing," *arXiv preprint arXiv:1906.08104,* 2019.

[25] P. Lertvittayakumjorn and F. Toni, "Human-grounded evaluations of explanation methods for text classification," *arXiv preprint arXiv:1908.11355,* 2019.

[26] M. Robnik-Šikonja and M. Bohanec, "Perturbation-based explanations of prediction models," in *Human and machine learning*: Springer, 2018, pp. 159-175.

[27] W. Marrakchi, "Explaining by Conversing: The Argument for Conversational Xai Systems," Harvard University, 2021.

[28] G. Caldarini, S. Jaf, and K. McGarry, "A literature survey of recent advances in chatbots," *Information,* vol. 13, no. 1, p. 41, 2022.

[29] K. Mikhail. "Overview — ELI5 0.11.0 documentation." https://eli5.readthedocs.io/en/latest/overview.html (accessed Aug. 22,, 2022).

[30] D. Alvarez-Melis and T. S. Jaakkola, "On the robustness of interpretability methods," *arXiv preprint arXiv:1806.08049,* 2018.

[31] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180-186.

[32] N. Puri, P. Gupta, P. Agarwal, S. Verma, and B. Krishnamurthy, "Magix: Model agnostic globally interpretable explanations," *arXiv preprint arXiv:1706.07160,* 2017.

[33] M. Danilevsky, S. Dhanorkar, Y. Li, L. Popa, K. Qian, and A. Xu, "Explainability for Natural Language Processing," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 4033-4034.

[34] D. Rajagopal, V. Balachandran, E. Hovy, and Y. Tsvetkov, "Selfexplain: A self-explaining architecture for neural text classifiers," *arXiv preprint arXiv:2103.12279,* 2021.

[35] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*, 2017: PMLR, pp. 3145-3153.

[36] Z. Xue, R. Li, and M. Li, "Recent Progress in Conversational AI," *arXiv preprint arXiv:2204.09719,* 2022.

[37] S. Pérez-Soler, S. Juarez-Puerta, E. Guerra, and J. de Lara, "Choosing a chatbot development tool," *IEEE Software,* vol. 38, no. 4, pp. 94-103, 2021.

[38] A. Singh, K. Ramasubramanian, and S. Shivam, "Introduction to Microsoft Bot, RASA, and Google Dialogflow," in *Building an Enterprise Chatbot: Work with Protected Enterprise Data Using Open Source Frameworks*. Berkeley, CA: Apress, 2019, pp. 281-302.

[39] Z. Chen *et al.*, "Towards explainable conversational recommendation," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 2994-3000.

[40] V. B. Nguyen, J. Schlotterer, and C. Seifert, "Explaining Machine Learning Models in Natural Conversations: Towards a Conversational XAI Agent," 2022.

[41] C. Werner, "Explainable AI through Rule-based Interactive Conversation," in *Edbt/icdt workshops*, 2020.

[42] R. Dazeley, P. Vamplew, C. Foale, C. Young, S. Aryal, and F. Cruz, "Levels of explainable artificial intelligence for human-aligned conversational explanations," *Artificial Intelligence,* vol. 299, p. 103525, 2021.

[43] N. Nobani, F. Mercorio, and M. Mezzanzanica, "Towards an Explainer-agnostic Conversational XAI," in *IJCAI*, 2021, pp. 4909-4910.

[44] M. Kuźba, "Conversational explanations of Machine Learning models using chatbots."

[45] A. Khurana, P. Alamzadeh, and P. K. Chilana, "ChatrEx: Designing explainable chatbot interfaces for enhancing usefulness, transparency, and trust," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2021: IEEE, pp. 1-11.

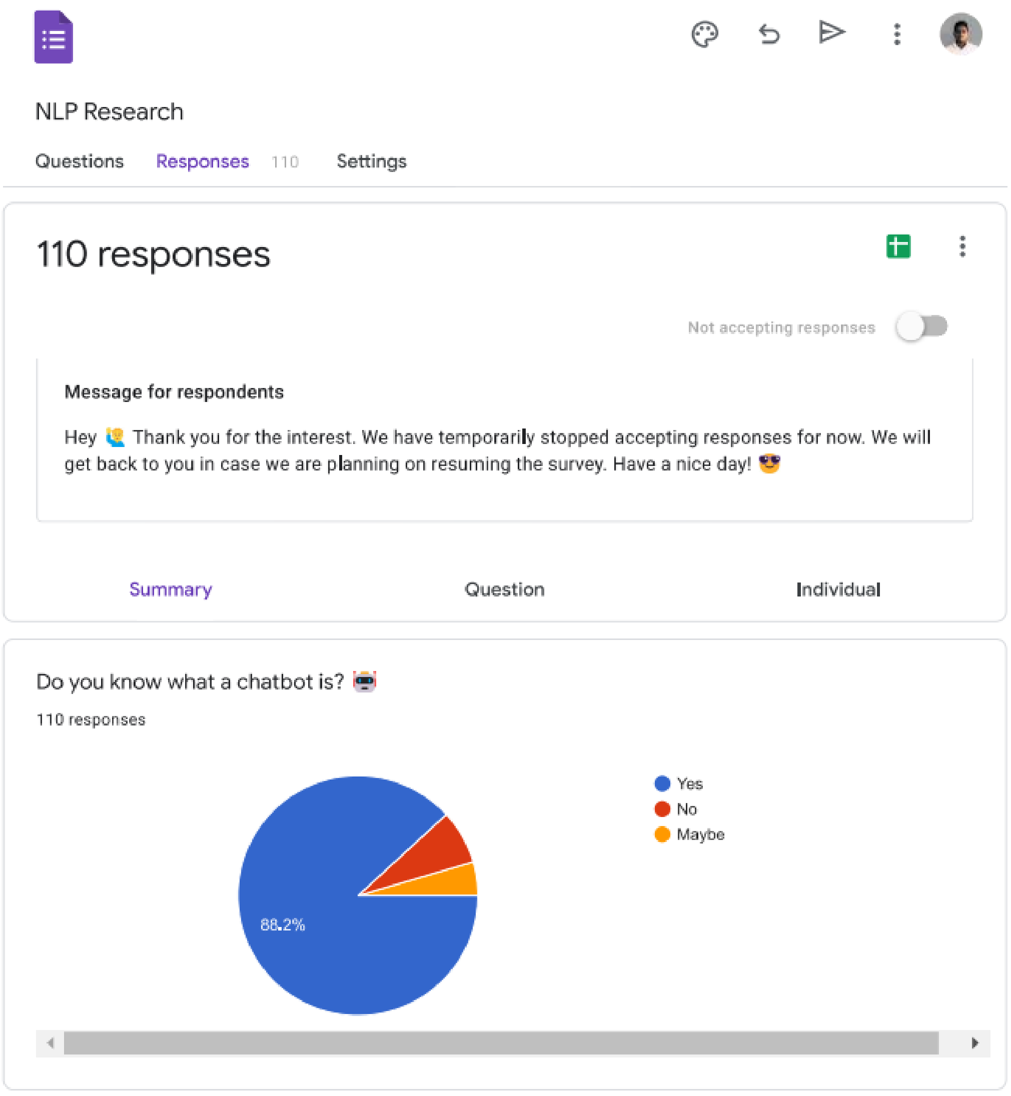
GLOSSARY

Term	Definition
Artificial Intelligence (AI)	A field of study and often refers to the intelligence demonstrated by machines in contrast to natural intelligence.
Machine Learning (ML)	A subfield of AI that concerns development of methods that leverages on data to improve performance in executing a task or set of tasks.
Natural Language Processing (NLP)	A subfield of linguistics, computer science, and artificial intelligence that concerns how to enable machines to process enormous quantities of natural language data.
Explainable AI (XAI)	A subfield of AI that concerns interpreting black-box machine learning models in a way humans can comprehend the decisions taken by AI.
Rasa Open-Source	An open-source conversational AI development platform developed and maintained by Rasa HQ
Training Dataset	A comprehensive set of data utilized to train machine learning and deep learning models
Holdout Set	A portion of the dataset held back to provide a final estimation of a machine learning or deep learning model performance after training
Intent Classification	The task of associating words, phrases, queries, or expressions with an exact category in NLP
Feature Importance	The technique of assigning a score to input features of a machine learning or deep learning model based on how useful they are for the decisions carried out by the model

Feature Perturbation	Generating varying samples from a given data instance by systematically eliminating or replacing input features with noise
Bag of Words	A simplifying representation utilized in NLP to represent a given text instance or set of instances considering distinct words present in the text

APPENDICES

Appendix A: Survey Form



NLP Research

110 responses

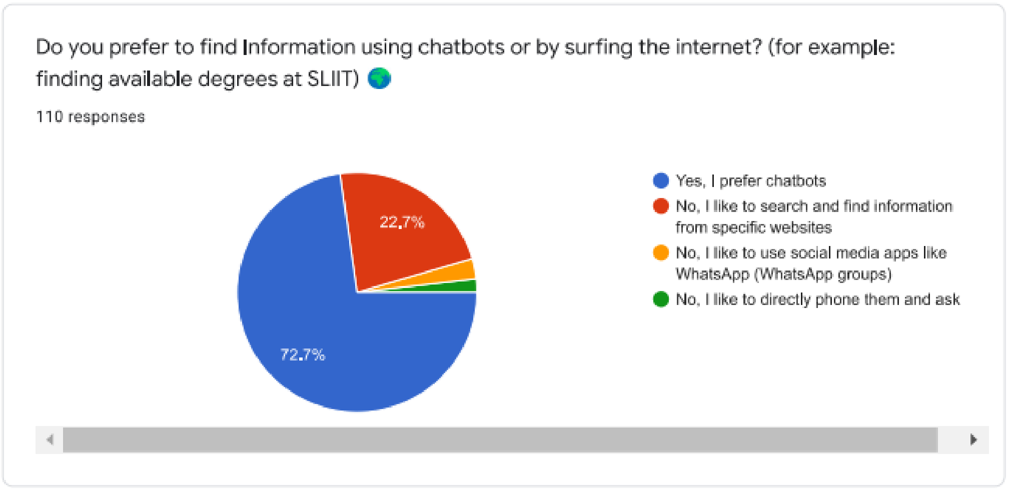
Message for respondents

Hey 🙋 Thank you for the interest. We have temporarily stopped accepting responses for now. We will get back to you in case we are planning on resuming the survey. Have a nice day! 😎

Do you know what a chatbot is? 🤖

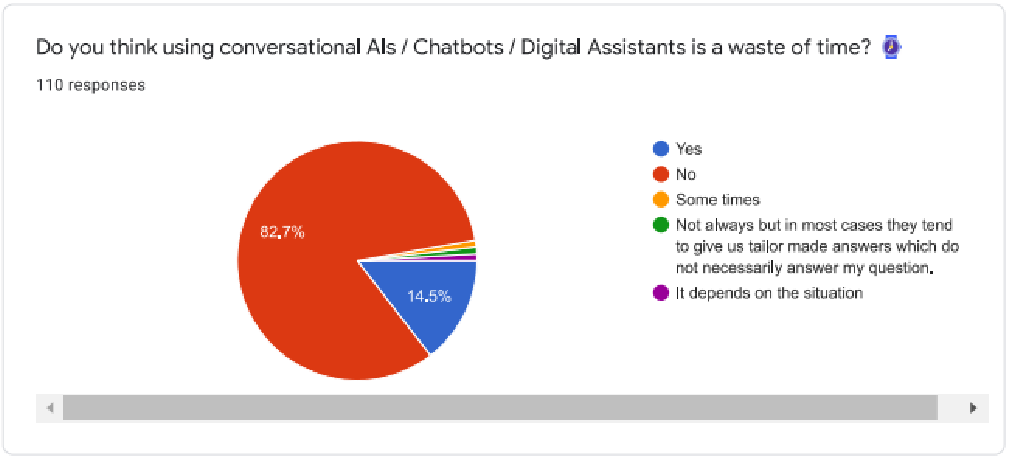
110 responses

Figure A.1: Complete survey form questions and responses – part 1


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

110 responses

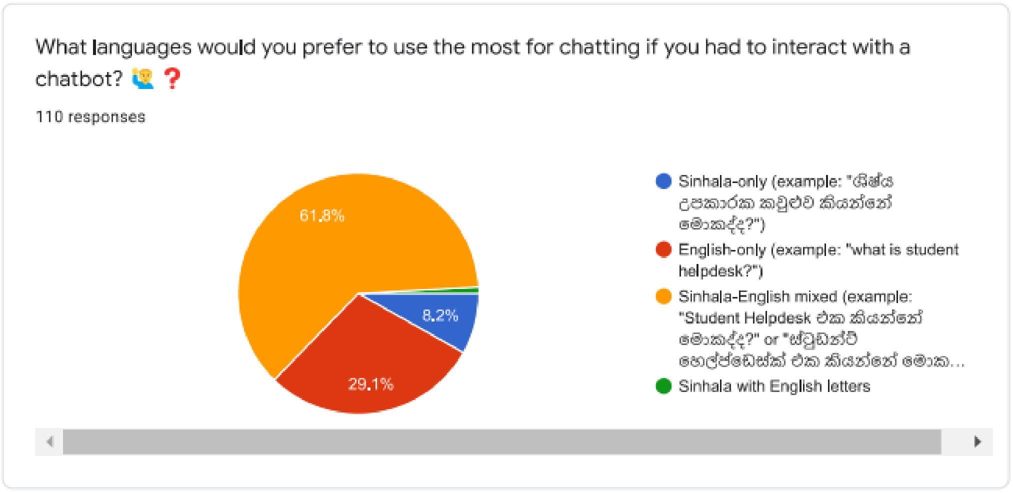
22.7%
72.7%
● Yes, I prefer chatbots
● No, I like to search and find information from specific websites
● No, I like to use social media apps like WhatsApp (WhatsApp groups)
● No, I like to directly phone them and ask


Do you think using conversational AIs / Chatbots / Digital Assistants is a waste of time? ⌚

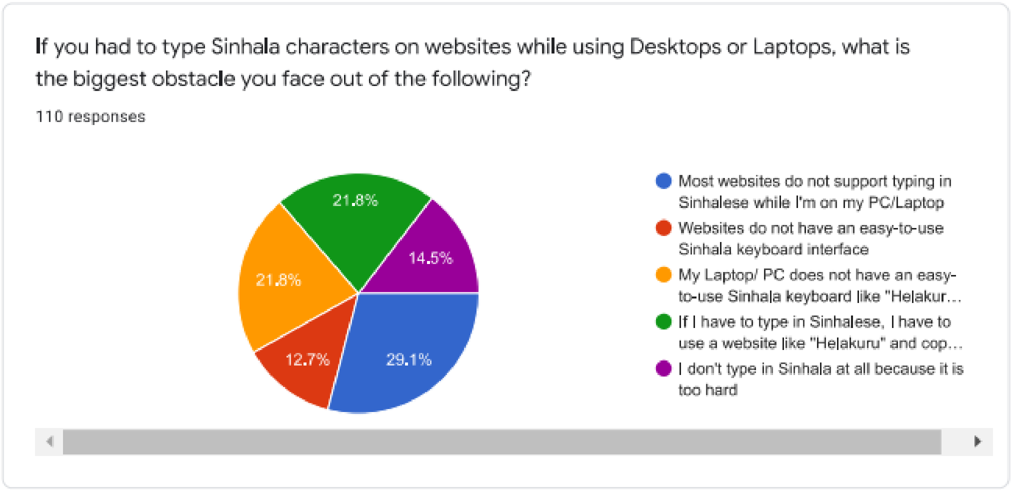
110 responses

82.7%
14.5%
● Yes
● No
● Some times
● Not always but in most cases they tend to give us tailor made answers which do not necessarily answer my question.
● It depends on the situation

Figure A.2: Complete survey form questions and responses – part 2

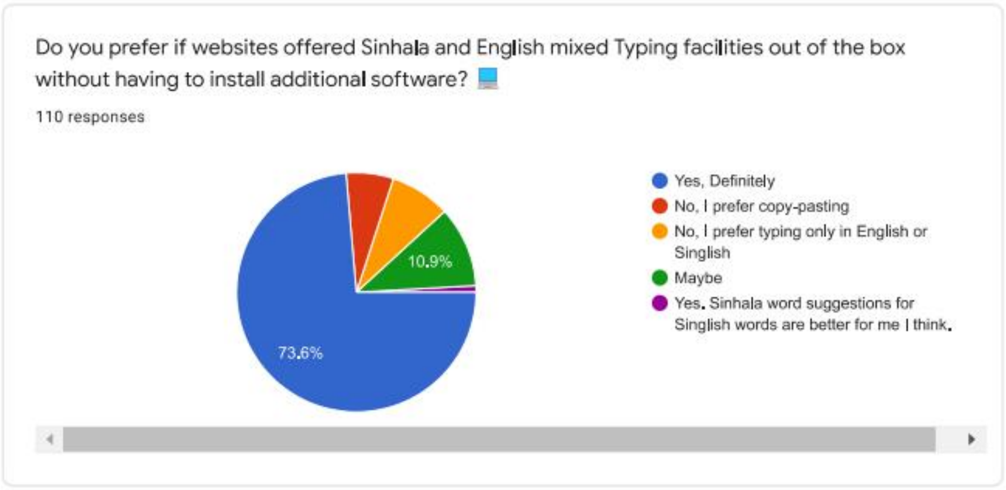

What languages would you prefer to use the most for chatting if you had to interact with a chatbot? 🙋 ❓

110 responses

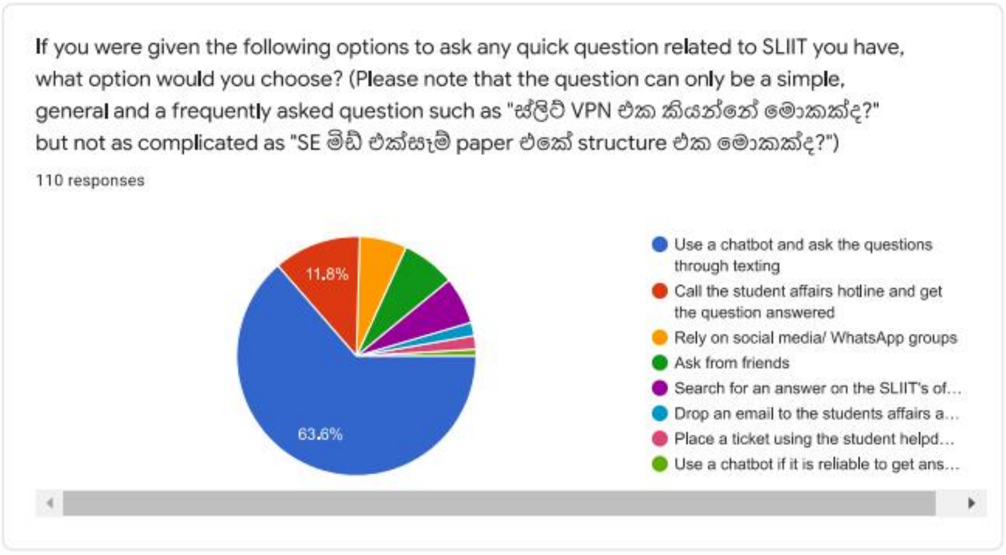

If you had to type Sinhala characters on websites while using Desktops or Laptops, what is the biggest obstacle you face out of the following?

110 responses

Figure A.3: Complete survey form questions and responses – part 3

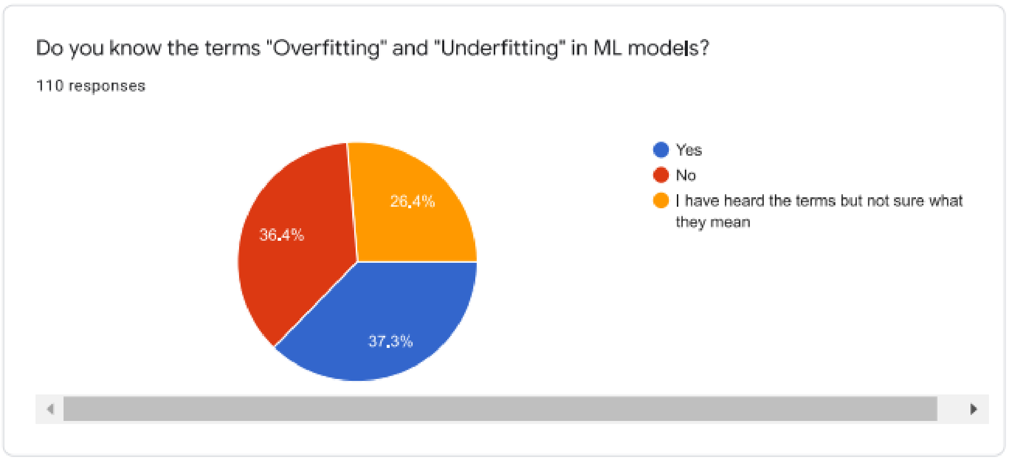

Do you prefer if websites offered Sinhala and English mixed Typing facilities out of the box without having to install additional software? 💻

110 responses

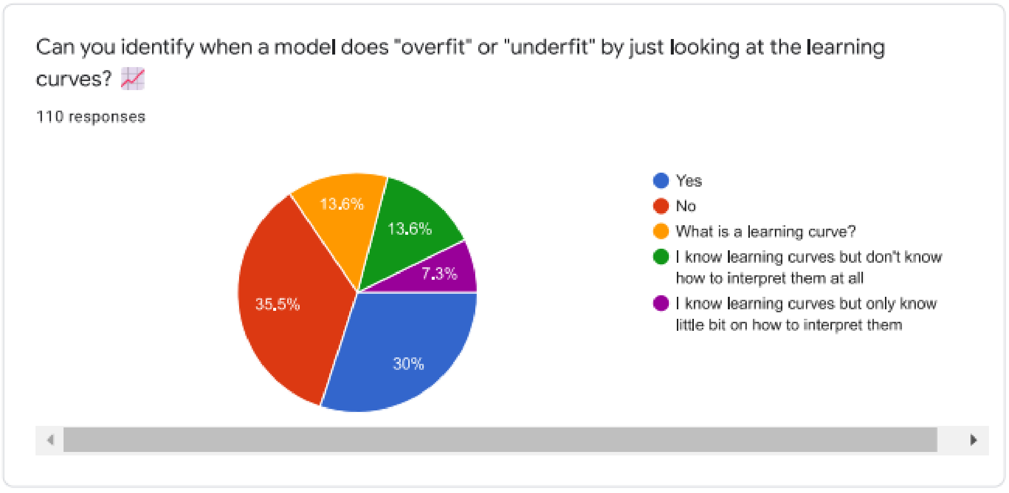

If you were given the following options to ask any quick question related to SLIIT you have, what option would you choose? (Please note that the question can only be a simple, general and a frequently asked question such as "ස්ලිට් VPN එක කියන්නේ මොකක්ද?" but not as complicated as "SE මිඩ් එක්සෑම් paper එකේ structure එක මොකක්ද?")

110 responses

Figure A.4: Complete survey form questions and responses – part 4

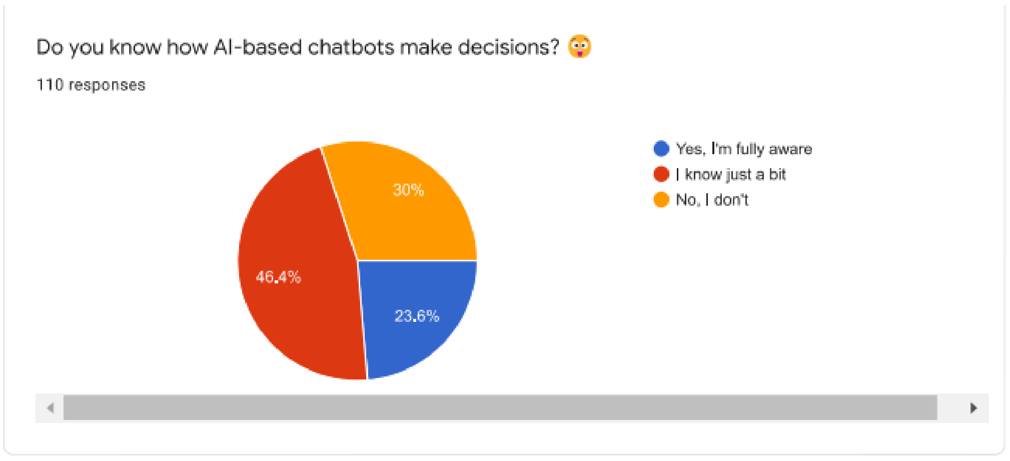

Do you know the terms "Overfitting" and "Underfitting" in ML models?

110 responses

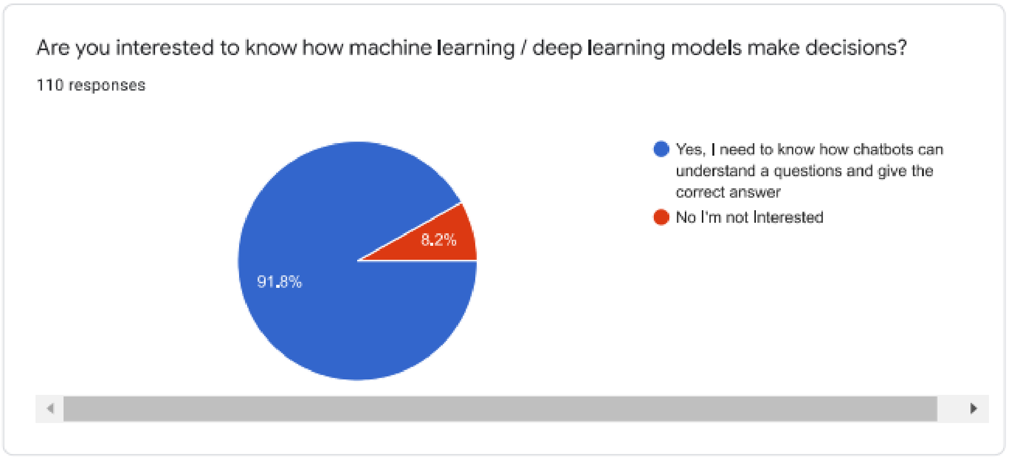

Can you identify when a model does "overfit" or "underfit" by just looking at the learning curves? 📈

110 responses

Figure A. 5: Complete survey form questions and responses – part 5

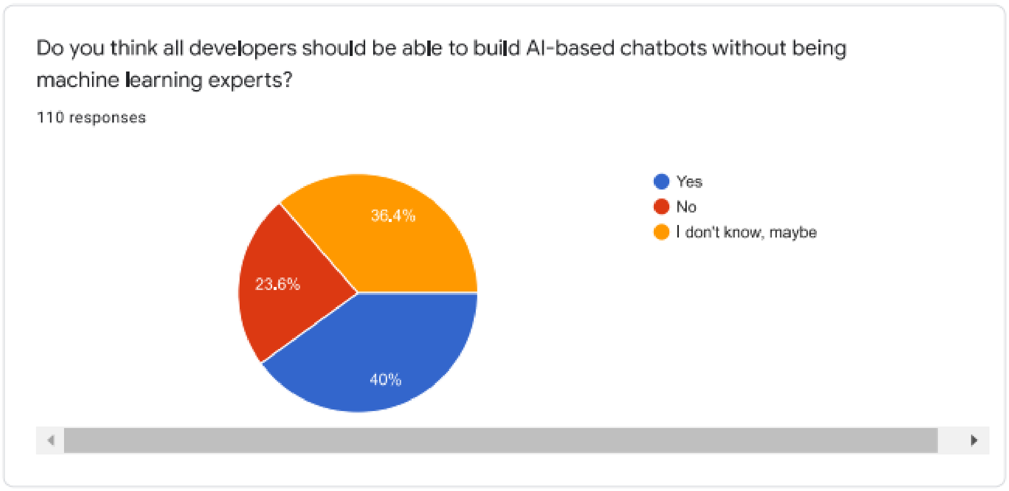

Do you know how AI-based chatbots make decisions? 😳

110 responses

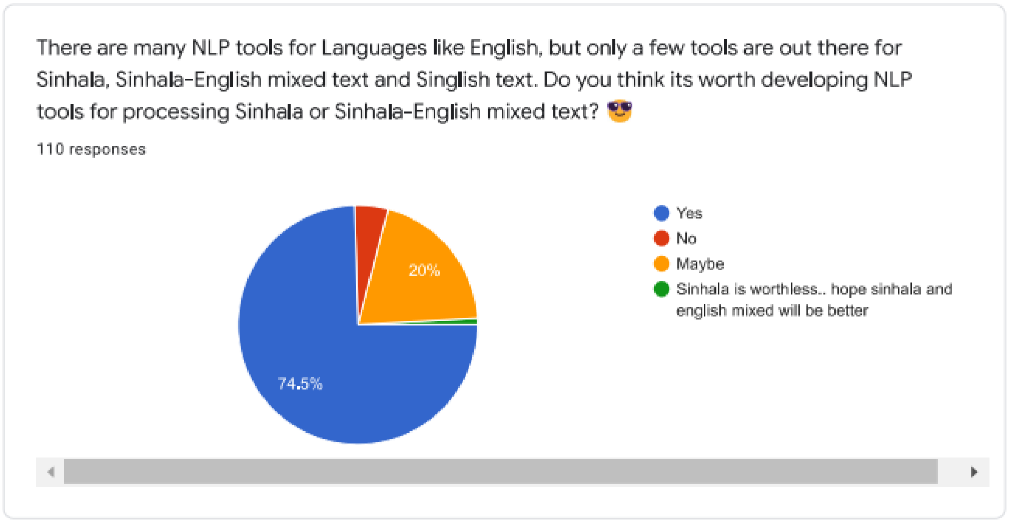

Are you interested to know how machine learning / deep learning models make decisions?

110 responses

Figure A.6: Complete survey form questions and responses – part 6

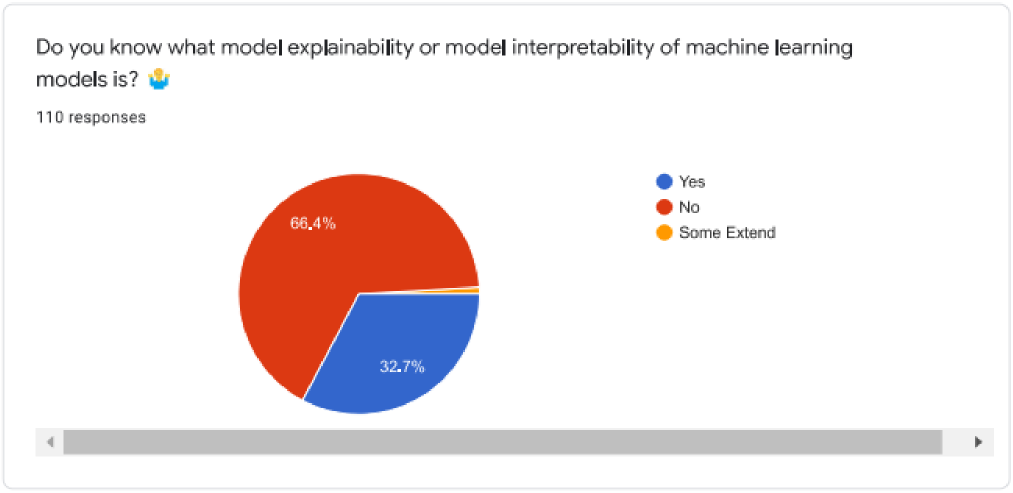

Do you think all developers should be able to build AI-based chatbots without being machine learning experts?

110 responses

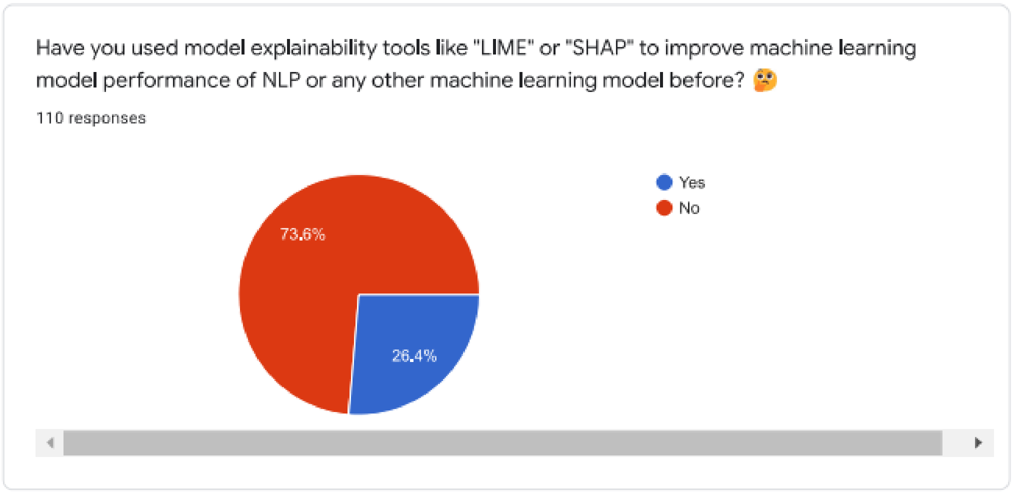

There are many NLP tools for Languages like English, but only a few tools are out there for Sinhala, Sinhala-English mixed text and Singlish text. Do you think its worth developing NLP tools for processing Sinhala or Sinhala-English mixed text? 😎

110 responses

Figure A.7: Complete survey form questions and responses – part 7

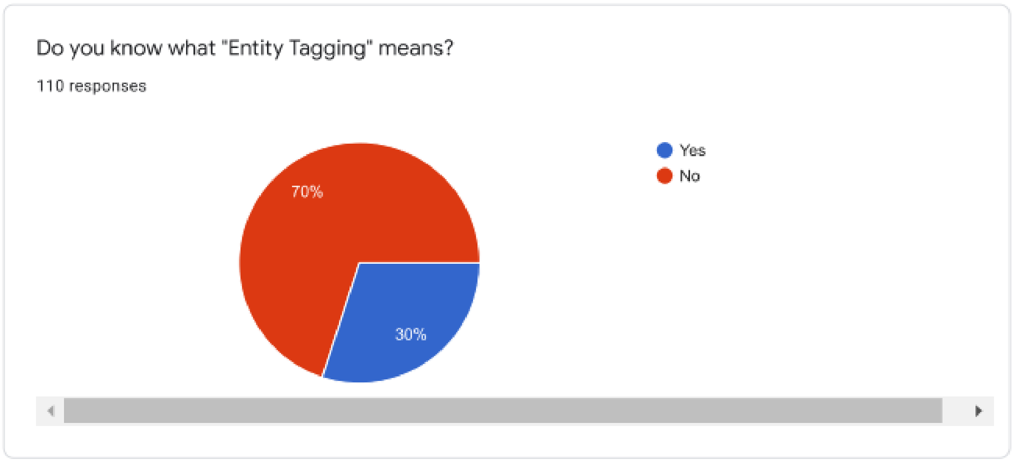

Do you know what model explainability or model interpretability of machine learning models is?

110 responses


Have you used model explainability tools like "LIME" or "SHAP" to improve machine learning model performance of NLP or any other machine learning model before?

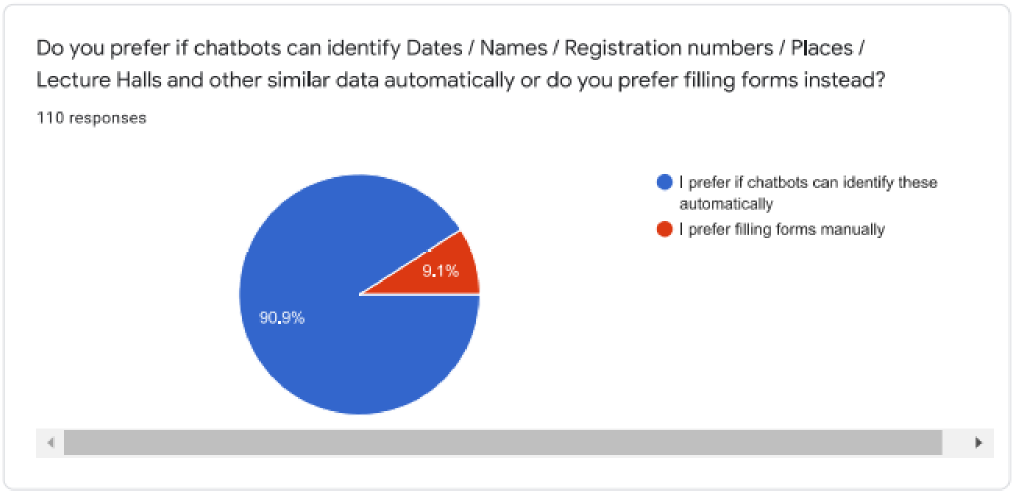
110 responses

Figure A.8: Complete survey form questions and responses – part 8



Do you know what "Entity Tagging" means?

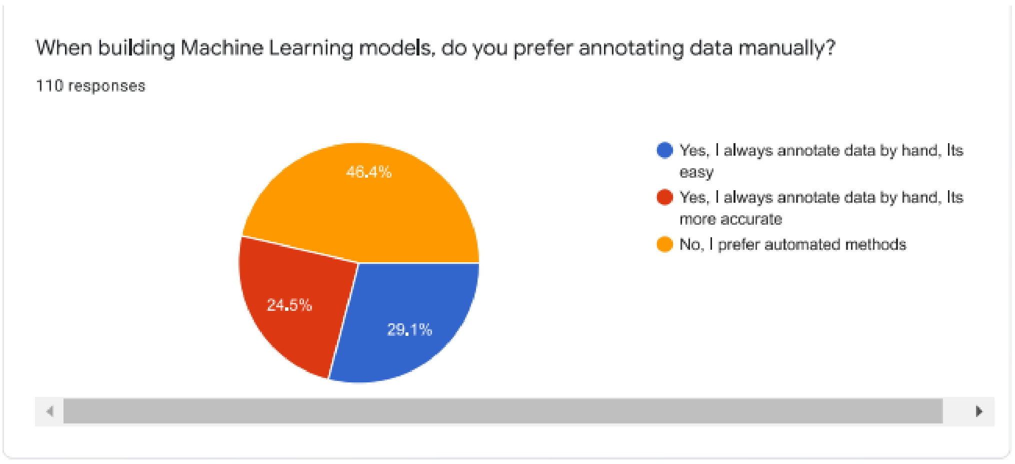
110 responses



Do you prefer if chatbots can identify Dates / Names / Registration numbers / Places / Lecture Halls and other similar data automatically or do you prefer filling forms instead?

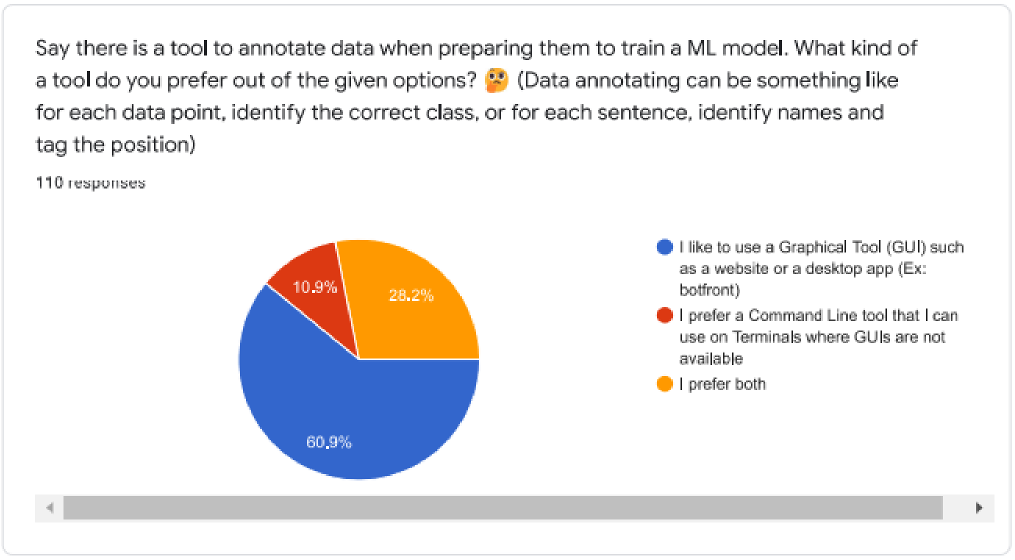
110 responses

Figure A.9: Complete survey form questions and responses – part 9


When building Machine Learning models, do you prefer annotating data manually?

110 responses

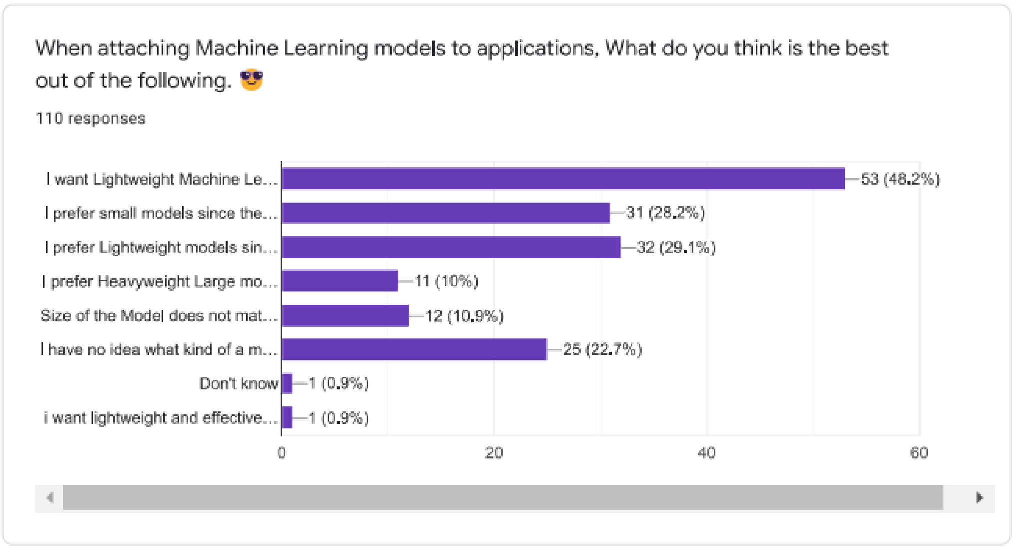
● Yes, I always annotate data by hand, Its easy
● Yes, I always annotate data by hand, Its more accurate
● No, I prefer automated methods


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)

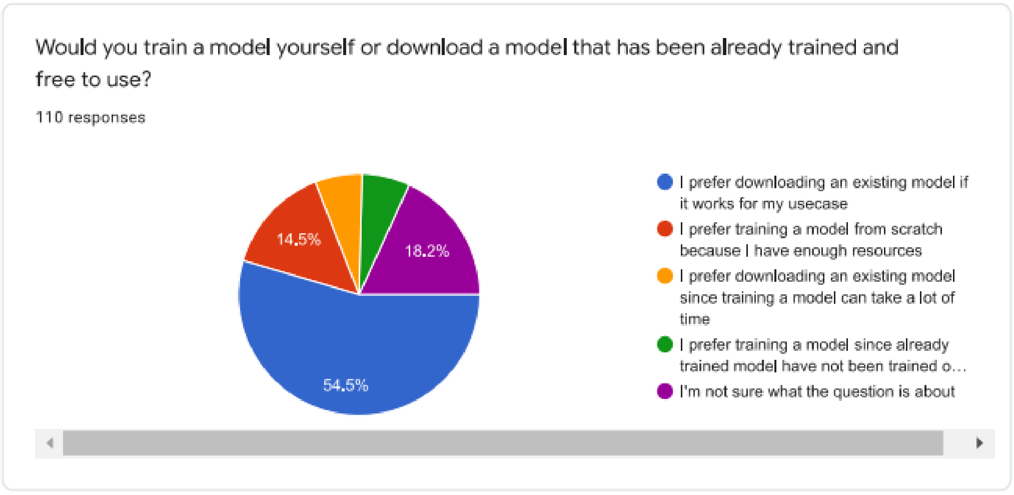
110 responses

● I like to use a Graphical Tool (GUI) such as a website or a desktop app (Ex: botfront)
● I prefer a Command Line tool that I can use on Terminals where GUIs are not available
● I prefer both

Figure A.10: Complete survey form questions and responses – part 10


When attaching Machine Learning models to applications, What do you think is the best out of the following. 😎

110 responses

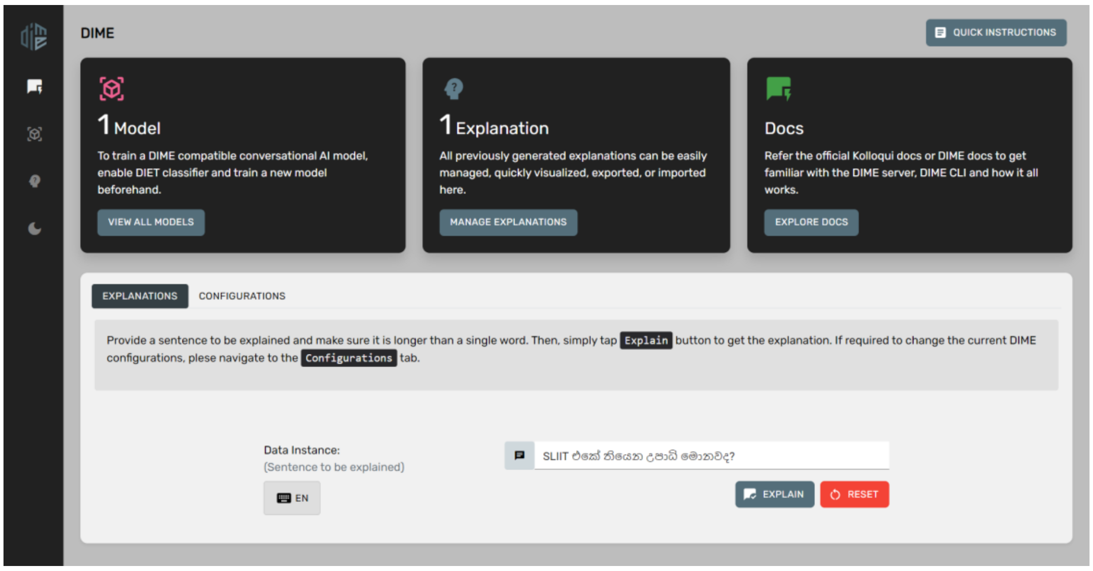

Would you train a model yourself or download a model that has been already trained and free to use?

110 responses

Thank you! 🙏

Figure A.11: Complete survey form questions and responses – part 11

Appendix B: DIME Server User Interfaces



1 Model

To train a DIME compatible conversational AI model, enable DIET classifier and train a new model beforehand.

1 Explanation

All previously generated explanations can be easily managed, quickly visualized, exported, or imported here.

Docs

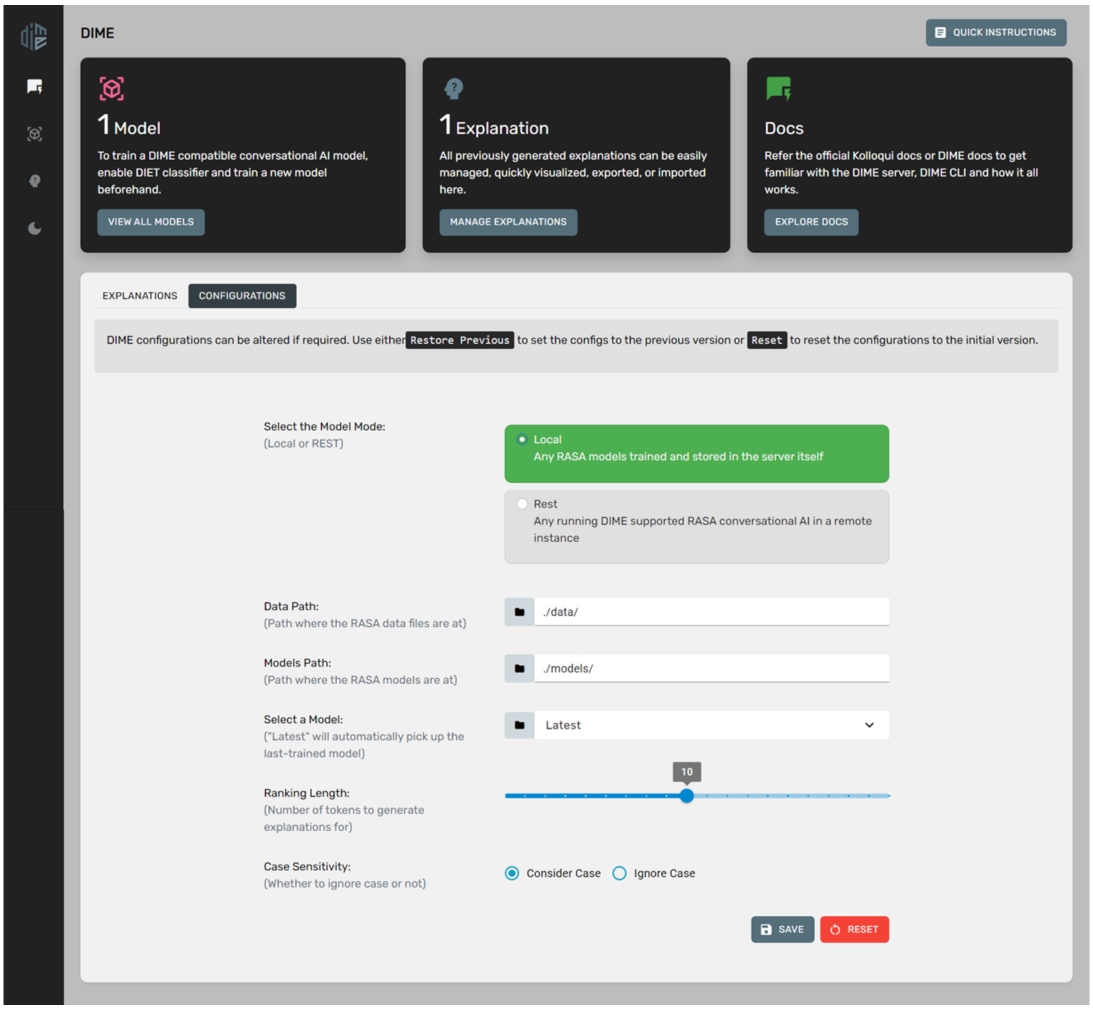
Refer the official Kolloqui docs or DIME docs to get familiar with the DIME server, DIME CLI and how it all works.

Provide a sentence to be explained and make sure it is longer than a single word. Then, simply tap `Explain` button to get the explanation. If required to change the current DIME configurations, plese navigate to the `Configurations` tab.

Data Instance:
(Sentence to be explained)

SLIIT එකේ තියෙන උපාධි මොනවද?

Figure B.1. DIME Server Dashboard User Interface



1 Model
To train a DIME compatible conversational AI model, enable DIET classifier and train a new model beforehand.

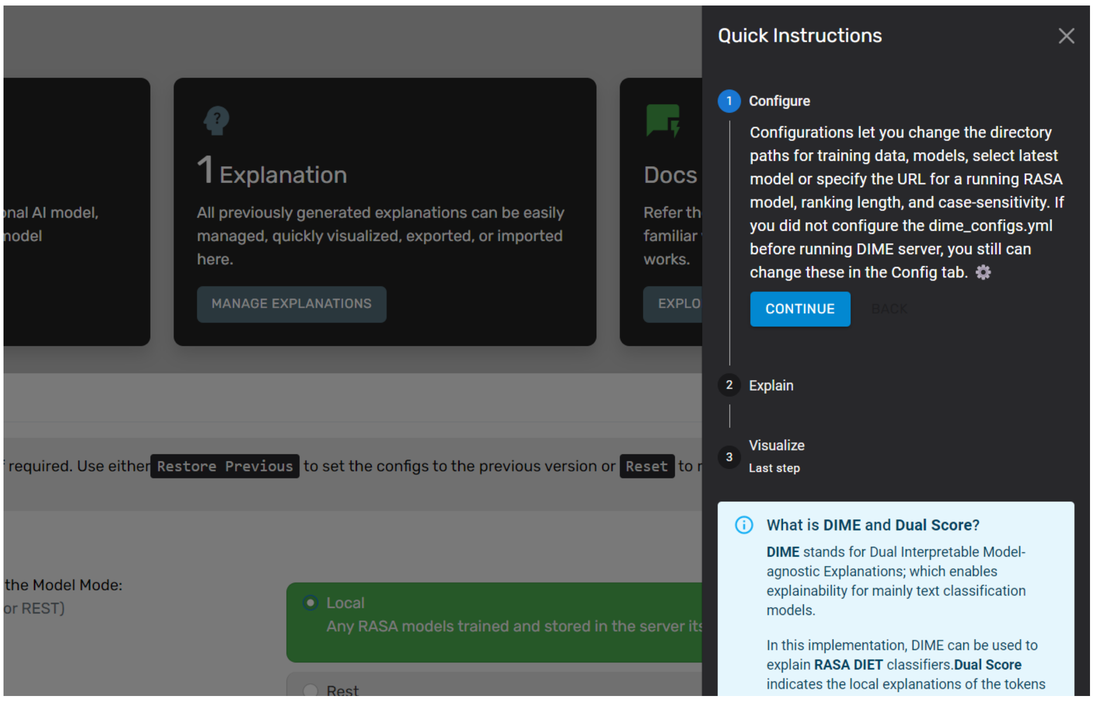
1 Explanation
All previously generated explanations can be easily managed, quickly visualized, exported, or imported here.

Docs
Refer the official Kolloqui docs or DIME docs to get familiar with the DIME server, DIME CLI and how it all works.

DIME configurations can be altered if required. Use either `Restore Previous` to set the configs to the previous version or `Reset` to reset the configurations to the initial version.

Select the Model Mode: (Local or REST)	[✓ Local — Any RASA models trained and stored in the server itself] Rest — Any running DIME supported RASA conversational AI in a remote instance
Data Path: (Path where the RASA data files are at)	./data/
Models Path: (Path where the RASA models are at)	./models/
Select a Model: ("Latest" will automatically pick up the last-trained model)	Latest
Ranking Length: (Number of tokens to generate explanations for)	10
Case Sensitivity: (Whether to ignore case or not)	[✓ Consider Case] Ignore Case

Figure B.2. DIME Server Dashboard User Interface - Configurations Tab


nal AI model,
nodel

1 Explanation

All previously generated explanations can be easily managed, quickly visualized, exported, or imported here.

MANAGE EXPLANATIONS

Docs

Refer th
familiar
works.

EXPLO

required. Use either `Restore Previous` to set the configs to the previous version or `Reset` to r

the Model Mode:
or REST)

Local
Any RASA models trained and stored in the server its

Rest

Quick Instructions

1 Configure

Configurations let you change the directory paths for training data, models, select latest model or specify the URL for a running RASA model, ranking length, and case-sensitivity. If you did not configure the dime_configs.yml before running DIME server, you still can change these in the Config tab.

CONTINUE BACK

2 Explain

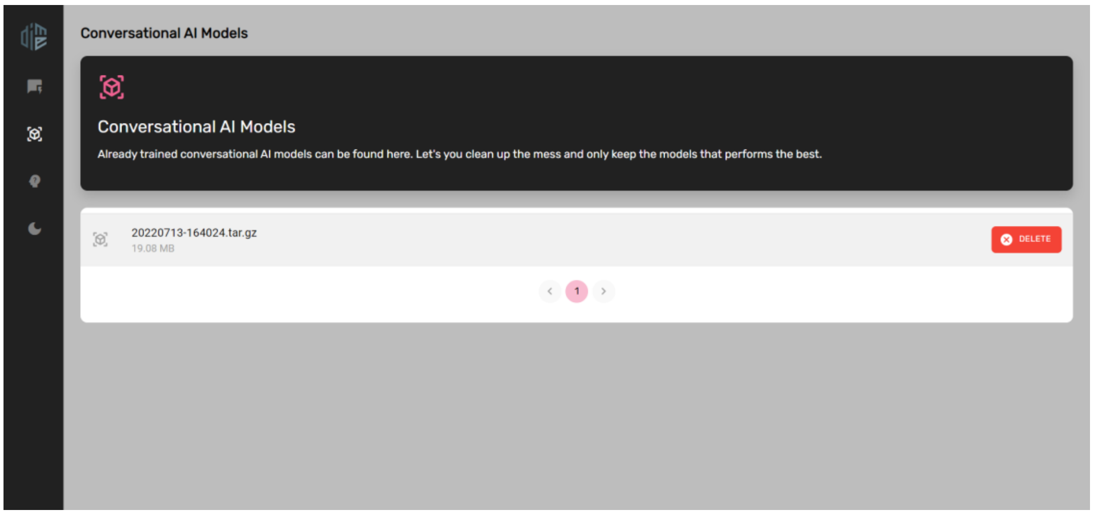
3 Visualize
Last step

What is DIME and Dual Score?

DIME stands for Dual Interpretable Model-agnostic Explanations; which enables explainability for mainly text classification models.

In this implementation, DIME can be used to explain **RASA DIET** classifiers.**Dual Score** indicates the local explanations of the tokens

Figure B.4. DIME Server Dashboard User Interface – Quick Instructions Overlay


Conversational AI Models

Conversational AI Models

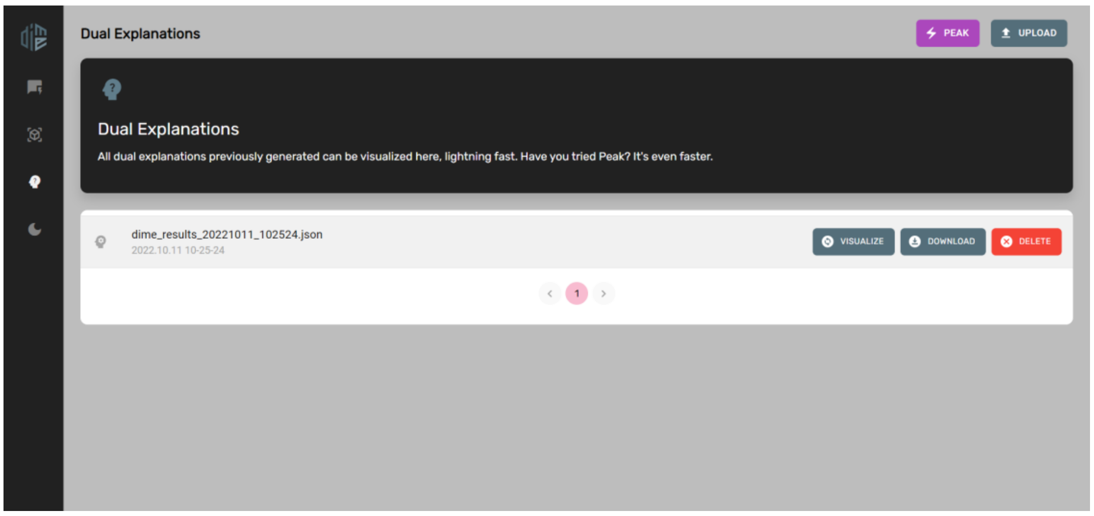
Already trained conversational AI models can be found here. Let's you clean up the mess and only keep the models that performs the best.

20220713-164024.tar.gz
19.08 MB

DELETE

‹ 1 ›

Figure B.3. DIME Server Models User Interface – List of Models

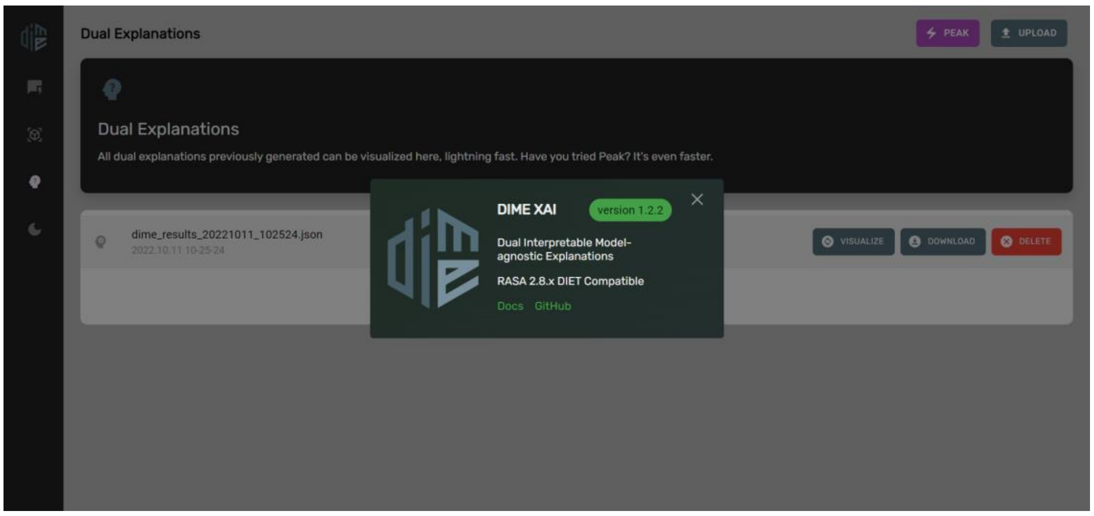


Dual Explanations

All dual explanations previously generated can be visualized here, lightning fast. Have you tried Peak? It's even faster.

dime_results_20221011_102524.json
2022.10.11 10-25-24

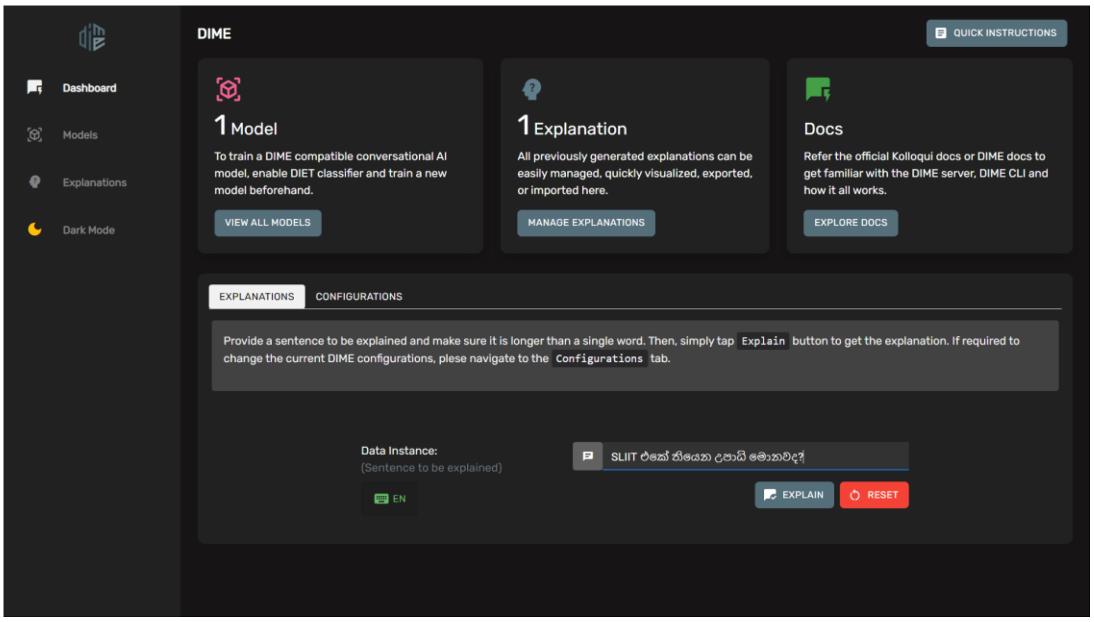
Figure B.6. DIME Server Explanations User Interface – List of Explanations, *Peak* and *Upload* features



Dual Interpretable Model-agnostic Explanations

RASA 2.8.x DIET Compatible

Figure B.5. DIME Server Version Overlay



1 Model

To train a DIME compatible conversational AI model, enable DIET classifier and train a new model beforehand.

1 Explanation

All previously generated explanations can be easily managed, quickly visualized, exported, or imported here.

Docs

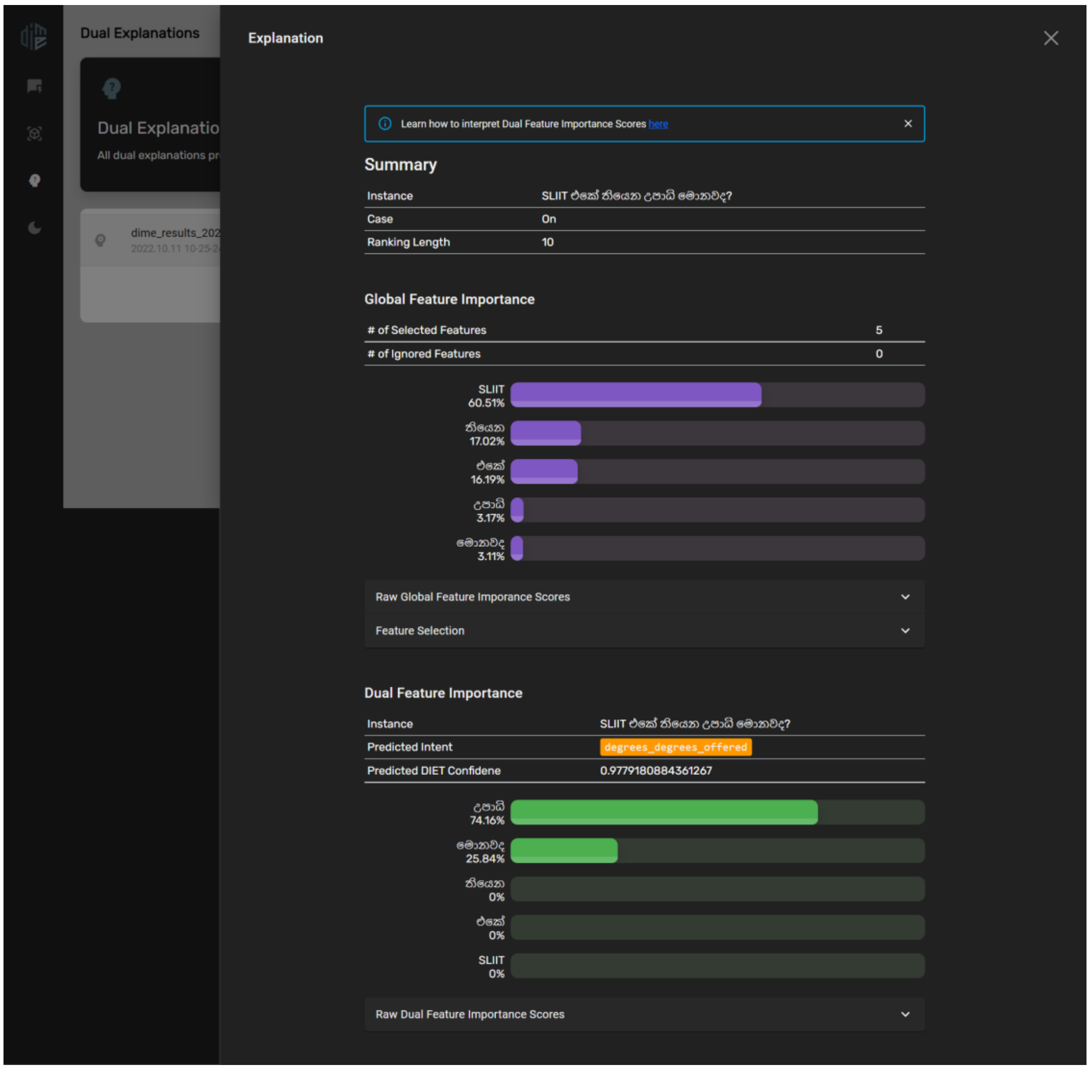
Refer the official Kolloqui docs or DIME docs to get familiar with the DIME server, DIME CLI and how it all works.

Provide a sentence to be explained and make sure it is longer than a single word. Then, simply tap `Explain` button to get the explanation. If required to change the current DIME configurations, plese navigate to the `Configurations` tab.

Data Instance:
(Sentence to be explained)

SLIIT එකේ තියෙන උපාධි මොනවද?

Figure B.7. DIME Server Dark Mode



Learn how to interpret Dual Feature Importance Scores here

Summary

Instance	SLIIT එකේ තියෙන උපාධි මොනවද?
Case	On
Ranking Length	10

Global Feature Importance

# of Selected Features	5
# of Ignored Features	0

SLIIT	60.51%
තියෙන	17.02%
එකේ	16.19%
උපාධි	3.17%
මොනවද	3.11%

Raw Global Feature Imporance Scores

Feature Selection

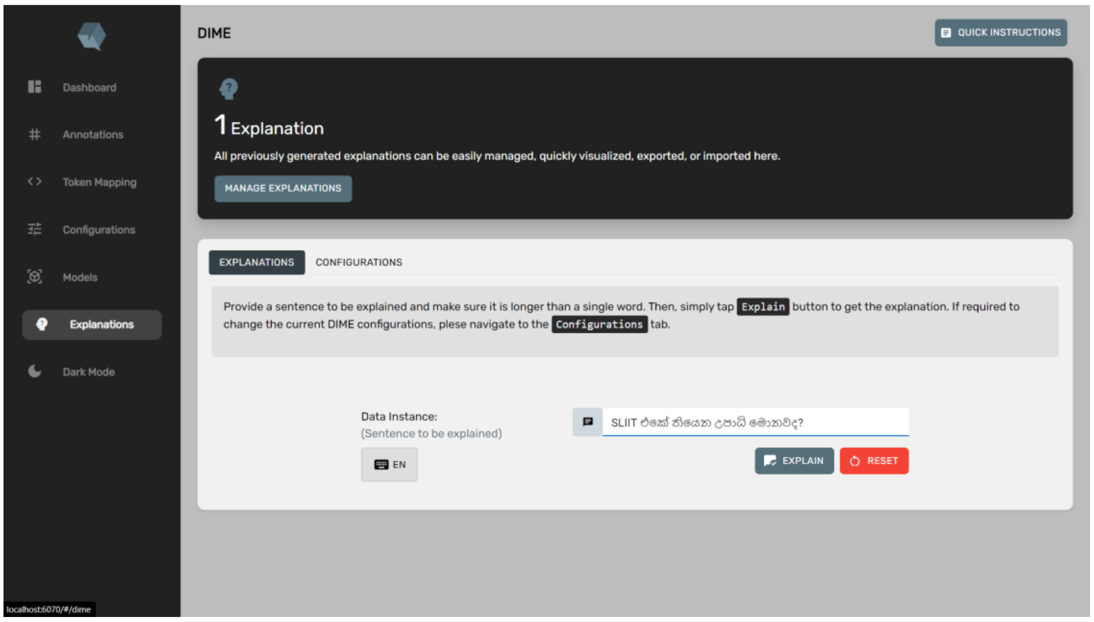
Dual Feature Importance

Instance	SLIIT එකේ තියෙන උපාධි මොනවද?
Predicted Intent	degrees_degrees_offered
Predicted DIET Confidene	0.9779180884361267

උපාධි	74.16%
මොනවද	25.84%
තියෙන	0%
එකේ	0%
SLIIT	0%

Raw Dual Feature Importance Scores

Figure B.8. DIME Server Explanation Visualization



1 Explanation

All previously generated explanations can be easily managed, quickly visualized, exported, or imported here.

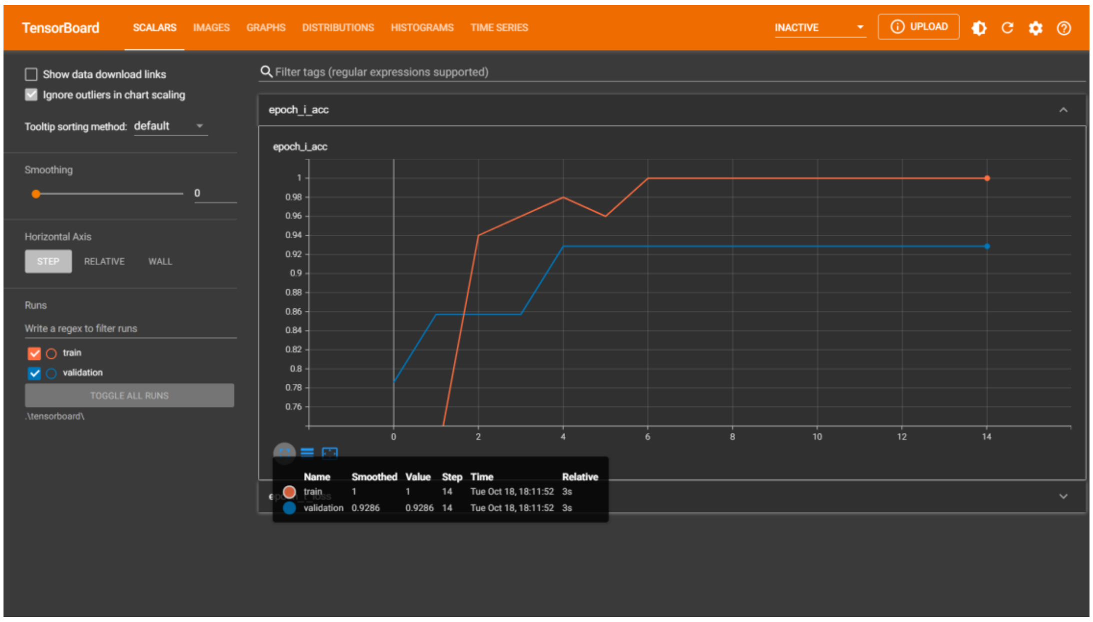
Provide a sentence to be explained and make sure it is longer than a single word. Then, simply tap `Explain` button to get the explanation. If required to change the current DIME configurations, plese navigate to the `Configurations` tab.

Data Instance:
(Sentence to be explained)

SLIIT එකේ තියෙන උපාධි මොනවද?

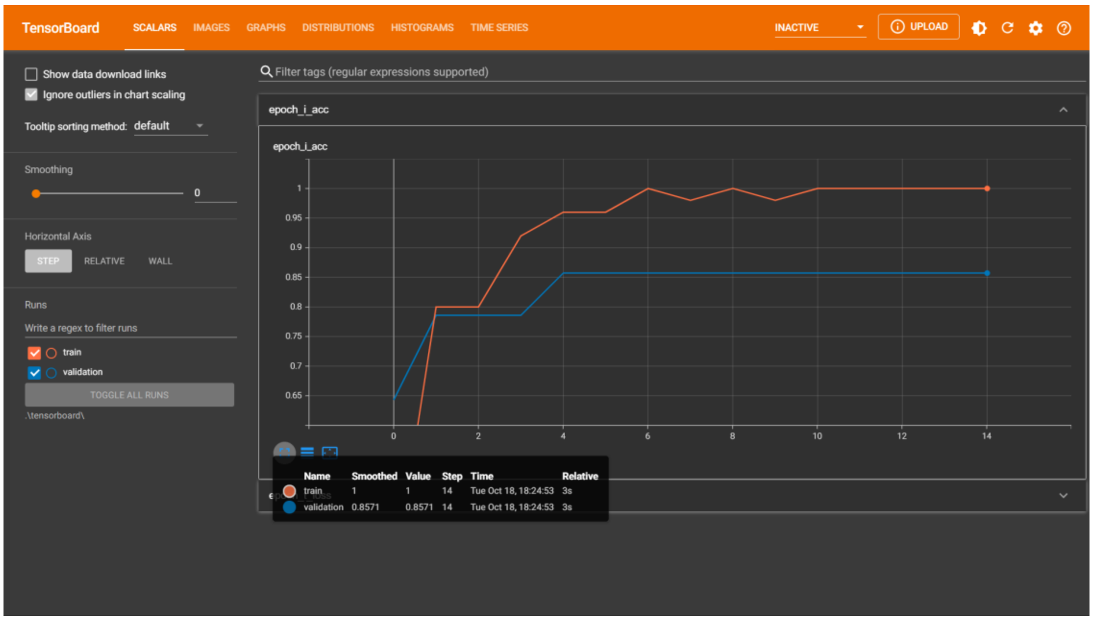
Figure B.9. DIME-XAI UI integrated into Kolloqe Developer Console

Appendix C: DIME Experiment Learning Curves



	Name	Smoothed	Value	Step	Time	Relative
●	train	1	1	14	Tue Oct 18, 18:11:52	3s
●	validation	0.9286	0.9286	14	Tue Oct 18, 18:11:52	3s

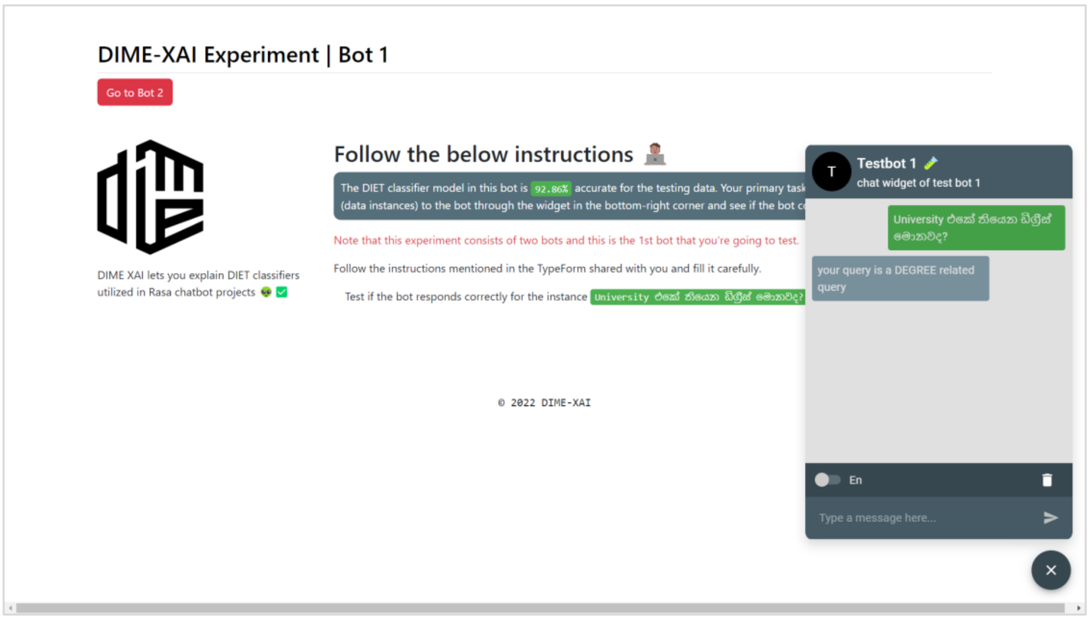
Figure C.1. Intent Classification Accuracy Curve of Bot-1 used for DIME Experiment



	Name	Smoothed	Value	Step	Time	Relative
●	train	1	1	14	Tue Oct 18, 18:24:53	3s
●	validation	0.8571	0.8571	14	Tue Oct 18, 18:24:53	3s

Figure C.2. Intent Classification Accuracy Curve of Bot-2 used for DIME Experiment

Appendix D: DIME Experiment Setup



DIME-XAI Experiment | Bot 1

Go to Bot 2

DIME XAI lets you explain DIET classifiers utilized in Rasa chatbot projects

Follow the below instructions

The DIET classifier model in this bot is 92.86% accurate for the testing data. Your primary task
(data instances) to the bot through the widget in the bottom-right corner and see if the bot c

Note that this experiment consists of two bots and this is the 1st bot that you're going to test.

Follow the instructions mentioned in the TypeForm shared with you and fill it carefully.

Test if the bot responds correctly for the instance `University එකේ තියෙන ඩිග්‍රීස් මොනවද?`

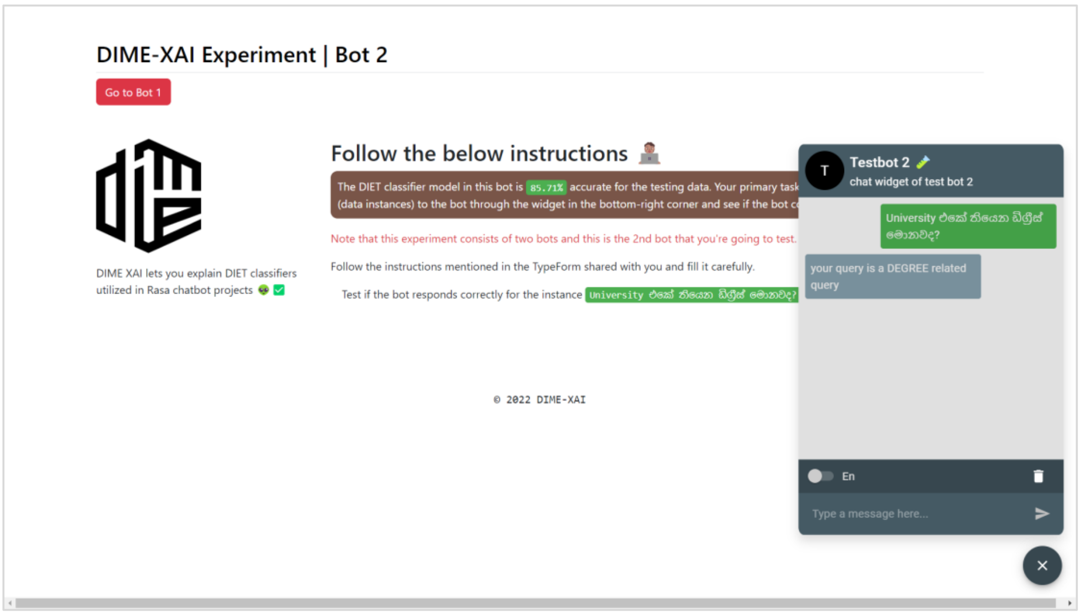
© 2022 DIME-XAI

Testbot 1
chat widget of test bot 1

University එකේ තියෙන ඩිග්‍රීස් මොනවද?

your query is a DEGREE related query

Figure D.1. DIME Experiment Setup of Bot-1



DIME-XAI Experiment | Bot 2

Go to Bot 1

DIME XAI lets you explain DIET classifiers utilized in Rasa chatbot projects

Follow the below instructions

The DIET classifier model in this bot is 85.71% accurate for the testing data. Your primary task
(data instances) to the bot through the widget in the bottom-right corner and see if the bot c

Note that this experiment consists of two bots and this is the 2nd bot that you're going to test.

Follow the instructions mentioned in the TypeForm shared with you and fill it carefully.

Test if the bot responds correctly for the instance `University එකේ තියෙන ඩිග්‍රීස් මොනවද?`

© 2022 DIME-XAI

Testbot 2
chat widget of test bot 2

University එකේ තියෙන ඩිග්‍රීස් මොනවද?

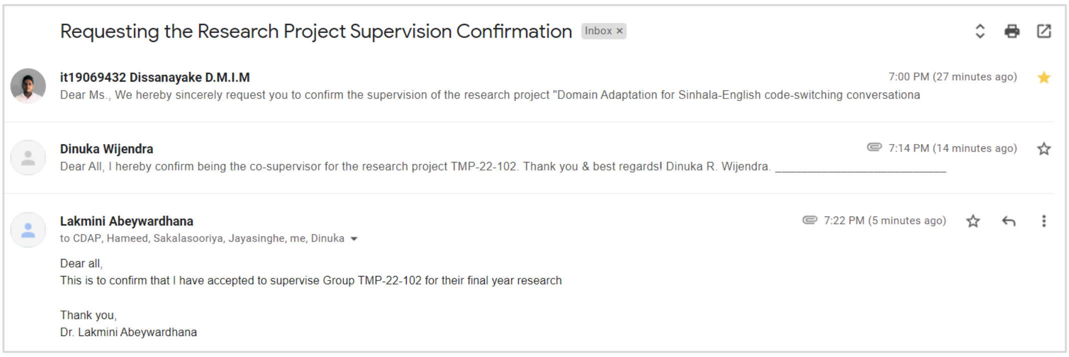
your query is a DEGREE related query

Figure D.2. DIME Experiment Setup of Bot-2

Appendix E: DIME Experiment Observations

DIME experiment observations are available at https://rr6txd9ijim.typeform.com/report/RxIayEXi/OH1gKeQjvUP88Bsu

Appendix F: Supervision Confirmation Emails



Requesting the Research Project Supervision Confirmation Inbox ×

it19069432 Dissanayake D.M.I.M 7:00 PM (27 minutes ago)
Dear Ms., We hereby sincerely request you to confirm the supervision of the research project "Domain Adaptation for Sinhala-English code-switching conversationa

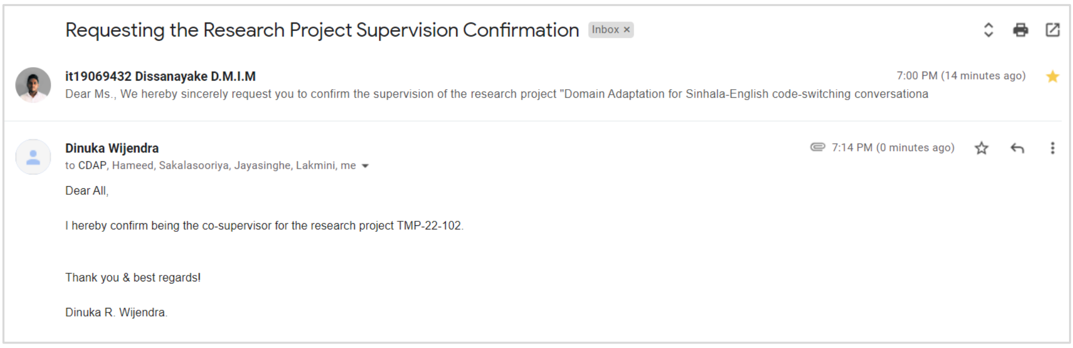
Dinuka Wijendra 7:14 PM (14 minutes ago)
Dear All, I hereby confirm being the co-supervisor for the research project TMP-22-102. Thank you & best regards! Dinuka R. Wijendra. ________________________

Lakmini Abeywardhana 7:22 PM (5 minutes ago)
to CDAP, Hameed, Sakalasooriya, Jayasinghe, me, Dinuka

Dear all,
This is to confirm that I have accepted to supervise Group TMP-22-102 for their final year research

Thank you,
Dr. Lakmini Abeywardhana

Figure F.1. Research project supervision confirmation email



Requesting the Research Project Supervision Confirmation Inbox ×

it19069432 Dissanayake D.M.I.M 7:00 PM (14 minutes ago)
Dear Ms., We hereby sincerely request you to confirm the supervision of the research project "Domain Adaptation for Sinhala-English code-switching conversationa

Dinuka Wijendra 7:14 PM (0 minutes ago)
to CDAP, Hameed, Sakalasooriya, Jayasinghe, Lakmini, me

Dear All,

I hereby confirm being the co-supervisor for the research project TMP-22-102.

Thank you & best regards!

Dinuka R. Wijendra.

Figure F.2. Research project co-supervision confirmation email