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

Ishara Dissanayake
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
isharadissanayake@icloud.com

Shamikh Hameed
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
shamikhhameed@gmail.com

Akalanka Sakalasooriya
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
himesha@outlook.com

Dinushi Jayasinghe
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
jayasinghedt@gmail.com

Lakmini Abeywardhana
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
lakmini.d@sliit.lk

Dinuka Wijendra
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
dinuka.w.sliit.lk

***Abstract*— Natural language processing has become an essential part of modern conversational AIs. However, applying natural language processing to low-resource languages is challenging due to their lack of digital presence. Sinhala is the native language of approximately nineteen million people in Sri Lanka and is one of many low-resource languages. Moreover, the increase in using code-switching: alternating two or more languages within the same conversation, and code-mixing: the practice of representing words of a language using characters of another language, has become another major issue when processing natural languages. Apart from natural language processing, the explainability of opaque machine learning models utilized in conversational AIs has become another prominent concern. None of the existing modern conversational AI platforms supports explainability out of the box and relies on just a performance score such as accuracy or f1-score. This paper proposes a no-code conversational AI platform with a series of built-in novel natural language processing, model evaluation, and explainability tools to tackle the problems of processing Sinhala-English code-switching and code-mixing natural language data and model evaluation in modern conversational AI platforms.**

Keywords—Conversational AI, Natural Language Processing, Code-switching, Code-mixing, Explainable AI

I. INTRODUCTION

In this era of Artificial Intelligence (AI), many top-tier applications and platforms utilize Natural Language Processing (NLP). Almost all domains have adopted NLP in ample ways to perform Natural Language Understanding (NLU), Natural Language Generation (NLG), or both. One such popular application of NLP is chatbots. Advanced chatbots and Conversational AIs have adopted AI to perform NLU and NLG instead of following rule-based or pattern-matching techniques. Most conversational AIs have advanced machine learning models that are constantly trained based on Conversation-Driven Development (CDD), which stands for training dialogue engines on natural human conversations, allowing them to handle complex queries and flawless conversations [1].

However, processing natural language queries can be challenging depending on the language of the training data fed to a conversational AI. For example, although the Sinhala language is one of the two official languages of Sri Lanka, with around 19 million speakers out of the total population of 21 million, it still falls under the category of low-resource languages due to its less digital presence [2], [3]. Most Sri Lankans prefer using the Sinhala-English code-switching typing style, according to a recent survey conducted on 110 people from the public[1]. Processing Sinhala-English code-switched textual data becomes further challenging when compared to processing a low-resource language since there are even fewer resources and NLP tools available, which is one of the main concerns of this paper.

Having multiple equivalent words is one of the many issues in a code-switching corpus [4]. For example, a Sinhala-English code-switching corpus may contain the two similar queries යුනිවර්සිටි එකේ තියෙන උපාධි මොනවද? and University එකේ තියෙන Degrees මොනවද? (*What are the degrees available at the University?*). Here, the words උපාධි and Degrees essentially have the same underlying meaning, and the same applies to University and යුනිවර්සිටි. A conversational AI trained on an equivalent word-rich code-switching dataset will recognize these as distinct words and assign different features, which degrades the overall model performance. The same conversational AI will fail to understand a user query such as යුනිවර්සිටි එකේ තියෙන ඩිග්‍රීස් මොනවද?, as it does not know that ඩිග්‍රීස් is another equivalent word for Degrees and will often discard that as an out-of-vocabulary (OOV) word.

Entity annotating is another significant task as it enables conversational AIs to recognize entities in user queries that helps the decision-making process and drives a conversation. Currently available entity annotating tools included with most conversational AI development frameworks have evolved primarily around English. For example, consider the Sinhala-English code-switched training data instances මේ පුටුවේ price එක කියද? (*What is the price of this chair?*), පුටුවක මිල කියද? (*What is the price of a chair?*), and මේ පුටුව ගන්න කියක් යනවාද? (*How much would it cost to purchase this chair?*). There is no existing entity annotating tool that can identify පුටුවේ (*of the chair*), පුටුවක (*of a chair*), and පුටුව (*the chair*) is the same entity.

In Sinhalese, it is possible to construct words by joining prefixes, root words, and suffixes in a meaningful manner,

[1] The survey is available at https://forms.gle/Ch99hES5zpYyQ9tL8

979-8-3503-9809-0/22/$31.00 ©2022 IEEE

according to the නාම වරනැගිල්ල: Sinhala noun declension rules [5]. A combination of prefixes and base words can have multiple suffixes. For example, the Sinhala word පොත (*book*) can have multiple representations in terms of noun declension, such as පොත් (*the books*), පොතක් (*a book*), පොතට (*to the book*), පොතකට (*to a book*), පොතෙන් (*from the book*), and පොතකින් (*from a book*) which have the same base form පොත් combined with various suffixes. These combinations have almost the same meaning from the perspective of entity annotation. Therefore, this paper proposes a method of annotating entities by assigning entity types to the base form of a specified word.

Although there are numerous conversational AI development frameworks and platforms, only a few of them support low-resource languages such as Sinhala. Many of these frameworks are cloud-based such as Google Dialogflow, Amazon Lex, and Microsoft Language Understanding Intelligent Service (LUIS), and have a graphical user interface (GUI) as the developer console. In addition to that, there are also local conversational AI development frameworks such as Rasa open-source with a command-line interface (CLI) [6]. None of the existing frameworks offers Sinhala typing support within the user interface (UI) or dedicated Sinhala data pre-processing tools essential for development tasks. Although the cloud-based conversational AI frameworks provide an easy-to-use GUI, developers cannot fully configure the process of training machine learning models. In local conversational-AI frameworks, the developers have the trade-off of being limited to a CLI that requires framework-specific expertise.

Most conversational AI frameworks produce advanced machine learning models apt to the trade-off between model explainability and accuracy [7]. These models are not human-interpretable due to their complex and opaque nature and are known as black-box models. Explainable AI (XAI) allows debugging black-box models by explaining if the model looks at the accurate features [8]. In conversational AIs, developers can utilize XAI to debug machine learning models such as intent classification and entity recognition. The existing frameworks do not have built-in support for explainability.

This paper presents Kolloqe, a web-based no-code conversational AI development platform with built-in support for typing Sinhala-English code-switching text anywhere in the GUI, expanding the native language support beyond English. It offers Sinhala-English code-switching-specific NLP tools to map equivalent tokens and annotate entities efficiently through reverse-stemming and text similarity-based entity suggestions. The platform also provides the developers full support to configure the NLU pipeline and eliminates the need to interact with complex backends or CLI interfaces. Training and evaluating machine learning models of conversational AIs is supported within the GUI along with easy-to-configure NLU pipelines and model-specific accuracy and loss curves. The platform also has out-of-the-box support for machine learning model explainability that explains any given query based on feature importance.

II. Literature Review

Although some research has explored Sinhala-English code-switched text processing up to an extent, the number of available resources is still considerably low. N. de Silva [2] has revealed adequate research papers written around Sinhala and Sinhala-English code-switched text processing and was a valuable resource. A few research papers have focused on developing Sinhala chatbots [3], [9]. As in [3], word embedding models such as fastText can be used to increase accuracy and is closer to the objectives of this research. B. Hettige and A. S. Karunananda [9] have developed a Sinhala chatbot, claiming it is the first Sinhala chatbot, incorporating NLP components such as morphological analyzers, Sinhala parsers, and knowledge bases. However, it does not focus on code-switched or code-mixed text data processing. Although [10] embodies training a word2vec model on the University of Colombo School of Computing (UCSC) Sinhala News dataset using the continuous bag of words (CBOW) method, the research paper only mentions elementary text pre-processing techniques such as stop word removal and lemmatization. Overall, none of the research papers mentioned above highlights text pre-processing methods for Sinhala-English code-switched or code-mixed textual data.

Although [11], [12], and [13] have utilized Sinhala-English code-switched and code-mixed text data for word-level language detection, code-switching point detection, and pre-training machine learning models, they do not significantly accentuate proper text pre-processing techniques for code-switched textual data. According to [11], code-switching is a challenge in modern Sinhala text pre-processing, and it has introduced a dictionary mapping method to standardize the representation of Sinhala characters written using the English alphabet. This technique is somewhat similar to the method proposed by this paper in implementing the code-switching keyboard interface. Although [3] and [10] focus on training word embedding models for Sinhala, they do not consider Sinhala-English code-switching text processing. Although [3] mentions training fastText models for chatbots, it does not emphasize significant changes to the data pre-processing steps.

Only a few research papers are available on automated and semi-automated text annotation tools. The Automated Named Entity Annotation (ANEA) tool [14] inherits knowledge from Wiktionary (an online dictionary). If Wiktionary does not contain the required word, this tool cannot guess the word entity. Brat Rapid Annotation Tool (BRAT) [15] is another tool for auto-annotating entities that introduces the semantic class disambiguation algorithm. GATE Teamware [16] mainly focuses on collaborative annotation, which is not in the scope of the entity annotation method proposed in this paper. YEDDA [17] is another tool that provides an entity recommendation method during the text annotation process by the maximum matching algorithm, which is a text segmentation algorithm. None of these research papers suggests an optimized entity annotating approach or similar work for Sinhala-English bilingual text data.

Recent XAI research that has introduced Local Interpretable Model-agnostic Explanations (LIME) [18], Shapley additive explanations (SHAP) [19] and explainable AI (XAI) libraries such as Explain Like I'm Five (ELI5) [20] have discussed various methods of implementing explainability for both NLP and non-NLP machine learning models. The LIME paper presents how to generate local explanations in a model-agnostic manner, taking a trained model and relevant set of classes as arguments to the LIME and training an explainable local surrogate model. SHAP paper discusses a slightly different approach while improving upon LIME, addressing its limitations. SHAP can calculate local and global feature importance inspired by shapely values found in game theory [21]. However, SHAP calculates the

local feature contributions first and then finds the feature importance globally by summing the absolute SHAP values of each prediction [19]. The python XAI library ELI5 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) with random words (noise), which is closely related to one of the XAI approaches introduced in this paper. Although some researchers have attempted to calculate the global feature importance based on individual local feature importance scores, none of the referred research has mentioned deriving local feature importance using global-level feature importance, which is the XAI approach introduced the in this paper.

Rasa Core and Rasa NLU are open-source python libraries specializing in building conversational AI software. The core objective of Rasa is to make machine-learning-based dialogue management and NLU accessible to the average developers [6]. However, Rasa does not have a GUI and is limited to a CLI and yet another markup language (YAML) file-based conversational AI development. Although the objectives of Rasa align with this research, it does not include a built-in GUI-based developer console, Sinhala typing support, Sinhala NLP tools, or machine learning model explainability.

III. Methodology

The Sinhala-English code-switching text dataset utilized in this research contains extracted information from the official website of the Sri Lanka Institute of Information Technology (SLIIT) and publicly available documents on the same website[2]. The collected data were pre-processed and augmented to generate a high-quality Sinhala-English code-switching textual dataset. The pre-processed dataset used to build and train the NLP tools presented in this paper consists of 770 training data examples related to the education domain and falls under 72 distinct classes (also known as intents in conversational AI terminology).

The following sub-sections elaborate on prominent features offered by the Kolloqe chatbot development platform introduced as the primary outcome of the research.

A. *Rule-based code-switchable keyboard interface*

Kolloqe has a built-in Sinhala-English code-switchable keyboard interface embedded into all input fields in its developer console and the chat widget attached for testing already trained chatbot models. Sinhala-English code-switching typing support enables developers and conversational AI users to interact with the UI more efficiently and expands the native language support beyond English. It eliminates the need for complicated Sinhala Unicode keyboard layouts, third-party keyboard software, or online Sinhala keyboard web applications by enabling native support for typing efficiently in English, Sinhala, or both.

The core of the keyboard interface implementation relies on a rule-based character mapping algorithm, which consists of six rules to convert an English character or a combination of characters into the respective Sinhala representation. (1) non-joining characters mapping, (2) special characters mapping, (3) "Rakaranshaya" mapping, (4) consonant + vowel mapping, (5) pure-consonants mapping, and (6) pure-vowels mapping are the six rules given in their order of execution. To type in Sinhala, a user must follow a character map developed based on phonetics and type the relevant English characters or phrases that map into Sinhala characters primarily based on the sound. Fig. 1 depicts the character map that maps combinations of English characters or individual characters to the respective Sinhala characters based on their phonetic similarity. Table 1 visualizes how to write a Sinhala word according to the character map and how the ruleset converts it into the respective Sinhala word.

[2] https://www.sliit.lk/downloads/

B. *Sinhala-English Equivalent Token Mapping*

Sinhala-English Equivalent Token Mapping (SEETM) technique takes a set of user-defined equivalent word maps and replaces all the mapped words (also known as tokens in NLP terminology) with the equivalent base word as the first step of the chatbot training process in Kolloqe. SEETM utilizes global regular expression patterns and string replacements to replace the equivalent tokens with the base words specified in the maps for all training data instances. SEETM tokenizer is a custom machine learning NLU pipeline component in Kolloqe that loads the user-defined token maps at run-time and dynamically replaces the equivalent words with the base word just before the tokenization step by passing training data examples to the SEETM mapper component. For example, the SEETM mapper maps the equivalent tokens උපාධි, degrees, ඩිග්‍රී, and ඩිග්‍රීස් to the base token උපාධි.

SEETM also suggests the Sinhala representation of valid English words while generating the token maps, which

a අ	aa ආ	A ඇ	AA ඈ	Aa ඇ	ae ඇ	i ඉ	ii ඊ	ie ඊ	e එ	ee ඒ
ea ඒ	ei ඒ	I ඓ	E ඓ	o ඔ	oo ඕ	oe ඕ	O ඕ	au ඖ	u උ	uu ඌ
a 	aa ා	A ැ	AA ෑ	Aa ෑ	ae ෑ	i ි	ii ී	ie ී	e ෙ	ee ේ
ea ේ	ei ේ	E ෛ	I ෛ	o ො	oo ෝ	oe ෝ	O ෞ	au ෞ	u ු	uu ූ
k ක්	kh ඛ	K ඛ	g ග්	G ඝ්	gh ඝ්	X ඞ	nng ඟ්	ch ච	c ච	Ch ඡ
C ඡ	j ජ්	t ට	T ඨ	d ඩ	D ඪ	N ණ්	nnd ඬ	th ත්	Th ථ	dh ද්
Dh ධ	n න්	nndh ඳ	J ඣ	p ප්	P ඵ	ph ඵ	b බ	B භ	bh භ	m ම
mmb ඹ	Y ය්	y ය්	r ර්	l ල්	KN ඤ්	v ව	w ව	sh ශ	Sh ෂ	s ස්
h හ	L ළ	Lu ළු	f ෆ්	GN ඦ්	\\r ර්	x ං	H ඃ	R ඍ	ruu ූa	ru ුa

Fig. 1. English to Sinhala character map: (1) Gray – pure vowels, (2) Green – vowel suffixes, (3) Blue – pure consonants, (4) Red – non-joining characters, (5) Yellow – special characters.

TABLE I. Character Mapping Rule Execution Breakdown

Rule Execution Order	Rule Execution Breakdown for "aakramaNaya"	
	Character Mappings	***Output***
Non-joining Character Mapping	skipped as there are no non-joining characters	aakramaNaya
Special Characters Mapping	skipped as there are no special characters	aakramaNaya
"Rakaranshaya" Mapping	kra → ක් + ර් + අ → ක්‍ර	aaක්‍රmaNaya
Consonant + Vowel Mapping	ma → ම් + අ → ම Na → ණ් + අ → ණ ya → ය් + අ → ය	aaක්‍රමNaya aaක්‍රමණya aaක්‍රමණය
Pure Consonants Mapping	skipped as there are no pure consonants left	aaක්‍රමණය
Pure Vowels Mapping	aa → ආ	ආක්‍රමණය

increases the overall efficiency of the equivalent token mapping process. Suggestions are generated based on the phonetics of a given English word. For example, when a developer adds the word Degree to a token map, SEETM converts the word Degree to its phonetic representation (dɪˈgri) based on International Phonetics Alphabet (IPA). The IPA mappings are done based on the Carnegie Mellon University (CMU) Pronouncing Dictionary, an open-source pronouncing dictionary [22]. As the next step, SEETM converts the IPA mappings to the relevant Sinhala characters (ඩිග්‍රී) based on the IPA to Sinhala character map depicted in Fig. 2. Overall, SEETM allows Kolloqe to handle OOV issues caused by equivalent tokens at the chatbot inference time by effectively mapping similar words to a known base word for both English and Sinhalese.

C. Sinhala and English entity annotating

Sinhala-English Entity Annotator (SIENA) is the entity annotating tool bundled with Kolloqe that enables efficient Sinhala-English code-switching entity annotations in the developer console. SIENA provides (1) entity suggestions during the annotation task based on the previously annotated data for both Sinhala and English and (2) auto-annotation support that automatically annotates words with the same underlying base form all at once. At annotation time, SIENA employs stemming and converts a specified word to its base form by breaking it down into a combination of vowels and consonants and removing suffixes by cross-checking them with a predefined Sinhala suffixes list extracted from the book බසක මහිම (*Basaka Mahima*) [5] and assigns the entity to it. Stemming is the process of converting a specified word into its base form without considering its morphological information. Since SIENA supports both Sinhala and English, it utilizes Porter Stemmer [23] to stem English words or phrases. SIENA knowledge base, the primary component of the tool, keeps track of previously annotated entities and base forms. Then the SIENA introduces a reverse-stemming approach that inspects the base form of a selected word for tagging and suggests entities if there are existing annotated entities for the same base form in the knowledge base.

The reverse-stemming approach can often capture slightly different base forms to the desired base form of some words due to (1) spelling mistakes, (2) variations of Sinhala typing patterns, and (3) stemming algorithm limitations. Since varying base forms can negatively affect the ranking of the entity suggestions list, the SIENA employs two word-similarity algorithms, cosine-similarity, and bi-grams, to solve the issue. SIENA converts words to vector representations by counting the distinct letters against the Sinhala and English alphabets to calculate the cosine similarity. The bi-gram similarity is calculated by dividing a phrase into sequences of the length of two items and getting the portion of equally matched elements. SIENA assigns an overall similarity score calculated by taking the average of cosine and bi-gram similarity for each entry in the knowledge base against a selected phrase to render a list of entity suggestions in the descending order of the similarity score. Chatbot developers that utilize Kolloqe can efficiently annotate both Sinhala and English entities manually with the help of entity suggestions provided by SIENA or automatically using the auto annotator.

eɪ ඒ	aɪ අයි	aʊ අව්	ɔ අව්	ɔɪ ඕ	oʊ ඕ	æ ඇ		
ɪ ඊ	u ඌ	ɪ ඉ	ʊ උ	ə අ	ɛ එ	ɝ අර්	æ ඇ	ɑ ආ
tʃ ච්	dʒ ජ්	p ප්	b බ්	t ට්	d ඩ්	k ක්	g ග්	θ ත්
ð ද්	f ෆ්	v ව්	s ස්	z ස්	ʃ ශ්	ʒ ෂ්	w ව්	m ම්
n න්	ŋ න්ග්	ɹ ර්	r ර්	j ය්	h හ්	ɬ ල්	l ල්	

Fig. 2. IPA to Sinhala character map: (1). Blue – diphthongs, (2). Green – monophthongs, and (3) Gray – consonants.

D. NLU pipeline configuration, machine learning model training, and evaluation

The Kolloqe developer console provides a set of checkboxes and dropdowns in the structure of a simple Hyper-Text Markup Language (HTML) form to enable or disable the NLU pipeline and policy components. An NLU pipeline comprises NLP and machine learning components responsible for converting unstructured training data examples and queries into intents and entities, while policies are pipeline components that predict actions to take in each step of a conversation [6]. The developer console streamlines the NLU pipeline configuration process by self-ordering the pipeline components, avoiding spelling and file structural mistakes frequent in traditional backend development environments where developers must write configuration files manually, and take care of the backend framework specifics, including executing repetitive CLI commands. With Kolloqe, developers do not require framework-specific knowledge and only need just-enough machine learning knowledge on which pipeline components to choose for better performance.

The GUI allows triggering conversational AI training with a single click, taking care of executing a series of bash commands behind the scenes on behalf of the developers. Each training request is attached with a unique training request identity (ID) and sent to the Kolloqe backend API that runs asynchronously. The backend collects the training requests and puts them in an in-memory process queue, which means that the developers can freely navigate the developer console and perform other tasks while a model is training and can abort an ongoing training request at their convenience. The developer console automatically shows the accuracy and loss scores for each trained conversational AI model, along with the ability to view the accuracy and loss curves for all training epochs. It also provides an interface to manage trained models.

E. Dual Interpretable Model-agnostic Explanations

Kolloqe has natively enabled conversational AI machine learning model explainability through Dual Interpretable Model-agnostic Explanations (DIME), a novel XAI approach introduced in this paper that combines both global and local feature importance that can explain how any text classification model works. DIME requires an already trained text classification model, the training dataset of the text classifier, and any text query that requires explanations. For the provided text query, DIME calculates the global and dual feature importance at its core, where dual feature importance is a new term introduced in this paper that refers to the local feature importance derived using global scores calculated beforehand.

DIME first retrieves individual model confidence for the training data examples in the original dataset and aggregates the scores as overall model confidence. Then for each distinct word in the text query to be explained, DIME calculates the overall model confidence by removing the word from every

training data example in the dataset and feeding the altered dataset as the model input. For a given unique word, DIME assigns a global feature importance score as the difference between the overall model confidence for the original and altered datasets. This approach is also known as the permutation feature importance calculation. DIME utilizes the global feature importance as a feature selection score and prunes globally unimportant words present in the query to be explained. It is also possible to set a ranking length: the number of features to select as input, and for the English language-based datasets, developers can toggle the case sensitivity if required.

After the global feature selection, DIME starts the dual feature importance calculation by first finding the model confidence score for the original text query and then it removes only the selected features separately from the text query and provides the altered text queries as the model input. For each word, DIME assigns the decrement in the confidence score as the dual feature importance of the word. Since DIME iterates through the training dataset n+1 times for n number of features where n is the ranking length, the execution time increases proportionally to the number of features selected. Thus, maintaining a ranking length of around ten is recommended for an average explanation job. However, the ranking length may vary depending on the number of training data examples and unique words in a given dataset.

DIME gives the global and dual feature importance scores calculated as the output, along with easy-to-interpret probability scores and visualizes the results in the Kolloqe developer console with horizontal bar charts. The implementation of DIME integrated with Kolloqe is specific to the Dual Intent-Entity Transformer (DIET) intent classification models available in the Rasa framework [24]. However, the core concept of DIME expands and applies to any text classifier that can output the model confidence.

IV. Results and Discussion

The overall research outcome is a web-based code-less chatbot development platform that comprises tools for native Sinhala-English code-switching typing anywhere in the UI, equivalent token-based OOV token handling, Sinhala-English annotation suggestions and auto annotating entities, code-less NLU pipeline configuration, model training, and evaluation support, and chatbot model explainability.

The Sinhala-English code-switching keyboard interface in Kolloqe runs entirely in the front end and converts the text in real time, eliminating the need for additional backend processing and network latency. It has a linear time complexity (O(n)) as the worst case since it relies on string replacement, where n is the length of the input string. Additionally, the authors of this paper have released the developed input component as an open-source react package, which is publicly available in the Node package manager (NPM)[3]. It allows react-based web apps to expand the typing support beyond English in the front end, providing the Sinhala audience with a native way to interact with the web apps without having unnecessary external keyboard interfaces. Although the users of the keyboard interface must know the correct character mappings beforehand, most character mappings have a phonetic-based relationship, minimizing the room for human error. The authors have tested the intuitiveness of the keyboard using a human-grounded experiment with ten testers by dividing the testers randomly into two groups where the first group used the introduced Kolloqe keyboard, and the second group used Helakuru, a widely utilized third-party Sinhala keyboard. Each group was given eleven Sinhala words to type based on their intuition of phonetic-based character mappings without actually exposing the mappings beforehand. Based on the observations, the Kolloqe Sinhala-English code-switchable keyboard demonstrated a 90.91% success rate within 16.52 minutes in contrast to the 83.64% success rate delivered by Helakuru, where users spent 22.23 minutes. Based on the observations, it is evident that the introduced keyboard interface in this paper is intuitive and efficient.

The authors have demonstrated that the SEETM approach allows machine learning models utilized in chatbots to handle OOV issues caused by equivalent tokens by mapping unseen words in a dataset to a known base word. The authors conducted a human-grounded experiment with five randomly selected testers by providing them with two chatbots, (1) a chatbot with SEETM enabled and (2) a chatbot without SEETM that shared the same training dataset and machine learning pipeline. The testers were instructed to ask the same five questions from both chatbots and observe the responses. Based on the observations, 64% of the SEETM-enabled chatbot responses were accurate, whereas the chatbot without SEETM gave only 24% correct answers. However, depending on the nature of the conversational AI training dataset, the set of equivalent token maps may differ. Developers are responsible for creating the token maps that better suit the dataset before training the chatbot models in Kolloqe.

The auxiliary IPA-based SEETM technique introduced in this paper resulted in 100 exact matches out of 224 English tokens in the evaluation dataset, yielding a 44.65% exact-match rate. The evaluation process utilized edit distance (Levenshtein distance) to measure the dissimilarity between the generated suggestions and the expected output. The overall evaluation dataset yielded an average edit distance of 1.19, which indicate that the algorithm can breed suggestions very close to the expected word in terms of similarity that may contain one to two incorrect or missing characters on average.

The authors evaluated SIENA, the entity annotation and suggestion tool in Kolloqe, in two ways. The first method evaluated the base word identification ability of SIENA for five entities selected from the dataset mentioned in the methodology section by averaging the base word identification accuracy for each entity obtained by dividing the number of accurately annotated words by the total number of all potentially taggable words. The evaluation yielded an overall accuracy of 57.10% for base word identification. The second evaluation was a human-grounded experiment for quantifying the efficiency of the SIENA tool, which incorporated four testers annotating five entities manually and using SIENA taken from the dataset previously mentioned. Annotating entities without SIENA took 3.08 minutes, whereas annotating with SIENA took only 2 minutes. Thus, it is evident that SIENA is nearly 54% faster than the manual annotation process. i.e., Kolloqe provides 1.5 times faster entity annotation.

The dual feature importance score given by DIME, the XAI technique integrated with Kolloqe, is more consistent when compared to well-known existing XAI techniques such as LIME [18]. In particular, extensive testing on Unicode text instances revealed that LIME does not work well with

[3] https://www.npmjs.com/package/@kolloqe/input

Unicode-based inputs such as Sinhala and cannot adequately tokenize a given Unicode query, making it impossible to integrate it with chatbot development platforms that support non-English Unicode-based languages. The authors evaluated the explanation generation in Kolloqe using another human-ground experiment by crafting two chatbots using a relatively small dataset that consisted of 54 training examples and 14 validation examples. However, one of the datasets contained highly biased data and the chatbot utilized that dataset (bot-1) had an accuracy of 92.86%. The second chatbot with the low-biased dataset (bot-2) had 85.71% accuracy. The authors then randomly selected five testers and instructed them to identify the best model without exposing them to explanations, where everyone picked bot-1 due to its high accuracy. Then the testers were given the text instance "University එකේ තියෙන ඩිග්‍රීස් මොනවද?" (*What are the degrees available at the University?*) and asked which word comprises the highest importance for intent classification, solely based on their intuition. Finally, the authors disclosed the explanation generated by each bot for the previous query using Kolloqe and asked them to re-select the best chatbot, and the testers selected bot-2 as the best model. It is noteworthy that in the explanations, bot-1 revealed University as the most important feature, whereas the bot-2 revealed ඩිග්‍රීස් as the most important word, which is the same feature selected by testers before seeing explanations, solely based on intuition. The observations exemplify that Kolloqe provides a reliable model validation and debugging technique compared to performance score-based model validation.

V. Conclusion

In this paper, the authors have introduced Kolloqe, a chatbot development platform that comprises a no-code developer console that does not require machine learning and backend-development expertise and extends native language support from English to Sinhala in the front end. The platform has built-in novel NLP tools for Sinhala-English code-switching training data pre-processing, machine learning model evaluation, and model explainability, addressing limitations in currently available chatbot development platforms in the areas of code-switching data processing and model evaluation. Kolloqe supports both cloud-based and local deployments, and the authors plan to release Kolloqe as a cloud-based proprietary chatbot development platform. Individual components integrated with Kolloqe, including SEETM[4], SIENA[5], DIME[6], and a simplified version of the developer console[7], are available and actively maintained as open-source python packages in the Python Package Index (PyPI) as an attempt to make the research and implementations openly accessible to a broader audience. Overall, Kolloqe and its components are undergoing constant advancements, and their current state is never a "finished" state.

References

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

[2] N. de Silva, "Survey on publicly available sinhala natural language processing tools and research," *arXiv preprint arXiv:1906.02358*, 2019.

[3] B. Gamage, R. Pushpananda, and R. Weerasinghe, "The impact of using pre-trained word embeddings in Sinhala chatbots," in *2020 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2020: IEEE, pp. 161-165.

[4] K. N. Lam, F. A. Tarouti, and J. Kalita, "Creating lexical resources for endangered languages," *arXiv preprint arXiv:2208.03876*, 2022.

[5] D. Jayaratna, *Basaka mahima*. Sumitha Publication.

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

[7] 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.

[8] 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.

[9] B. Hettige and A. S. Karunananda, "First Sinhala chatbot in action," *Proceedings of the 3rd Annual Sessions of Sri Lanka Association for Artificial Intelligence (SLAAI), University of Moratuwa*, vol. 13, 2006.

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

[11] A. Kugathasan and S. Sumathipala, "Standardizing sinhala code-mixed text using dictionary based approach," in *2020 International Conference on Image Processing and Robotics (ICIP)*, 2020: IEEE, pp. 1-6.

[12] I. Smith and U. Thayasivam, "Sinhala-English Code-Mixed Data Analysis: A Review on Data Collection Process," in *2019 19th International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2019, vol. 250: IEEE, pp. 1-6.

[13] I. Smith and U. Thayasivam, "Language Detection in Sinhala-English Code-mixed Data," in *2019 International Conference on Asian Language Processing (IALP)*, 2019: IEEE, pp. 228-233.

[14] A. Zhukova, F. Hamborg, and B. Gipp, "ANEA: Automated (Named) Entity Annotation for German Domain-Specific Texts," *arXiv preprint arXiv:2112.06724*, 2021.

[15] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. i. Tsujii, "BRAT: a web-based tool for NLP-assisted text annotation," in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 102-107.

[16] K. Bontcheva *et al.*, "GATE Teamware: a web-based, collaborative text annotation framework," *Language Resources and Evaluation*, vol. 47, no. 4, pp. 1007-1029, 2013.

[17] J. Yang, Y. Zhang, L. Li, and X. Li, "YEDDA: A lightweight collaborative text span annotation tool," *arXiv preprint arXiv:1711.03759*, 2017.

[18] 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.

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

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

[21] 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.

[22] "The CMU Pronouncing Dictionary." http://www.speech.cs.cmu.edu/cgi-bin/cmudict (accessed Aug. 29, 2022).

[23] M. F. Porter, "An algorithm for suffix stripping," *Program*, 1980.

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

[4] https://pypi.org/project/seetm/
[5] https://pypi.org/project/siena/
[6] https://pypi.org/project/dime-xai/
[7] https://pypi.org/project/rasac/