INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Language Independent Data Augmentation for Text Classification and Sentiment Analysis

Rupesh Tailor, Harshit Raj, Chetan Anand, and Srividhya Vasanth

Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri, India

Abstract

Developing an effective text classification model in low-resource languages is challenging due to limited labeled data availability, which often incurs high costs for acquiring substantial labeled datasets. To address this challenge, a solution-oriented approach, Language-independent Data Augmentation (LiDA), leverages a multilingual language model to enhance semantic analysis and classification. LiDA operates at the sentence embedding level, allowing it to be agnostic to specific languages. Through evaluation across three languages using various dataset fractions, an observed enhanced performance was seen in both LSTM and BERT models. An ablation study was conducted to assess the impact of LiDA's components on overall performance, highlighting its effectiveness in improving text classification accuracy and semantic analysis..

Keywords: Text classification,Data augmentation, Multilingual language model, LSTM and BERT

Introduction

Text classification is a fundamental task in Natural Language Processing (NLP) with diverse applications across multiple domains, including spam detection, sentiment analysis, lysis, and topic detection. The recent advancements in deep learning have significantly enhanced the performance of text classification models. However, achieving high performance often requires a vast amount of labeled data, posing a challenge in low-resource languages like Indonesian. Acquiring such extensive labeled data is not only difficult but also expensive. To address this challenge, data augmentation techniques are employed to create synthetic data from existing labeled data. These techniques typically operate at the word or sentence level, aiming to transform original sentences into synthetic ones. At the word level, strategies include random word replacements with synonyms or the closest words in the embedding space, as well as sentence structure modifications like word deletion, insertion, or swapping. On the other hand, sentence-level techniques such as back-translation and generative methods are also commonly used. Back-translation generates sentences with the same meaning as the original through machine translation, while generative methods leverage text generation models to create synthetic sentences based on the probability of word occurrences in previous sequences.

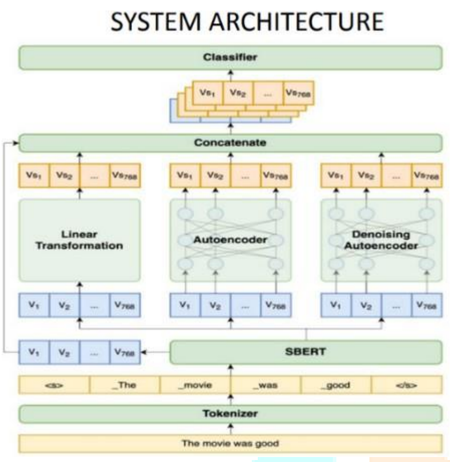


Figure 1: Visual representation and overview of system architecture.

Software Requirements

Software requirements encompass defining the necessary software resources and prerequisites that must be installed on a computer to ensure the optimal functioning of an application. These prerequisites are typically not bundled with the software installation package and need to be installed separately before the software can be installed. A platform, in computing, refers to a framework, whether in hardware or software, that enables software to operate. Common platforms include a computer's architecture, operating system, or programming languages along with their runtime libraries. The operating system is a primary consideration when outlining system requirements for software. Compatibility with different versions of the same operating system may vary, with some software not being compatible with older versions. APIs and drivers play a crucial role, especially for software reliant on specialized hardware devices, such as high-end display adapters, necessitating specific APIs or updated device drivers. For instance, DirectX is a set of APIs commonly used for multimedia tasks, particularly in game programming on Microsoft platforms. Web browsers are essential for web applications and software heavily reliant on internet technologies, often utilizing the system's default browser. For example, Microsoft Internet Explorer is frequently utilized for software running on Microsoft Windows, leveraging ActiveX controls despite their security vulnerabilities. As for specific software requirements, the Visual Studio Community Version, Node.js (Version 12.3.1), and Python IDLE (Python 3.7) are listed.

1. Hardware Requirements

Hardware requirements, also known as physical computer resources, are a common set of criteria defined by operating systems and software applications. These requirements are often accompanied by a hardware compatibility list (HCL), which catalogs tested and compatible hardware devices. Key aspects of hardware requirements include architecture, processing power, memory, secondary storage, display adapter, and peripherals. For instance, operating systems are designed for specific computer architectures, and software often defines processing power based on CPU model and clock speed. Memory requirements are determined by application demands and the needs of supporting software and files. Similarly, secondary storage requirements depend on software installation size and usage of temporary files. High-end display adapters may be necessary for software with advanced graphical demands. Some applications also require specific peripherals, such as CD-ROM drives or network devices. As for specific hardware requirements, the software mandates Windows as the operating system, an i5 processor or above, at least 4GB of RAM, and a minimum of 50GB of hard disk space.

2. System Analysis

2.1 Existing System

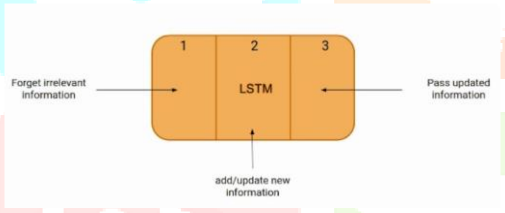
Many data augmentation approaches can be utilized to produce synthetic data for the text classification problem. The easiest method at the word level is to substitute words at random. In this method, the sentence’s random words are changed to their synonyms [1], [2], [3], their nearest word neighbors [4], or a predicted word from the language model [5], [6]. Another method involves altering the sentence structure by deleting words, adding words at random, or switching around terms already present in the text [3]. The most widely used method at the sentence level is back translation. This method uses machine translation models [7], [8], [9], and [10] to create a sentence that has the same meaning as the original text.

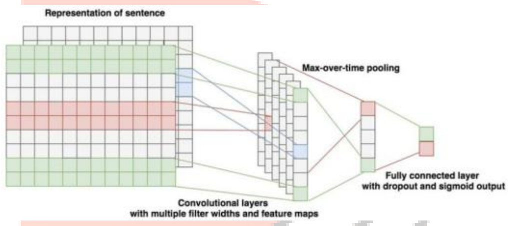
2.2 Disadvantages of Existing System

In low-resource languages, the scarcity of labeled data presents significant challenges for developing high-performance text classification models. The existing methods for enhancing dataset diversity, such as altering sentence structures by deleting, inserting, or swapping words, may not sufficiently address the need for robust semantic analysis and accurate classification in languages with limited linguistic resources.

2.3 Proposed System

In this paper LiDA is introduced, LiDA: A Language independent Data Augmentation technique for text classification. The approach used in this project works at the sentence embedding level, unlike previous methods that perform at the word and sentence level. The approach used in this project was inspired by insufficient data augmentation techniques in computer vision where synthetic images would be created by transforming the original image vector with some functions such as flipping, shifting, rotating, zooming, etc. Similarly, the approach used in this project transformed sentence embedding with some functions to create a new synthetic sentence embedding. LSTM, BERT, and CNN play crucial roles in this LiDA project. LSTM helps capture long-term dependencies in sequential data, such as text, making it suitable for tasks like text classification. BERT provides contextualized word embeddings, enhancing the understanding of language semantics. CNN extracts local features from text, aiding in tasks like sentiment analysis. Combining these techniques enables LiDA to effectively process multilingual text data and generate high-quality sentence embeddings, facilitating tasks like text classification and information retrieval with improved accuracy and efficiency.





Representation of sentence
Max-over-time pooling
Fully connected layer with dropout and sigmoid output
Convolutional layers with multiple filter widths and feature maps

(a) LSTM Gate (b) CNN Structure

Figure 2: LSTM Gate and Convulational Neural Network for Text Classification

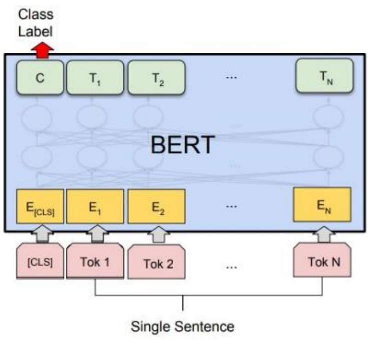


Figure 3: Bidirectional Encoder Representations from Transformers

BERT, or Bidirectional Encoder Representations from Transformers, is a pre-trained language model developed by Google. It utilizes a transformer architecture to generate contextualized word embeddings by considering the entire input text bi-directionally. BERT has revolutionized natural language processing tasks by capturing deeper linguistic contexts and nuances, leading to significant improvements in various NLP applications such as text classification, sentiment analysis, and question answering.

3. Functional requirements

- Data Collection
- Data Preprocessing
- Training And Testing
- Modeling
- Predicting

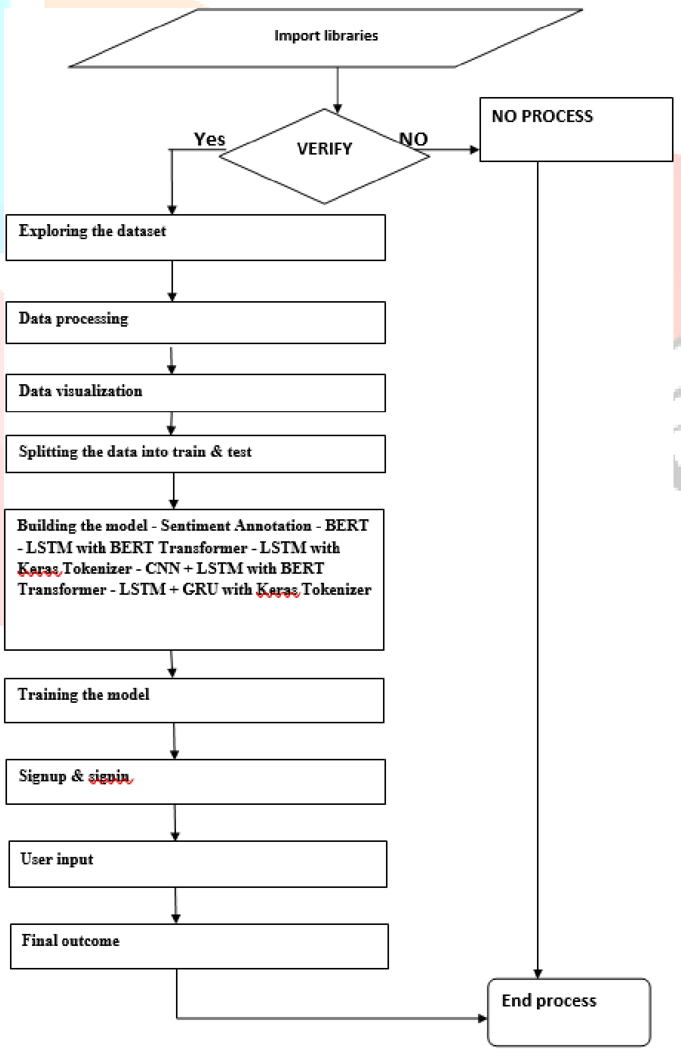

Import libraries
VERIFY
Yes
NO
NO PROCESS
Exploring the dataset
Data processing
Data visualization
Splitting the data into train & test
Building the model - Sentiment Annotation - BERT - LSTM with BERT Transformer - LSTM with Keras Tokenizer - CNN + LSTM with BERT Transformer - LSTM + GRU with Keras Tokenizer
Training the model
Signup & signin
User input
Final outcome
End process

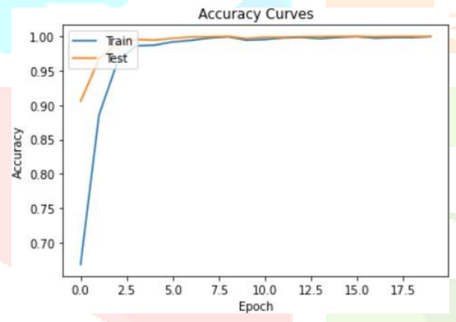
Figure 4: Working flowchart

4. Non-Functional Requirements

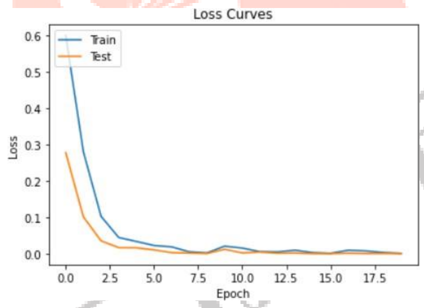
Non-functional requirements (NFRs) specify the quality attributes of a software system, evaluating aspects like responsiveness, usability, security, and portability. They are critical for assessing the success of the software. For instance, a non-functional requirement could address the speed at which a website loads. Failing to meet these requirements can lead to systems that do not meet user needs. NFRs enable the imposition of constraints or restrictions on system design across agile backlogs. For example, a requirement might specify that the site must load within 3 seconds when the number of simultaneous users exceeds 10,000. Describing non-functional requirements is as crucial as defining functional requirements. Common types of non-functional requirements include usability, serviceability, manageability, and recoverability.

Serviceability refers to the ease with which a system can be maintained, repaired, or serviced over its lifecycle. Manageability pertains to the ease of managing and controlling the system's resources and configurations. Recoverability involves the system's ability to recover from failures or disruptions and restore normal operations efficiently.

5. Performance metrics and Model Accuracy



(a) Accuracy curve plot



(b) Loss curve plot

Figure 5: LSTM+GRU Model Accuracy and Loss curve plot

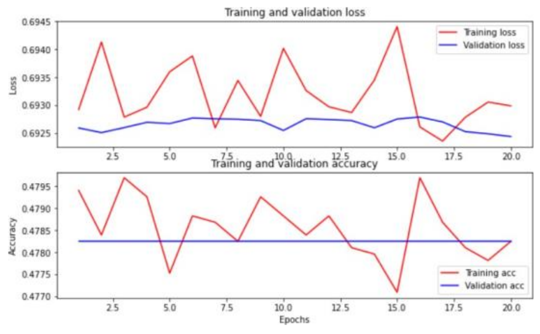
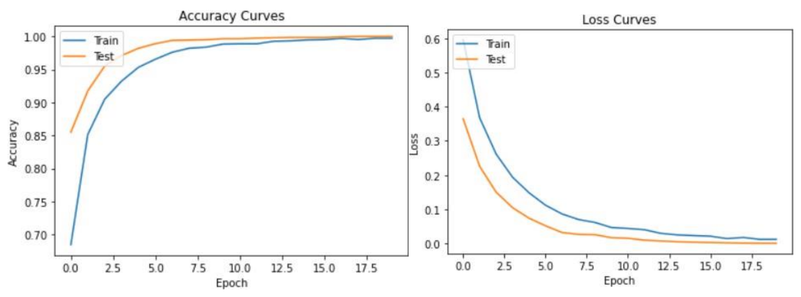


Figure 6: Training and Validation Loss for LSTM+BERT Model



(a) Accuracy curve plot (b) Loss curve plot

Figure 7: BERT Model Accuracy and Loss curve plot

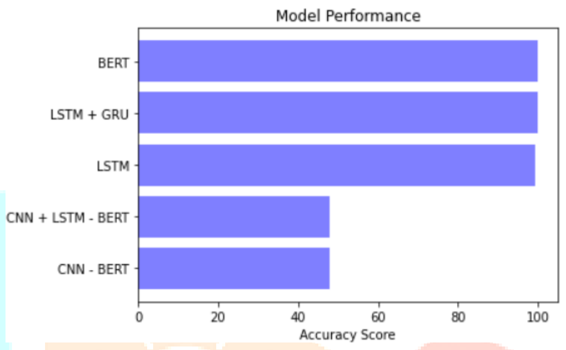


Figure 8: Performance comparison amongst all Models

- LSTM+GRU Model Accuracy: 99.86%
- LSTM Model Accuracy: 99.49%
- CNN-BERT Model Accuracy: 47.82%
- CNN+LSTM+BERT Model Accuracy: 47.82%
- BERT Model Accuracy: 100%

6. Conclusions

The conclusion of the LiDA project highlights the effectiveness of Language-independent Data Augmentation (LiDA) in addressing the challenge of creating high-performance text classification models in low-resource languages. Through LiDA, semantic analysis is being performed in the training dataset using a multilingual language model, transcending language barriers and enhancing the performance of both LSTM and BERT models. The evaluation is performed across multiple languages and demonstrated improved classification accuracy, showcasing LiDA's potential to mitigate the scarcity of labeled data in diverse linguistic contexts. Furthermore, an ablation study revealed insights into the significant components of this approach, emphasizing the importance of sentence-level embedding and language independence. Overall, LiDA presents a promising solution for multilingual text classification tasks, offering a scalable and effective approach to leverage existing data resources in linguistically diverse environments.

Acknowledgments

The authors express their gratitude to the Department of Computer Science at Amrita Vishwa Vidyapeetham, Amritapuri campus, for their unwavering support. Special thanks are extended to Mrs. Athira Joshi, whose guidance and assistance were invaluable throughout the duration of this project.

References

- Bhaskar, J., Sruthi, K., Nedungadi, P. (2014). Enhanced sentiment analysis of informal textual communication in social media by considering objective words and intensifiers. In International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014) (pp. 1-6). IEEE.
- HB, B. G., Kumar, M. A., Soman, K. P. (2016). Distributional semantic representation for text classification and information retrieval. In FIRE (Working Notes) (pp. 126-130).
- Nedungadi, P., Harikumar, H., Ramesh, M. (2014). A high performance hybrid algorithm for text classification. In The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014) (pp. 118- 123). IEEE.
- HB Barathi Ganesh, R Vinayakumar, M Anand Kumar, and KP Soman. "NLP CEN AMRITA@ SMM4H: Health Care Text Classification through Class Embeddings.
- Shriya Se, R Vinayakumar, M Anand Kumar, and KP Soman. "AMRITA-CEN@ SAIL2015: sentiment analysis in Indian languages." In Mining Intelligence and Knowledge Exploration: Third International Conference, MIKE 2015, Hyderabad, India, December 9-11, 2015, Proceedings 3, pp. 703-710. Springer, 2015.
- Radhika, K., Bindu, K. R., Parameswaran, L. (2018). A text classification model using convolutional neural network and recurrent neural network. International Journal of Pure and Applied Mathematics, 119(15), 1549-1554.
- Pandey, R., Singh, J. P. (2023). BERT-LSTM model for sarcasm detection in code-mixed social media post. Journal of Intelligent Information Systems, 60(1), 235-254. Springer.