MONOLINGUAL SENTENCE SIMILARITY MEASUREMENT USING SIAMESE NEURAL NETWORKS FOR SINHALA AND TAMIL LANGUAGES

Nilaxan Satkunanantham

179337B

M.Sc. in Computer Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

MONOLINGUAL SENTENCE SIMILARITY MEASUREMENT USING SIAMESE NEURAL NETWORKS FOR SINHALA AND TAMIL LANGUAGES

Nilaxan Satkunanantham

179337B

This dissertation submitted in partial fulfillment of the requirements for the Degree of MSc in Computer Science Specializing in Data Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2021

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other University or institute of higher learning 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 acknowledgment is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic, or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

UOM Verified Signature

Signature: ..... Date: 2021-05-31

Name: Nilaxan Satkunanantham

The supervisor/s should certify the thesis/dissertation with the following declaration.

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the MSc PG Diploma Project.

Signature of the supervisor: ..... Date: .....

Name: Dr. Surangika Ranathunga

ABSTRACT

Sentence similarity plays a key role in text-processing related research such as plagiarism checking and paraphrasing. So far, only conventional unsupervised sentence similarity techniques such as string-based, corpus-based, knowledge-based, and hybrid approaches have been used to measure sentence similarity for Tamil and Sinhala languages. In this research, we introduce a Deep Learning methodology to measure sentence similarity for these two languages, which makes use of Siamese Recurrent Neural Networks techniques together with a word-embedding model as the input representation. This approach achieved a 3.07% higher Pearson correlation coefficient for the Tamil dataset of 2500 sentence pairs and a 3.61% higher Pearson correlation coefficient for the Sinhala dataset of 5000 sentence pairs. Both these results outperform that of the conventional unsupervised sentence similarity techniques applied on the same datasets.

Keywords - Sentence-similarity, Sinhala, Tamil, Siamese neural network, LSTM, deep-learning, fastText, natural language processing

ACKNOWLEDGEMENT

I would like to express profound gratitude to my advisor, Dr. Surangika Ranathunga, for her invaluable support by providing relevant knowledge, materials, advice, supervision, and useful suggestions throughout this research work. Her expertise and continuous guidance enabled me to complete my work successfully.

I am grateful for the support and advice given by Dr. Charith Chitraranjan, by encouraging the continuation of this research. Further, I would like to thank all my colleagues for their help in finding relevant research material, sharing knowledge and experience, and for their encouragement.

I am as ever, especially indebted to my parents for their love and support throughout my life. I also wish to thank my loving wife, who supported me throughout my work. Also, I wish to express my gratitude to all my colleagues at Sysco LABS, for the support given to me to manage my MSc research work.

Finally, I would also like to expand my deepest gratitude to all those who have directly and indirectly guided us in completing this research.

TABLE OF CONTENTS

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENT iii

TABLE OF CONTENTS iv

LIST OF FIGURES vi

LIST OF TABLES vii

LIST OF ABBREVIATIONS viii

INTRODUCTION 1

1.1 Problem and Motivation 1

1.2 Problem Statement 2

1.3 Overall Objective 2

1.4 Research Contribution 3

1.5 Report Structure 3

LITERATURE REVIEW 4

2.1 String based similarity 4

2.1.1 Character based similarity measures 5

2.1.2 Term-based Similarity Measures 7

2.2 Corpus based similarity 11

2.3 Knowledge Based Similarity 13

2.3.1 WordNet 14

2.3.2.1 Tamil WordNet and Corpus 14

2.3.2.2 Sinhala WordNet and Corpus 14

2.3.2 Semantic Similarity Measures 15

2.3.3 Semantic Relatedness Measures 16

2.5 Hybrid Similarity Measurement Techniques 16

2.4 Deep Learning Based Techniques 19

2.4.1 Convolutional Neural Networks Based Approach 20

2.4.2 Recurrent Neural Networks Based Approach 22

2.4.3. Long Short-Term (LSTM) Memory Based Approach 22

2.4.3. Gated Recurrent Units Based Approach 24

2.6 Sentence Similarity Techniques used for Tamil and Sinhala 26

2.7 Vector Representation of Words 26

2.7.1 Word Co-occurrence Matrix 26

2.7.2 Word embeddings 27

2.7.2.1 Shallow Word Embeddings 28

2.7.2.2 Contextualized Word-Embeddings 28

RESEARCH METHODOLOGY 31
3.1 Dataset 31
3.1.1 Dataset for Sinhala 31
3.1.2 Dataset for Tamil 33
3.2 Visualization of embedding layer 35
3.3 Preprocessing 38
3.3.1 Sentence cleanup 38
3.3.2 Stop words removal 39
3.3.3 Spelling correction 41
3.3.3.1 Traditional spell correction 41
3.3.3.2 Contextual spell correction 42
3.3.4 Tokenization 43
3.4 Architecture and Implementation 43
3.4.1 Experiment with Siamese networks 44
3.4.2 Experiment with Cosine similarity 45
3.5 Embedding Layer 45
3.6 Hidden Layer 46
3.7 Sentence similarity measurement layer 47

SYSTEM EVALUATION AND RESULTS 48
4.1 Baseline experiment 48
4.2 Effects of preprocessing 48
4.3.1 Effects of cleanup 49
4.3.2 Effects of stop word removal 49
4.3.3 Effects of spell correction 50
4.3 Model evaluation 52
4.4 Results summary 55
4.5 Error Analysis 59

CONCLUSION 62
5.1 Future works 63

REFERENCES 64

APPENDIX-I 70

APPENDIX-II 71

LIST OF FIGURES

Figure 2.1:	String based similarity measurement techniques [5]	5
Figure 2.2:	Manhattan distance for similarity measures	8
Figure 2.3:	Hamming distance for similarity measures	8
Figure 2.4:	Euclidean distance for similarity measures	9
Figure 2.5:	Cosine similarity for similarity measures	9
Figure 2.6:	Jaccard similarity for similarity measures	10
Figure 2.7:	Corpus based similarity measurement techniques [5]	11
Figure 2.8:	Knowledge based similarity measurement techniques [5]	15
Figure 2.9:	Deep Learning based similarity measurement techniques	20
Figure 2.10:	A typical Siamese neural network architecture [32]	25
Figure 2.11:	CBoW, Skip-gram models architectures [64]	27
Figure 3.1:	Questionnaire for Tamil short sentence dataset collection	33
Figure 3.2:	Manual similarity score calculation method for Tamil dataset	34
Figure 3.3:	Visualization of Tamil dataset using t-SNE	36
Figure 3.4:	Visualization of Sinhala dataset using t-SNE	37
Figure 3.5:	Preprocessing steps for short sentence dataset	38
Figure 3.6:	Simplified architecture of MaLSTM model [8]	44
Figure 3.7:	System for short sentence similarity measurement [8]	45
Figure 4.1:	Model training and validation performance for Tamil dataset	53
Figure 4.2:	Model training and validation performance for Sinhala dataset	54
Figure 4.3:	Model evaluation performance for Sinhala	58
Figure 4.4:	Model evaluation performance for Tamil	58

LIST OF TABLES

Table 2.1:	Sentence similarity for STS-2017 dataset	24
Table 3.1:	Gold Relatedness Score	32
Table 3.2:	Manually Annotated Score for Sinhala	32
Table 3.4:	Tamil - Manual similarity score calculation strategy	34
Table 3.5:	Manually annotated score for Tamil	35
Table 3.6:	List of Sinhala stop-words	39
Table 3.7:	List of Tamil stop-words	40
Table 4.1:	Baseline experiment results	48
Table 4.2:	Sentence cleanup experiment results	49
Table 4.3:	Stop words removal experiment results	50
Table 4.4:	Spelling errors or misspelling words in Tamil dataset	51
Table 4.5:	Spelling errors or misspelling words in Sinhala dataset	51
Table 4.6:	Sentence spelling correction experiment results	52
Table 4.7:	MaLSTM model parameters	53
Table 4.8:	Tamil - Short similarity score comparison	54
Table 4.9:	Sinhala - Short sentence similarity score comparison	56
Table 4.10:	Performance comparison using Pearson correlation coefficient	57
Table 4.11:	Loanwords in Tamil dataset	59
Table 4.12:	Loanwords in Sinhala dataset	59

LIST OF ABBREVIATIONS

Abbreviation	Description
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
BoW	Bag of Words
CBoW	Continuous Bag-of-Words
POS	Part of Speech
IR	Information Retrieval
Q&A	Question and Answer
VSM	Vector Space Model
MaLSTM	Manhattan LSTM
LCS	Longest Common SubString
STS	Semantic Text Similarity
SVD	Singular Value Decomposition
HAL	Hyperspace Analogue to Language
GLSA	Generalized Latent Semantic Analysis
ESA	Explicit Semantic Analysis
CL-ESA	Cross-Language Explicit Semantic Analysis
PMI-IR	Pointwise Mutual Information - Information Retrieval
SCO-PMI	Second-order Co-Occurrence Pointwise Mutual Information
NGD	Normalized Google Distance
DISCO	DIStributionally similar words using CO-occurrences
Bi-LSTM	Bidirectional LSTM
GRU	Gated Recurring Units
Bi-GRU	Bidirectional GRU
STS	Semantic Text Similarity
RDF	Resource Description Framework
ERCNN	Enhanced Recurrent Convolutional Neural Networks
CARNN	Context Aligned RNN
SA-BiLSTM	Self-Attention based BiLSTM
NNLM	Feedforward Neural Net Language Model
CoVe	Contextual Word Vectors
BERT	Bidirectional Encoder Representations from Transformers
ELMo	Embeddings from Language Model
ULMFiT	Universal Language Model Fine-tuning for Text Classification
CVT	Cross-View Training
t-SNE	t-Distributed Stochastic Neighbouring Embedding

1. INTRODUCTION

Measuring the similarity of sentences is the basis for many text-processing related research and applications [1]. The sentence similarity must determine how close two sentences in a pair are in surface proximity (lexical similarity), as well as in meaning (semantic similarity). An efficient sentence similarity technique should be able to ascertain the semantic similarity of sentences, even if they differ in the surface (lexical) form.

Conventionally, sentence similarity is measured using techniques such as string based, corpus based, knowledge based and some simpler similarity measurement techniques that make use of features such as word-order and word-length [2, 3]. To determine the similarity of words, knowledge-based similarity measurement approaches are utilized to acquire information from semantic networks (e.g., WordNet) or lexical resources. To measure semantic similarity between words, corpus-based similarity measurement techniques use information from really massive corpora. In addition, analyzing the co-occurrence of words in a large corpus helps to accurately assess the similarity between these words. String based comparison focuses on character composition and string arrangements. String based method uses two types of similarity measures: character based similarity measures and term based similarity measures. Other similarity measures consider word order and word length information alone with other techniques [4].

1.1 Problem and Motivation

Tamil and Sinhala are the official languages of Sri Lanka, which are currently used by a population of over 21 million in Sri Lanka and over 93 million all over the world. The amount of information available in the text format of both languages is rapidly increasing. Thus, natural language processing (NLP) applications are progressively becoming more important and play a crucial role in facilitating the retrieval and analysis of these data.

Numerous techniques are available for sentence similarity measurement. However, quite a few conventional techniques experimented with Tamil and Sinhala languages. Yet the performance of such techniques is considerably low for these languages.

String-based similarity techniques are not able to capture the semantic similarity of sentences. Conversely, Corpus-based similarity techniques are domain-dependent, and knowledge-based methods rely on handcrafted dictionaries. Due to the limitations in these techniques, many preceding research works have combined two or more similarity methods to form a hybrid similarity measurement technique to obtain a higher accuracy [2, 3, 5, 6]. These methods calculated semantic similarity between sentences, and the output sequence was used to evaluate sentence similarity.

Kadupitiya et al. [2] and Anutharsha et al. [3] have reported sentence similarity measurement using a hybrid technique for short sentences in Sinhala and Tamil languages (respectively). This research uses a hybrid approach that combines semantic similarity (similarity information of knowledge base and corpus statistics) techniques [5]. Due to the incompleteness of semantic networks or lexical resources and large corpora, short sentence similarity measurement models for Tamil and Sinhala languages have yet to reach the stage of proliferation demonstrated by other languages. These hybrid approaches perform better compared to other conventional similarity techniques, yet their performance is considerably lower than that of the more recent sentence similarity measurement techniques.

More specifically, all the recent sentence similarity techniques are based on Deep Learning. These include ConvNet [7], skip-thoughts [8], Tree-LSTM [9], BiLSTM [10], Self-Attention-based BiLSTM [11], and MaLSTM [12]. These techniques have shown very promising results for languages such as English and Chinese.

1.2 Problem Statement

In this research, we focus on how to use Deep Learning-based techniques to measure the sentence similarity for low-resourced language short sentences, focusing on Tamil and Sinhala languages.

1.3 Overall Objective

The objective of this research is to identify an effective approach that uses Deep Learning techniques to improve the accuracy and performance of the short sentence similarity measurement techniques for low resourced languages, specifically focusing on Tamil and Sinhala languages.

1.4 Research Contribution

My research has contributed to the natural language text-processing domain and presented a Deep Leaning technique for measuring sentence similarity for low-resourced languages, focusing on the Tamil and Sinhala languages. It particularly focused on the short sentence similarity problem.

1.5 Report Structure

The remainder of this report is structured in the following manner. The fourth chapter examines the literature on the subject. This contains the details of the general sentence similarity measurement methods and previous work in the areas of short sentence similarity measurement for Tamil and Sinhala languages. The fifth chapter presents the research methodology. The sixth chapter completes the report and reveals future enhancements.

2. LITERATURE REVIEW

The resemblance between two sentences or expressions of the same context is defined as sentence similarity or text similarity measurement. In the literature, there are many approaches utilized to measure the similarity between sentences.

The common approaches used for sentence similarity range from simple word-vector dot products to pairwise classification, and more recently, deep neural networks [13]. The following sections give brief explanations for the most used methods.

2.1 String based similarity

String based similarity measurement performs on a string's sequences and character structure that can measure the similarity using approximate string matching or comparison between a pair of text strings [14]. String-based methods are simple to calculate, but they do not capture semantic similarity. String matching can be used to determine the similarity of words in a variety of ways.

String based similarity techniques are mainly broken down into the following categories (as shown in Figure 2.1):

- Character based similarity measures techniques
- Terms based similarity measures techniques

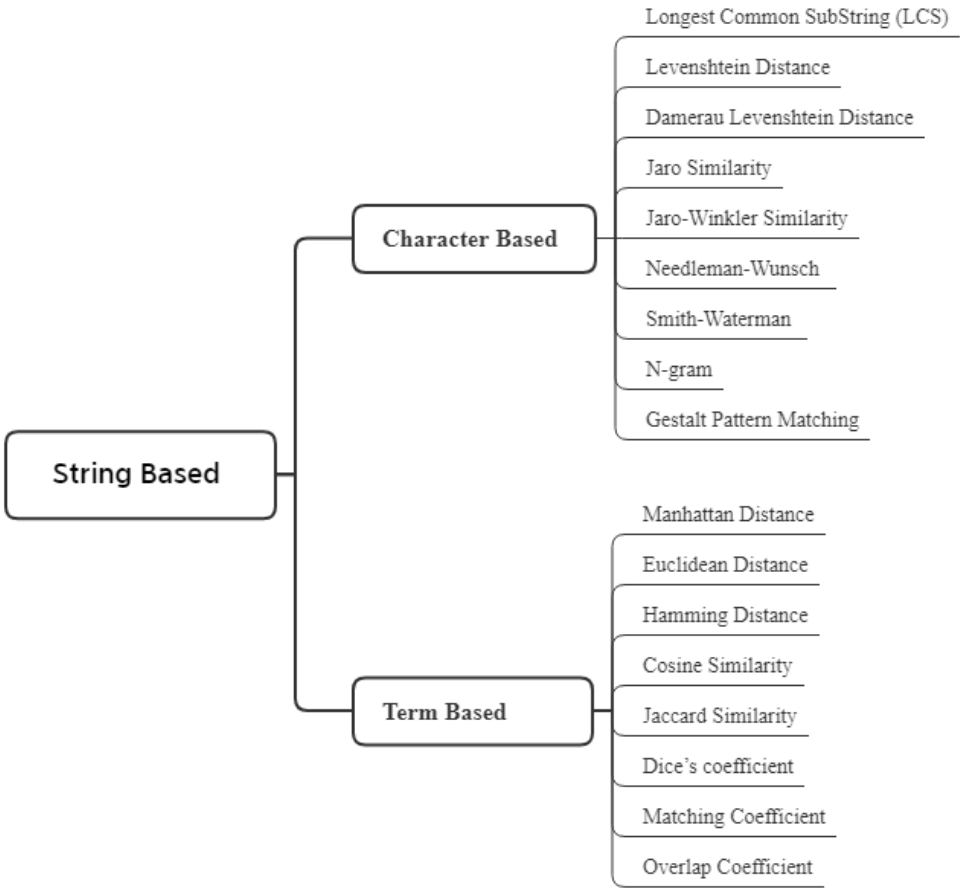


Figure 2.1: String based similarity measurement techniques [14]

2.1.1 Character based similarity measures

This similarity measurement technique is used to calculate the similarity between characters in the text, and the output of this process is to convey the similarity between text. The following subsections give brief explanations for the most popular methods.

Longest Common Substring (LCS) [15] is used to discover the lengthiest common substring between two strings. In other words, measure the similarity between two dependent strings on the length of the adjoining characters sequence that exists in the two strings.

For example, if s_1 = **'substring'** and s_2 = **'string'**, then the longest common substring of s_1 and s_2 is **'string'**

Levenshtein distance (LD) [16] is a string-based similarity measure and operates between two strings. In other words, the LD between two words is determined by the total number of single character edits (substitutions, insertions, or deletions) needed to turn one word into the other word.

Let two strings ***a, b,*** and length of $a = |a|$ and length of $b = |b|$;

The equation for calculating Levenshtein distance $lev_{a,b}(|a|,|b|)$ is given below:

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (2.1)$$

Damerau Levenshtein distance (DL) [17] is described as the smallest number of acts necessary to change one string into another. A single character may be added, omitted, or substituted, or two adjacent characters can be transposed.

To express the DL distance between strings a and b; function $d_{a,b}(i,j)$. The equation can be written as:

$$d_{a,b}(i,j) = \min \begin{cases} 0 & \text{if } i = j = 0 \\ d_{a,b}(i-1,j)+1 & \text{if } i > 0 \\ d_{a,b}(i,j-1)+1 & \text{if } j > 0 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \text{if } i,j > 0 \\ d_{a,b}(i-2,j-2)+1 & \text{if } i,j > 1 \text{ and } a[i] = b[j-1] \text{ and } a[i-1] = b[j] \end{cases} \quad (2.2)$$

For example, Distance between ***ZX*** and ***XYZ***. The Damerau-Levenshtein distance **LD *(ZX, XYZ)*** = ***2*** because ***ZX → XZ → XYZ***

Jaro Similarity [14] is a similarity metric that evaluates the similarity of two strings based on the amount and order of common characters. If the Jaro distance score is high, then it indicates two strings are more similar.

Jaro-Winkler similarity [14] is an enlargement of Jaro distance and also commonly used for examining the similarity between two strings. It is a similarity measure, rather than a distance measure; the Jaro-Winkler similarity value simply indicates that both strings are more similar.

Needleman-Wunsch Similarity [14] is based on dynamic programming and used in biological string sequence comparison. It finds the best arrangement in the entire string sequence of the two string sequences by performing a global alignment. This

method works well when two string sequences are similar in length and have a high level of overall similarity.

Smith-Waterman Similarity [14] is also based on dynamic programming. It finds the best arrangement over the protected domain of two string sequences by performing a local alignment. It is beneficial for heterogeneous sequences of string, which are assumed to comprise similarity regions or similar sequence of string motif regions in their more string sequence perspective.

N-gram Similarity [14]: This method evaluates the *n*-grams from every word or character in a pair of strings. The similarity is calculated by dividing the number of related n-grams by the number of n-grams with the highest number of n-grams.

Gestalt Pattern Matching [18]: It is a string matching-based similarity approach for determining how similar two strings are. This method divides the cumulative number of characters in both strings by twice the number of identical characters. Recursively, the matching characters are defined as the LCS plus the number of matching characters in both sides of the LCS's non-matching regions.

2.1.2 Term-based Similarity Measures

Term-based similarity measurement is based on the semantic relationships between terms to express the similarity between text. The following subsections give brief explanations for the most popular Term based similarity methods.

Manhattan distance [19] calculates the distance between two data points (shown in Figure 2.2) as the sum of the differences between its corresponding components.

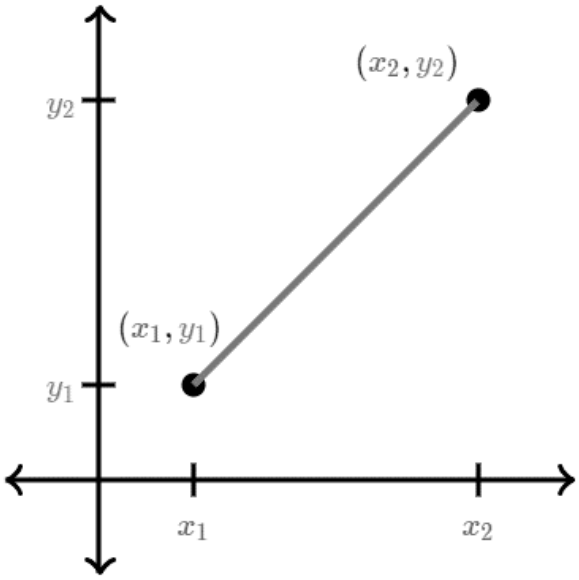


Figure 2.2: Manhattan distance for similarity measures.

Let A (x_1, y_1) and B (x_2, y_2) are the data points and the following equation is used for calculating Manhattan distance between A and B:

$$\text{Manhattan distance} = (|x_2 - x_1| + |y_2 - y_1|) \qquad (2.3)$$

Hamming distance [20] is used to compare two binary formats of strings (shown in Figure 2.3). When comparing two binary equal length strings, the Hamming distance is the number of bit positions of two different bits. When data is communicated through a computer network, it is used for error detection or error correction. It is also used to compare equal-length data in information coding theory.

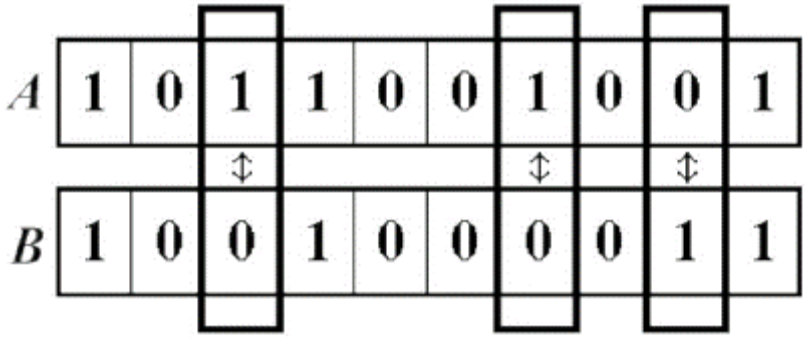


Figure 2.3: Hamming distance for similarity measures

Euclidean distance [21] is the straight line distance between two data points in vector space, as shown in Figure 2.4.

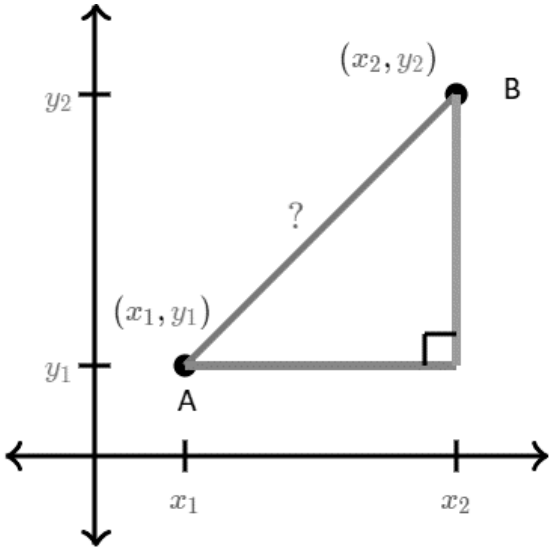


Figure 2.4: Euclidean distance for similarity measures

Let A (x_1, y_1) and B (x_2, y_2) are the data points and the equation for calculating Euclidean distance between A and B is given as below:

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2.4)$$

Cosine similarity [14] is the cosine angle between two data points in a multi - dimensional vectors space, as shown in Figure 2.5.

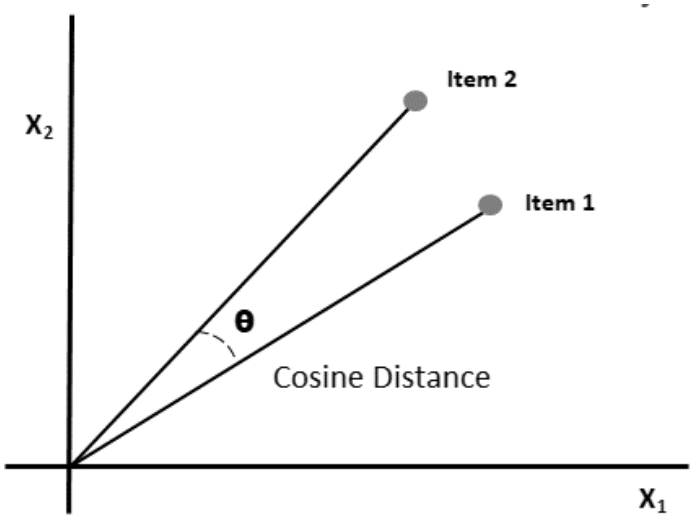


Figure 2.5: Cosine similarity for similarity measures

The equation for calculating Cosine distance between *Item1* and *Item2* is given as below:

$$\text{Cosine distance} = \frac{A \,.\, B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2.5)$$

Jaccard Similarity /coefficient [22] defines the level of similarity between two sets of data. It is computed by multiplying the intersection's size by the union's size of two sets, as shown in Figure 2.6. The equation (2.6) shows methods for calculating the Jaccard similarity.

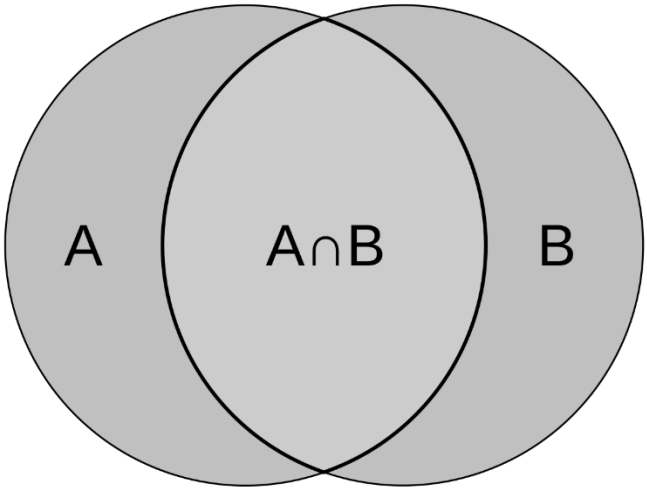


Figure 2.6: Jaccard similarity for similarity measures

$$\text{Jaccard Similarity} = \frac{|A \cap B|}{|A \cup B|} \quad (2.6)$$

Sørensen Dice coefficient [23] is a statistical method to compare two samples to see how close they are. The following equation is used to calculate the Dice's coefficient.

$$\text{Dice's coefficient} = \frac{2|A \cap B|}{|A| + |B|} \quad (2.7)$$

Simple Matching Coefficient (Rand similarity coefficient) [14] is a very simple strategy used to assess the similarity and variety of sample sets. The equation for calculating SMC is give as below:

$$\text{SMC} = \frac{number\ of\ matching\ attributes}{number\ of\ attributes} \quad (2.8)$$

Overlap Coefficient [14] is defined as the size of the intersection divided by the smaller of the two sets' sizes, as indicated in the following equation. To put it another way, it is a measurement of the overlap between two sets of data.

The size of the intersection divided by the smaller of the two sets' sizes is known as the Overlap Coefficient, as shown by the following equation. In other words, it is the measurement of the overlap between two sets.

$$\text{Overlap coefficient} = \frac{|A \cap B|}{Min(|A|,|B|)} \qquad (2.9)$$

2.2 Corpus based similarity

Corpus based similarity [14] calculates the similarity between words corresponding to information obtained from larger Corpora. Figure 2.7 shows the branch of corpus based similarity measure methods.

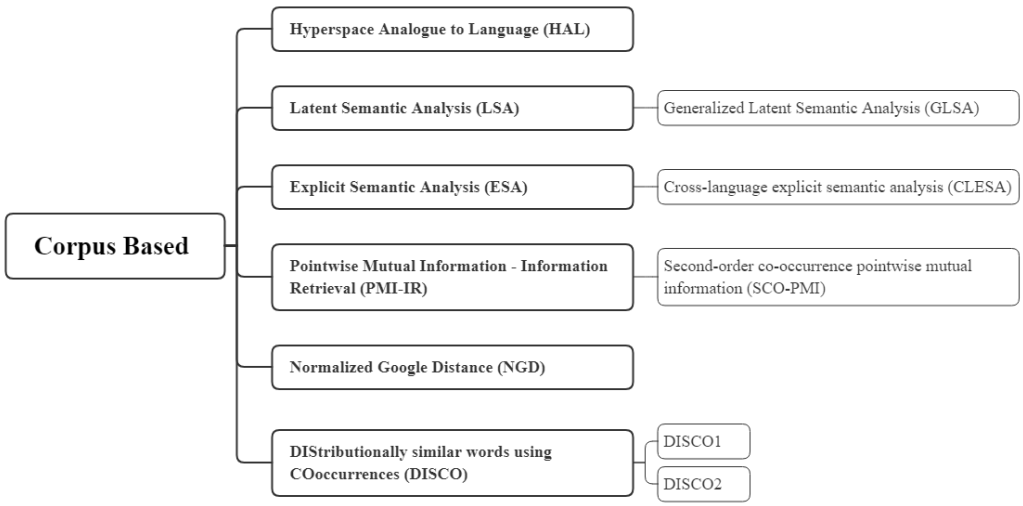


Figure 2.7: Corpus based similarity measurement techniques [14]

HAL (Hyperspace Analogue to Language) [24] uses word co-occurrences to construct a semantic space. Each matrix unit is a word-by-word matrix, indicating the strength of the relation between the row and column words. The operator of the algorithm will then determine whether or not to hold the low-entropy column in the matrix. The emphasis words are inserted at the start of the ten words window while processing the text, which documents the adjacent words that are considered as co-occurring. The co-occurrence is weighted in inverse proportion to the distance of the same target term, and the matrix value is cumulative. Neighboring words are weighted higher since they are assumed to represent the semantics of the target term. HAL additionally maintains word order by treating co-occurrences differently depending on whether they appear after or before the target word.

LSA (Latent Semantic Analysis) [25] is a widely used similarity based on corpus statistics information. The Latent Semantic Analysis method believes similar-meaning words will appear in similar-messages. A large section of text is used to create a matrix containing the word count of each paragraph (rows represent unique terms, and columns represent each paragraph). The Singular Value Decomposition (SVD) mathematical procedure is used to minimize the number of columns while maintaining the similarity structure among rows. The cosine distance measure is used to compare terms.

GLSA (Generalized Latent Semantic Analysis) [13] This method is mainly used for calculating text vectors and term vectors that are semantically inspired. It builds on the work of LSA [25] by relying on term vectors rather than the double representation of document terms. It includes a dimensionality reduction process and a test of semantic correlation between words. Any similarity calculation on the term space can be combined with appropriate dimensionality diminution method in the GLSA approach. The weights in the linear mixture of term vectors are calculated using the standard term text matrix in the final process.

ESA (Explicit Semantic Analysis) [13] is a metric for calculating the textual similarity between pairs of texts. Words (or sentences) are described as high-dimensional vectors in the Wikipedia technique, with each vector element showing the TF-IDF weight between the term and one Wikipedia article. The cosine similarity between the respective vectors reflects the semantic relatedness of two words.

CL-ESA (Cross Language - Explicit Semantic Analysis) [13] is a multilingual generalization of ESA. CL-ESA uses a document-associated multilingual reference list such as Wikipedia to represent a text as a language independent definition vector. The cosine resemblance between the respective vector representations is used to evaluate the relatedness of two documents in various languages.

PMI-IR (Pointwise Mutual Information - Information Retrieval) [26] This method is mainly used for calculating the likelihood of two terms being identical. It calculates the odds using the “AltaVista's Advanced Search” query syntax. The higher the PMI-IR similarity ranking, the more likely two words appear near each other on a web page.

SCO-PMI (Second order Co-Occurrence - Pointwise Mutual Information) [13] This method measures the semantic similarity that ranks the important neighbor word lists of target words pairs from a huge corpus based on point-by-point mutual knowledge. SOC-PMI has the advantage of calculating the resemblance between two terms that exist infrequently since they occur at the same time including the same adjacent words.

NGD (Normalized Google Distance) [13] is a different way for obtaining semantic similarity statistics based on the amount of Google hits for a set of keywords. In Google distance units, words with comparable or equal meanings in natural language tend to be "near," and words with distinct meanings tend to be "far".

DISCO (Distributionally similar words using co-occurrences) [14]: Words with similar meanings appear in similar ways, according to distributional similarities. To determine distributional similarity, large text sets are statistically evaluated. DISCO is a tool for measuring distributional similarity between terms by counting co-occurrences using a basic background window of size 3 words. When DISCO calculates the precise similarity between two keywords, it does so by obtaining their word vectors from the indexed data. DISCO returns the second order word vector for the provided word when the most distributionally near word is requested. The two most common resemblance measures are DISCO1 and DISCO2.

- **DISCO1**: Based on their collocation sets, this function calculates the first-order similarity of two input terms.

- **DISCO2**: Based on their sets of distributionally related terms, this function estimates the second-order similarity between two input word.

2.3 Knowledge Based Similarity

These approaches are mostly used to assess similarity using semantic network information [27]. The WordNet [28] is a widely used semantic network in knowledge-based techniques for determining similarity. Figure 2.8 shows the Knowledge-based similarity measure techniques.

2.3.1 WordNet

WordNet [28] is a lexical database of semantic relationships between words in English. WordNet links words into semantic relations including synonyms, hyponyms, and meronyms. Cognitive synonyms (synsets) are collections of nouns, verbs, adjectives, and adverbs that, each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet can be considered as a combination and extension of a dictionary and thesaurus.

2.3.2.1 Tamil WordNet and Corpus

WordNet for Tamil [29] is a lexical database for Tamil language and it is linked with 2000 synsets, but 2851 unique words are found for 1969 synsets. Tamil corpus [30] contains 60000 random sentences. Tamil WordNet is still under construction and not comprehensive. Therefore, it cannot produce accurate outputs in NLP tasks.

2.3.2.2 Sinhala WordNet and Corpus

WordNet for Sinhala [31, 32] is a lexical database for Sinhala language. There were two separate research carried out for developing WordNet for the Sinhala language. Welgama. et al. [31]'s Sinhala Wordnet consists of 1000 of the most common senses of contemporary Sinhala usage and Wijesiri. et al. [32]'s Sinhala WordNet consists of 2000 of synsets. Sinhala corpus [33] was constructed using 10 million words collected from newspaper articles. There are roughly 135,000 distinct words in the corpus. However, WordNet for Sinhala is not comprehensive and is still under construction. Therefore, we cannot produce accurate outputs in NLP tasks.

Semantic similarity and semantic relatedness measures are the two main categories of Knowledge-based similarity measures, as shown in Figure 2.8.

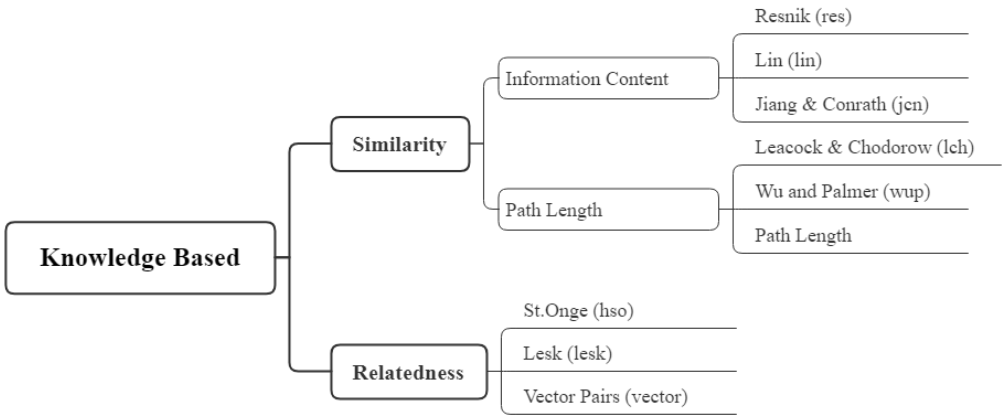


Figure 2.8: Knowledge based similarity measurement techniques [14]

2.3.2 Semantic Similarity Measures

Semantic similarity involves a broader scope and focusing the interactions between ideas that incorporate additional similarity relations such as *is-a-part-of, is-a-kind-of, is-the-opposite-of* and *is-a-specific-example-of.*

Information Content

- **Resnik (res)**: The information content (IC) of the Least Popular Subsumer is equal to the associated value in res [34]. The value's upper bound is usually very high, though it varies based on the scale of the corpus used to evaluate information quality values.
- **Lin (lin)**: The Least Common Subsumer's information content is scaled by this number in the lin [35] calculation.
- **Jiang & Conrath (jcn)**: The jcn [36] takes the difference between this amount and the Least Popular consumer's information content.

Path Length

- **Leacock & Chodorow (lch)**: The lch [37] measure generates a score that indicates how closely two word senses are connected. The smallest distance between the senses and the taxonomic depth at which the senses occur.

- **Wu & Palmer (wup)**: Based on the depth of their Least Common Subsumer and the depth of the two senses in the taxonomy, the wup [38] metric produces a score showing how closely two-word senses are related.

- **Path Length (path)**: The route metric in the is-a (hypernym/hyponym) taxonomy yields a score that reflects how closely two word senses are related based on the shortest path between them.

2.3.3 Semantic Relatedness Measures

Conversely, Semantic relatedness measure is a further widespread concept of relatedness, not precisely attached to the shape or structure of the concept.

- **St. Onge (hso)**: The hso [14] measure identifies lexical chains that connect the two-word senses. Extra-strong, strong, and medium-strong interactions are the three types of relationships that are considered.

- **Lesk (lesk)**: The lesk [39] metric works by looking for gloss overlaps between the two synsets. The sum of the squares of the overlap lengths determines the relatedness ranking.

- **Vector Pairs (vector)**: The vector pair [14] technique creates a co-occurrence matrix for each word in a corpus's WordNet notes, then uses an average of these co-occurrence vectors to characterize each gloss or definition.

2.5 Hybrid Similarity Measurement Techniques

In Hybrid approaches, two or more sentence similarity measurement techniques are combined to form an approach of word semantic similarity to determine the sentence similarity that helps to get better accuracy and performance. Word-to-word similarity gives better results when combining similarity measurement techniques.

Mihalcea et al. [27] presented a combination approach that measures the semantic similarity of text by using information derived from the similarity of constituent words. They use two corpus-based metrics, namely LSA [25] and PMI-IR [26], and six knowledge-based Metrics [34, 35, 36, 37, 38, 39] to analyze the semantic similarity of words and combined the results to indicate how to use these measures to derive a text similarity measure. They evaluate their methods based on paraphrase recognition tasks.

It uses eight different similarity methods to calculate the similarity of words. This is the main drawback, and it is not computationally efficient.

Islam et al. [40] presented a corpus-based technique (SOC-PMI) for determining the semantic similarity of sentences. and a normalized and modified version of the LCS [41]. This approach uses a mixture of semantic and syntactic knowledge to evaluate the similarities of two sentences. It first measured string-similarity and semantic-word-similarity, then, applied syntactic information using an optional generic word-order similarity function. Finally, normalization is used to combine string similarity, semantic similarity, and general word-order similarity to extract sentence similarity. This method obtains a good accuracy (Pearson's r) for 30 sentence pairs in the dataset, and the result is better than the Li et al. [5] approach.

Atish et al. [42] proposed another sentence similarity measurement method based on corpus analysis and background knowledge. This method treats the sentences as a word sequence and processes entire words in the sentence independently corresponding to its semantic and syntactic structure. Before conducting statistical analysis on the corpus, they first disambiguated the corpus words. Therefore, every word in the corpus is appended to the WordNet [28] synset. In addition, to calculate the similarity between words, the shortest path distance in WordNet is used. In addition, the word depth in the WordNet hierarchy is used to indicate the specificity level of the word.

Lintean et al. [43] introduced a semantic similarities measuring method for calculating similarity between short texts using greedy matching and word semantics. The method is based on the composition principle where, the overall meaning of a sentence can be captured by condensing the meaning of each section of the sentence. Based on this principle, the semantic similarity measure from word to word can quantify the semantic similarity at the sentence level. First, a set of mutually exclusive similar word pairs is constructed between two input texts. Then, these word pairs are used to calculate the overall similarity score between the texts through a weighted sum. Finally, the weighted length of the original text is used to normalize the calculated sum. A greedy strategy is used to find the closest word match. In addition, they exclude words with similarity values below a predefined threshold.

Ferreira et al. [44] introduced a sentence similarity procedure based on a similarity matrix that integrates various pre-processing methods to sentence representation and a metric to determine the degree of sentence similarity by encompassing three layers: lexical, syntactic, and semantic. The lexical layer receives a sentence as input and outputs a list of sentence tokens that describe it. This layer has two sections: Lexical representation and Lexical similarity. In Lexical representation, Lexical analysis, stop word removal, and Lemmatization steps are performed. The similarity of words is calculated using six different metrics (Path [14], LD [16], Res [34], Lin [35], lch [37], and wup [42]).

The second layer is based on syntactic analysis. This layer receives the token sequence generated in the lexical layer and visualizes them into a graph representation using Resource Description Framework (RDF) triples. This layer has two sections: Syntactic representation and Syntactic similarity. In Syntactic representation, Syntactic analysis and Graph creation transformation steps are followed. The relation of the syntactic layer determined by matching the vertices of the RDF triples is used to quantify syntactic similarity between sentences.

The final layer suggested is based on semantic study and explains the RDF graph with sense identification and entity roles. This layer has two sections, semantic representation, and semantic similarity. Semantic representation takes a series of token groups retrieved in the lexical layer as input and uses Semantic Role Annotation (SRA) to specify the functions of each entity and determine their "meaning" in the sentence. SRA is used by the semantic layer to conduct Sense recognition and Role annotation. The syntactic similarity between sentences is determined using the relation of the syntactic layer calculated by matching the vertices of the RDF triples in syntactic similarity.

Finally, the overall calculation of sentence similarity of the Ferreira et al. [44] approach is calculated by combining the syntactic, lexical, and semantic measures used here which is given by (2.10).

$$\text{similarity}(S_1,S_2) = \frac{(lex_n \times lex_s) + (syn_n \times syn_s) + (sem_n \times sem_s)}{lex_n + syn_n + sem_n} \qquad (2.10)$$

Li et al. [5] proposed a hybrid method for short sentence similarity calculation by

integrating semantic similarity measures (corpus and knowledge based similarity measures) and word order-based similarity measures. This procedure considers the implicit semantic and word order information in the sentences. The semantic similarity of two sentences is calculated using information from WordNet [28] and Brown corpus statistics. The semantic similarity is first determined using the corpus and knowledge base. The impact of word order on sentence context is then taken into account for this method. The number of distinct words and word pairs in different sequences was measured using the derived word sequence similarity. A sentence is interpreted as a sequence of words in this method, each of which provides valuable information about the context. The words and their combined structure make a sentence to convey a particular meaning. When comparing all the previous possible hybrid-based unsupervised techniques for English short sentence similarity, Li's et al. [5] method has given the most accurate results. Li et al. [5] and Mihalcea et al. [27] hybrid approaches do not consider the string similarity, which plays a crucial role in some situations.

2.4 Deep Learning Based Techniques

Semantic similarity approaches derived benefit from the recent advancements in Deep Learning algorithms to improve accuracy and performance. Deep Learning techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) [45], Bidirectional Long Short-Term Memory (Bi-LSTM) [10], Recursive Tree-LSTM [9], Gated Recurring Units (GRU) [47] and Bidirectional Gated Recurring Units (BiGRU) most frequently being used in recent research works. This section briefly discusses certain of the most popular Deep Learning-based approaches to measuring syntactic and semantic similarity between sentences. Figure 2.9 shows the most widely used Deep Learning techniques for similarity measurement.

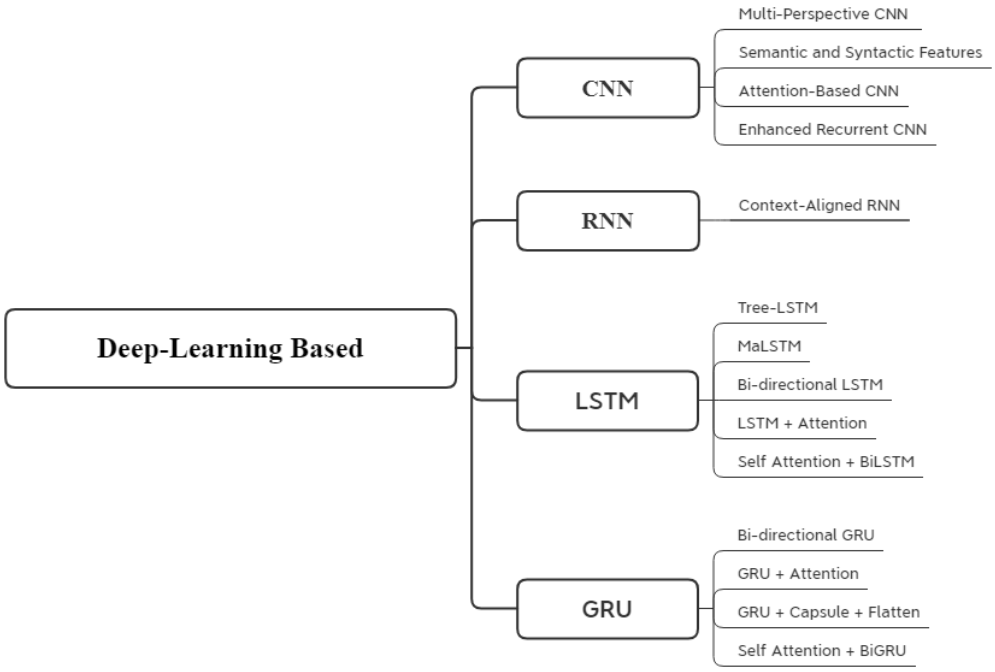


Figure 2.9: Deep Learning based similarity measurement techniques

2.4.1 Convolutional Neural Networks Based Approach

He et al. [7] proposed a Multi-Perspective Convolutional Neural Networks (CNN) called the ConvNet model. This ConvNet model has two major components: sentence modeling, and similarity measurement. Sentence model is used to convert sentences into a representation for similarity measurement. A CNN architecture with multiple convolution and pooling layers is used in the sentence model to capture different granularities of information in the inputs. The similarity measurement component compares local regions of the sentence representations generated by the sentence model. A Siamese structure [48] with two sub-networks that process each sentence in parallel is used in the similarity measurement component. The sub-networks share all their weights, and are joined by the similarity measurement layer, subsequently followed by a fully connected layer for similarity score. This model performs without using external resources such as WordNet or parsers, and also does not use sparse features generated using co-occurrence of words in a large corpus.

Xie et al. [49] propose an ERCNN (Enhanced Recurrent Convolutional Neural Network) model that learns sentence similarity with less computational sophistication by combining more fine-grained features and the interactive results of key points in

two sentences. Input encoding, local similarity modeling, and overall similarity modeling are the three components of this ERCNN model. First, bidirectional GRU (BiGRU) was used to encode the two input sentences, as well as column-wise average and maximum pooling, and this model supports stacking an arbitrary number of convolution-pooling blocks to extract progressively abstract functionality. Then, in local similarity modeling, the basic component for evaluating overall similarity is modeling local sub sentential similarity between two inputs. Between two input sentences, soft attention is used to connect the related pieces. Finally, a concatenation layer was created to perform similarity modeling on two sentences to determine their overall similarity. There are two forms of concatenation in this layer. For the interactive sentence representation, the first concatenation contains the difference and the element-wise product. The production of the first concatenation is further concatenated in the second concatenation.

Yin et al. [50] proposed an ABCNN (Attention Based Convolutional Neural Network) method for sentence pair modeling. It is classified into 3 structural designs based on the BCNN: ABCNN-1, ABCNN-2, and ABCNN-3, each of which implements an attention framework for modeling sentence pairs. Each column represents a cluster, a term at the lowest level and a phrase at the highest level, and it computes attention weights directly on the input representation with the goal of enhancing the convolutional features. ABCNN-2 reweights convolution performance by computing focus weights on it. Convolution is influenced implicitly by attention in the ABCNN-1, while pooling is influenced specifically by attention in the ABCNN-2. ABCNN-3 stacks the ABCNN-1 and ABCNN-2, combining their strengths by allowing the attention system to run on both the convolution and pooling sections of a convolution pooling block, as well as the input and more abstract output granularity.

Zhang et al. [51] suggested a convolutional neural networks (CNN) approach for calculating sentence similarity based on semantic and syntactic characteristics. There are two components of this model. The syntax model uses the syntactic features of a sentence to compute, while the semantic model uses the semantic features of the words in a sentence to compute. The model can be computed in a number of ways, including convolution and pooling. The following procedure explains how the similarity score between two sentences was calculated. First, given a sentence, two matrices are created: a syntax model input matrix and a semantic model input matrix.

The syntax model input matrix records some syntax features, while the semantic model input matrix records some semantic features. They follow the most efficient method of constructing the matrices by playing with various arrangements of describing the syntactic and semantic features of the sentences in the matrices. Second, these two matrices are fed into two neural networks, one for the sentence model and the other for the semantic model. The neural networks of the two models are convolutional from different angles. The outputs of the two models are merged to form a vector, which serves as the sentence's representation. Third, given the representation vectors of two sentences, a CNN layer computes the similarity score of these representations.

2.4.2 Recurrent Neural Networks Based Approach

In recent times, Recurrent Neural Networks (RNN) have shown good performance in sentence similarity modeling. But most RNNs focus on modeling the hidden states based on the current sentence, while the context information from the other sentence is not well investigated during the hidden state generation.

Chen et al. [52] proposed a Context-Aligned RNN (CARNN) model, which incorporates the contextual information of the aligned words in a sentence pair for the inner hidden state generation. Specifically, first perform word alignment detection to identify the aligned words in the two sentences. Then, present a context alignment gating mechanism and embed it into their model to automatically absorb the aligned words' context for the hidden state update.

2.4.3. Long Short-Term (LSTM) Memory Based Approach

LSTM [45] networks are a form of RNN that can learn long-term dependencies. LSTM is a sequence modeling technique that uses several inside layers to produce long-term sequences. The LSTM constructs a memory cell and employs gates to determine how much material must be forgotten or pass through the time steps. O_t output gate, C_t cell memory block (what information will be fed to the next neuron is determined by the current state.), I_t input gate, and F_t forget gates are the four components. The LSTM layer receives feedback in the form of real-valued vectors from the input I_t Between the gates, the secret state representations H_t are modified sequentially.

The upgrade steps depend solely on the C_t cell memory block. These four components determine which data is used in the model and which data is left out for final prediction.

Tai et al. [9] presented a Tree-LSTM model that generalizes the order-sensitive chain-structure of standard LSTM to tree-structured network topologies. The Tree-LSTM constructs its hidden state at a given tree node from the corresponding word as well as the hidden states of all child nodes after each sentence is transformed into a parse tree using a separately qualified parser. The expectation is that the parse tree-structured network, by representing syntactic properties of a phrase, would be able to spread required knowledge more effectively than a sequentially restricted architecture. Tree-output LSTM can be used to perform sentence similarity tasks.

Recently, Siamese recurrent neural networks are mostly used for sentence similarity calculation. Mueller et al. [12] proposed the MaLSTM model that uses a Siamese neural network. It has two identical LSTM networks, one per each sentence to be compared. An LSTM model projects a zero-padded word-embedding of an input sentence into a fixed-sized 50-dimensional vector. Each LSTM returns a vector representation of the final hidden state of each sentence. Manhattan distance function is used to measure sentence similarity.

Bidirectional RNNs [53] combine both potential and past meaning by running the input backwards into a different RNN. At each time point, the combined model's output is simply the concatenation of the forward and backward networks' outputs. Bi-directional LSTM (BiLSTM) [46] tends to understand the context more cleverly than Unidirectional LSTM and BiLSTM models have recently shown good results on standard NLP tasks.

The Enhanced BiLSTM Inference Model (EBIM) [54] encodes of sentence using a bidirectional LSTM over word2vec embeddings, produces a soft alignment matrix for the two sentences using a parameter-less attention process, and then uses another LSTM to infer over each timestep and its alignment. The prediction was completed by two totally linked layers.

Bi-LSTM layers are used to model forward and backward information for the fusion of left and right backgrounds. Then, using a self-awareness layer, context words that

are strongly associated with feature meanings are given higher weights. The polarity class of the focused output is detected using the SoftMax layer.

2.4.3. Gated Recurrent Units Based Approach

The Gated Recurrent Unit (GRU) [47] is an LSTM variant with similar performance and less parameters, thus it is faster to train. GRUs have been shown to exhibit better performance on small and medium datasets.

The Skip-Thought model [8] abstracts word vectors into sentence vectors, where the word2vec skip-gram model [55] is used to derive word vectors. An encoder-decoder architecture with GRU activations is used as the deep learning model. The encoder maps words to a sentence vector, and the decoder generates the surrounding of the sentences. The skip-thoughts model is used to encode the sentence and then tries to reconstruct the next and previous sentence. Skip-thought vectors are obtained for the sentence pair. A logistic regression classifier is trained for sentence similarity score prediction using features derived from absolute differences and component-wise products between skip-thought vectors for each sentence pair.

Ichida et al. [56] proposed a siamese GRU model based on Mueller et al. [12] model for calculating semantic similarity in a pair of sentences. The GRU architecture is used to overcome the limited size of the available labeled data set, because the parameters of the GRU unit are less than the LSTM unit. This model also uses Manhattan distance.

Table 2.1: Sentence similarity for STS-2017 dataset [57].

Approach	**Pearson correlation (τ)**
BiLSTM	0.8540
Capsule - Flatten – GRU	0.8545
MaLSTM	0.8651
Attention – GRU	0.8725
Attention – LSTM	0.8743
BiGRU	0.8750
GRU	0.8792

Ranasinghe et al. [20] compared the performance (shown in Table 2.1) of the various Deep Learning architectures such as ConvNet [6], Skip-thought [8], Dependency Tree-LSTM [9], and MaLSTM [12] based on an English dataset and reported that the Mueller et al approach achieved better results for English [12][57] than the other techniques discussed above.

Siamese Network

Siamese networks are deep neural networks that process two or more input vectors simultaneously and merge the output vector after sub-identical neural network computation [48]. Siamese networks tend to be excellent at similarity activities, and they have been used for things like sentence semantic similarity, identifying forged signatures, and many other things. Weights must be spread across all inputs to reduce training parameters and the risk of overfitting. Since the Siamese network shares weights on both ends, it is simpler to practice, as shown in Figure 2.10.

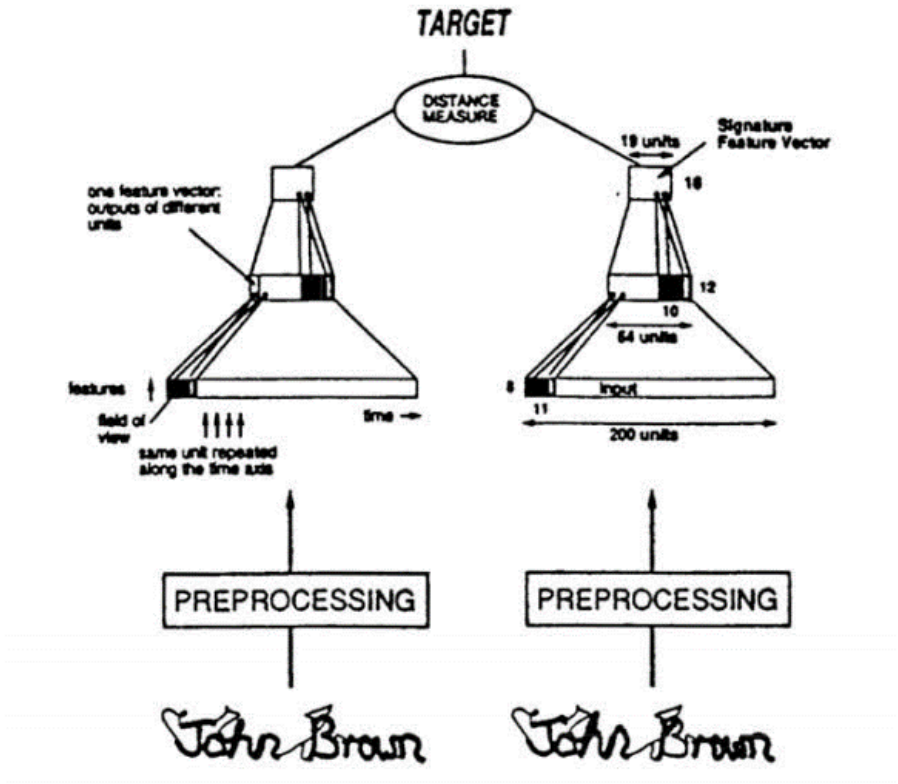


Figure 2.10: A typical siamese neural network architecture [48]

2.6 Sentence Similarity Techniques used for Tamil and Sinhala

Kadupitiya et al. [2] and Anutharsha et al. [3] followed a hybrid approach similar to Li et al. [5] for the Sinhala and Tamil languages, respectively. Their hybrid approach merged semantic similarity measures (Corpus and Knowledge based similarity) with word order based similarity measures. Due to the incompleteness of natural language processing (NLP) resources such as Sinhala WordNet [31, 32], a large corpus for the Sinhala language [33], Kadupitiya et al. [2] modified Li et al. [5]'s knowledge-based similarity measures procedure to use the Sinhala lexical resources also modified Li et al. [5]'s corpus-based similarity computation procedure to contemplate statistical information from glossaries of the Sinhala words. Anutharsha et al. [3] used Tamil WordNet [29] as the lexical database for knowledge-based similarity calculation and a Tamil corpus [30] for corpus-based similarity calculation for Tamil language. The overall similarity of a sentence is calculated by combining the semantic and word-order similarity scores.

2.7 Vector Representation of Words

The word vector is a number vector representing the meaning of the word. The word vector represents a significant increase in the ability to analyze the relationship between words. Word vectors are generated through two common ways:

- Counts of word/context co-occurrences (Co-occurrence Matrix)
- Predictions of context given word (Word embeddings)

2.7.1 Word Co-occurrence Matrix

Word co-occurrence matrix illustrates the way words collectively exist that in turn encapsulates the relationships between words. Word co-occurrence matrix is calculated by counting how two or more words collectively appear in a provided corpus.

GloVe

The GloVe is based on a global word co-occurrence matrix extracted from the corpus. It assesses similarity based on the idea that words that are similar comes together. The co-occurrence matrix of occurrence values is generated using a single pass through the underlying huge corpus. GloVe [59] was trained on five different corpora, the majority of which were Wikipedia dumps. When generating vectors, words are

employed inside a certain context window since words far apart have less value to the context word in question. GloVe model's loss function decreases the least-square difference between the context window's co-occurrence values and global values [13]. To differentiate words based on context, GloVe vectors are expanded to generate contextualized word vectors [61].

2.7.2 Word embeddings

Word-embeddings are word vector representations that keep the basic linguistic association between the words intact. These vectors are measured using a number of techniques, including neural-networks [58], matrix of co-occurrence words [59], and representations in terms of the word's context. Word embeddings are created using the unsupervised algorithms Skip-gram and CBoW. The architectures of CBoW and Skip-gram models are shown in Figure 2.11. The word-embeddings [55] represent a way of contrasting words semantically [60].

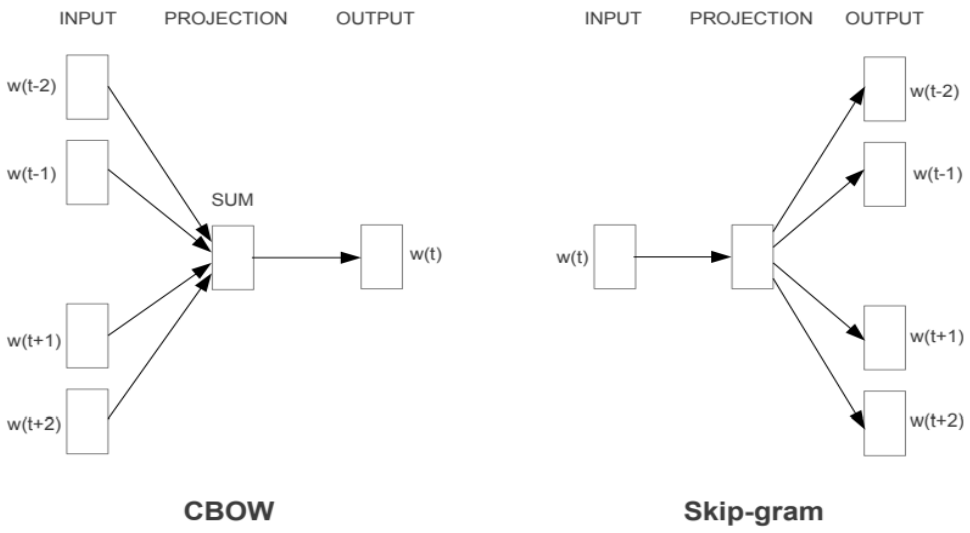


Figure 2.11: CBoW, Skip-gram models architectures [55]

CBoW

The Feedforward Neural Net Language Model (NNLM) architecture is similar to the Continuous Bag-of-Words model [55], except that the nonlinear hidden layer is omitted, and the projection layer is shared with all words; hence, all words are predicted into the same position. The CBoW architecture uses context to predict the current word.

Continuous Skip-gram

Instead of predicting the current term based on meaning, the Continuous Skip-gram model [55] design attempts to optimize classification of a word based on another word in the same statement. The skip-gram is used to predict the context word for a given target word.

2.7.2.1 Shallow Word Embeddings

Shallow word embedding is a function that maps each word type to a single vector. The following subsections give brief explanations of the most widely used shallow word embedding models.

Word2Vec

Word2vec [58] uses a basic Corpus to build distributed vector representations of words. It was made with the help of the Google News Dataset, which contains over 3 million vector representations of words and phrases. The Continuous Bag of Words (CBoW) and the Skip-gram models are two independent word2vec models. The network's architecture is straightforward, with an input layer, hidden layer, and an output layer. As input, the network receives a massive text corpus, and as output, the model produces a vector representation of texts. Word2vec models are efficient at representing word vectors while maintaining contextual similarity. The semantic similarity could be predicted well using word vector calculations [62].

FastText

FastText is a word vectors embedding model based on the skip-gram approach, in which each word is represented as a collection of character n-grams. In languages like Tamil and Sinhala, FastText [63] generates word embeddings as the average of its character embeddings, taking into account the word's "morphological shape" and providing useful information. In addition, non-dictionary words are classified as word vectors based on their characters or subunits.

2.7.2.2 Contextualized Word-Embeddings

Contextualized word embedding models build a vector for each word conditioned on its context. Contextualized word embeddings can effectively predict more? words

than standard word embeddings. It improved the performance on an enormous variety of text processing tasks in natural language processing [64].

Embeddings from Language Model (ELMo)

ELMo is a deep contextualized word representation introduced by Peters et al. [65]. It learns word syntax, semantics information, and linguistic context information using BiLSTM models. After pre-training, an internal state of vectors can be transferred to downstream natural language processing tasks. ELMo learns contextualized word representation through unsupervised pre-training of a language model.

Generative Pre-trained Transformer (GPT)

GPT [66] is another type of contextualized word embedding and is a massive transformer-based model for language with 1.5 billion parameters that was trained on a dataset of 8 million web pages that took up 40GB. GPT-2 and GPT-3 are qualified to anticipate the next word provided all the previous words in a text. Because of the dataset's diversity, this basic target contains spontaneously occurring demonstrations of a wide range of activities from various domains.

Bidirectional Encoder Representations Transformers (BERT)

BERT [67] is a machine learning algorithm based on the transformer model and mainly for NLP pre-training developed by Google. It is based on a bidirectional transformer architecture rather than a unidirectional transformer used in Open AI GPT [66]. In contrast to ELMo, which uses a shallow concatenation layer [65], BERT employs a deep concatenation layer. As a result, BERT is considered a very powerful embedding architecture.

- **BERT$_{base}$** – It has 12 layers (Encoders / Transformer blocks), 12 bidirectional attention heads with 110 million parameters.
- **BERT$_{large}$** – It has 24 layers (Encoders / Transformer blocks), 16 bidirectional attention heads with 340 million parameters.

BERT is based on stacked layers of encoders. The difference between BERT-base and BERT-large is in the number of encoder layers. The BERT-based model has 12 encoder layers stacked on top of each other, while the BERT-large model has 24 encoder layers stacked on top of each other.

Contextual Word Vectors (CoVe)

In an attentional sequence-to-sequence (seq2seq) machine translation model, CoVe [61] is a type of word embedding learned by an encoder. CoVe word representations are functions of the entire input sentence, unlike conventional word embedding.

Cross-View Training (CVT)

In ELMo, two autonomous models go through unsupervised pre-training and task-specific learning in two distinct training levels. Cross-View Training [68] puts it all together into a single semi-supervised learning procedure. Both supervised learning with classified data, unsupervised learning with unlabeled data on auxiliary tasks boost the representation of a BiLSTM [46] encoder in CVT.

Universal Language Model Fine Tuning (ULMFiT)

Howard and Ruder et. al. [69] provided a novel method for fine-tuning of neural models for inductive transfer learning. BERT is deeply bidirectional due to its novel shielding language modeling technology. On the other hand, ELMo uses a concatenation of right-to-left and left-to-right LSTMs, while ULMFiT uses a unidirectional LSTM.

3. RESEARCH METHODOLOGY

In this research, we present the first study on the use of Deep Learning techniques for sentence similarity measurement for Sinhala and Tamil languages. We employ an approach that was adopted from the MaLSTM model by Mueller et al. [12]. The similarity between pairs of short sentences is measured using the Siamese recurrent neural networks. The MaLSTM uses the Manhattan distance function to calculate the similarity of the final hidden state of both LSTM networks in a Siamese recurrent neural network.

Our system is based on the MaLSTM model introduced by Mueller et al. [12]. This model consists of a Siamese Deep Neural Network, which has two sub-networks: $LSTM_{left}$ and $LSTM_{right}$ with shared weights. Independently, they process one of the sentences in each pair that are passed as vector representations and return a hidden state that encodes the semantic meaning of the sentence. The fastText [63] word-embedding model is used to map words to vectors. Vector distance between the two hidden states is calculated using the Manhattan similarity function to produce a similarity score.

3.1 Dataset

We carried out this research using the same dataset from previous research by Kadupitiya et al. [2] for Sinhala and due to the unavailability of the Tamil dataset used by Anutharsha et al. [3], we created a new dataset for Tamil, using the same steps Kadupitiya et al. [2] used to create the Sinhala dataset. The following sections briefly describe dataset preparation strategies used for both Tamil and Sinhala languages.

3.1.1 Dataset for Sinhala

Kadupitiya et al. [2] followed a similar dataset preparation approach as Marelli et al. [70] and created a short sentence dataset for the Sinhala language. 500 images were selected from the SICK dataset [70] and They asked to write one short Sinhala sentence by describing each image & 5 individuals participated in this process. They were able to collect 2500 short Sinhala sentences. Then, these sentences were randomly paired, and a 5000 sentence pair dataset was formed for the Sinhala language. Finally, they hired 3 human judges to manually mark the similarity score

between sentence pairs in the dataset. The sentence similarity score was assigned based on the gold relatedness score [71] (shown in Table 3.1).

Table 3.2 illustrates examples of sentence pairs with semantic similarity scores (manually annotated score). We used this same dataset for our research.

Table 3.1: Gold Relatedness Score [71]

Definition	**Similarity score**
Both sentences are entirely similar and share the same meaning.	5
Both sentences are mostly similar, but certain trivial information differs.	4
Both sentences are approximately similar, but certain trivial information is missing.	3
Both sentences are not similar but refer some information to the same topic.	2
Both sentences are entirely different.	1

Table 3.2: Manually Annotated Score for Sinhala [2]

Sentences	**Similarity measure score**
A: කබරයෝ දෙදෙනෙක් වතුර කඩිත්තක අරගලයක නිරත වෙති (two monitors fight in a stream of water) B: කිඹුලන් දෙදෙකු වැවක සිටියි (two crocodiles in a lake)	0.67
A: රතු පැහැති මෝටර් රථයක් වේගයෙන් ධාවනය වේ (a red car drives fast) B: මිනිසෙක් යතුරු පැදියක් ධාවනය කරයි (A man drives a motor bike)	0.43
A: මෝටර් රථ යාන්ත්‍රිකයෙක් කාර් එකක උවසුන වෙතට බටයකින් වතුර පහරක් එල්ල කරයි (A motor mechanic aims water to the bonnet using a hose) B: මිනිසෙක් ඔසවා ඇති මෝටර් රථයක් සෝදමින් සිටියි (A man washes a car that is lifted up)	0.97

3.1.2 Dataset for Tamil

Due to the unavailability of the Tamil dataset used by Anutharsha et al. [3], we followed a similar approach as Kadupitiya et al. [2] and created the dataset for the Tamil short sentence dataset. We randomly extracted 100 images from the SICK dataset [70] that included domains such as sports, entertainment, nature, wildlife, and music. We used Google Forms to prepare a set of online questionnaires (shown in Figure 3.1) to collect Tamil short sentences. We used these 100 images to prepare 10 questionnaires, each containing 10 images. These questionnaires were shared with five students, and they were asked to write a short sentence about the images. 500 sentences were collected from students for the Tamil short sentence dataset.

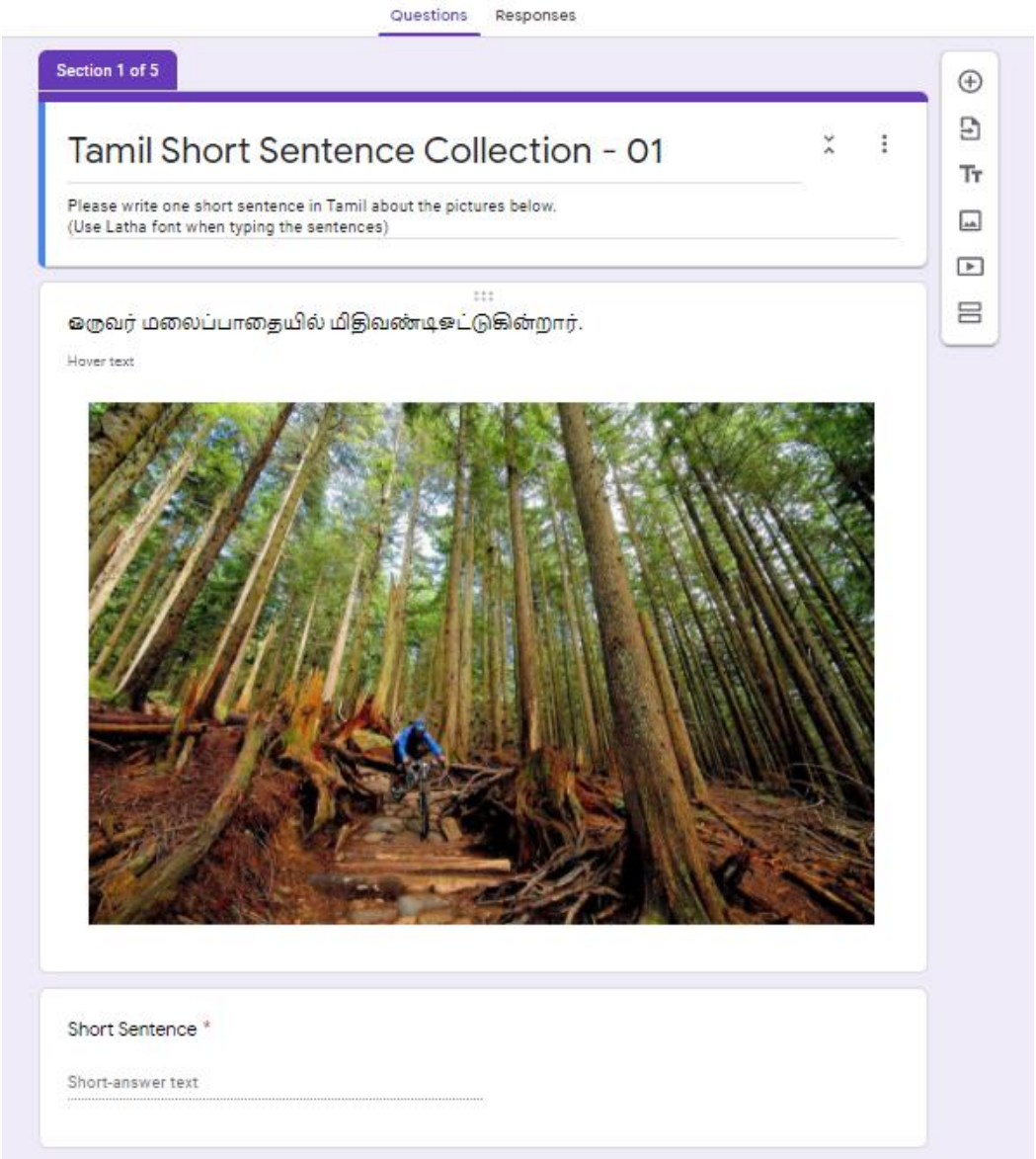
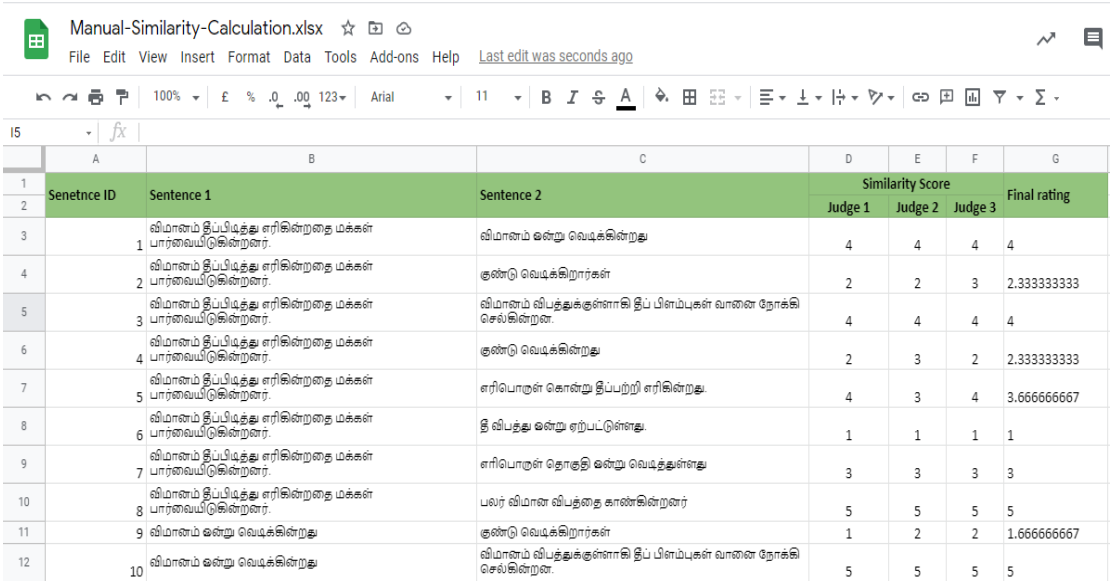


Figure 3.1: Questionnaire for Tamil short sentence dataset collection

We randomly paired the collected short sentences and got 2500 sentence pairs. Subsequently, we shared this sentence pair dataset with three human judges via a Google Sheet (shown in Figure 3.2). Table 3.4 shows the method used to get the final similarity score between pairs of short sentences in the given dataset. The sentence similarity score was assigned based on the gold relatedness score [71] (shown in Table 3.1).


Manual-Similarity-Calculation.xlsx

File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

Senetnce ID	Sentence 1	Sentence 2	Similarity Score			Final rating
			Judge 1	Judge 2	Judge 3	
1	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	விமானம் ஒன்று வெடிக்கின்றது	4	4	4	4
2	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	குண்டு வெடிக்கிறார்கள்	2	2	3	2.333333333
3	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	விமானம் விபத்துக்குள்ளாகி தீப் பிளம்புகள் வானை நோக்கி செல்கின்றன.	4	4	4	4
4	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	குண்டு வெடிக்கின்றது	2	3	2	2.333333333
5	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	எரிபொருள் கொன்று தீப்பற்றி எரிகின்றது.	4	3	4	3.666666667
6	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	தீ விபத்து ஒன்று ஏற்பட்டுள்ளது.	1	1	1	1
7	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	எரிபொருள் தொகுதி ஒன்று வெடித்துள்ளது	3	3	3	3
8	விமானம் தீப்பிடித்து எரிகின்றதை மக்கள் பார்வையிடுகின்றனர்.	பலர் விமான விபத்தை காண்கின்றனர்	5	5	5	5
9	விமானம் ஒன்று வெடிக்கின்றது	குண்டு வெடிக்கிறார்கள்	1	2	2	1.666666667
10	விமானம் ஒன்று வெடிக்கின்றது	விமானம் விபத்துக்குள்ளாகி தீப் பிளம்புகள் வானை நோக்கி செல்கின்றன.	5	5	5	5

Figure 3.2: Manual similarity score calculation method for Tamil dataset.

Table 3.4: Tamil - Manual similarity score calculation strategy

Sentence ID	Sentence 1	Sentence 2	Similarity Score			Final Score
			Judge 1	Judge 2	Judge 3	
1	தாவி குதித்து சாஹசம் புரிகின்றார்கள்	ஒருவர் குத்து கரணம் அடிக்கின்றார்	4	4	4	4
2	தாவி குதித்து சாஹசம் புரிகின்றார்கள்	இருவர் குதித்து சாகசம் புரிகின்றனர்	4	5	3	4
3	தாவி குதித்து சாஹசம் புரிகின்றார்கள்	ஒருவர் தலைகீழாகப் பாயும் போது மற்றையவர் அவருக்கு மேலாகப் பாய்கின்றார்.	5	5	3	4.33

4	தாவி குதித்து சாஹசம் புரிகின்றார்கள்	இருவர் கம்பிக்கு மேலாகப் பாய்கின்றனர்	5	3	3	3.66
5	தாவி குதித்து சாஹசம் புரிகின்றார்கள்	தலைகீழாக ஒருவர் விழுகின்றார்.	5	5	4	4.66

The manual similarity score is calculated by getting the average similarity scores collected from 3 human judges as given by Equation (3.1). Table 3.5 shows the example of manual similarity scores for Tamil short sentence pairs.

$$\text{Manual Similarity Score} = \frac{J1\ score + J2\ score + J3\ Score}{3} \qquad (3.1)$$

Table 3.5: Manually Annotated Score for Tamil [3]

Sentences	**Similarity score**
ஒருவர் வீதியில் ஓடுகின்றார். (A person is running on the road) ஒருவர் ஓடிக்கொண்டிருக்கின்றார். (A person is running)	4.67
ஒரு நபர் துப்பாக்கியால் சுடுகின்றார். (A person is shooting with a gun) துப்பாக்கியில் இருந்து குண்டு செல்கின்றது. (A bullet is coming out of a gun)	2.00
போட்டியின்போது விபத்து நடந்துள்ளது. (Accident has happened in sports) விளையாட்டு வீரர்கள் விளையாட்டில் மும்முரமாக ஈடுபட்டுள்ளனர். (Players play very well)	1.00

3.2 Visualization of embedding layer

We used the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm used for doing nonlinearly dimensionality reduction and visualizing our multidimensional data model. This algorithm helps to visualize the high-dimensional data point into a two or three-dimensional plan [72].

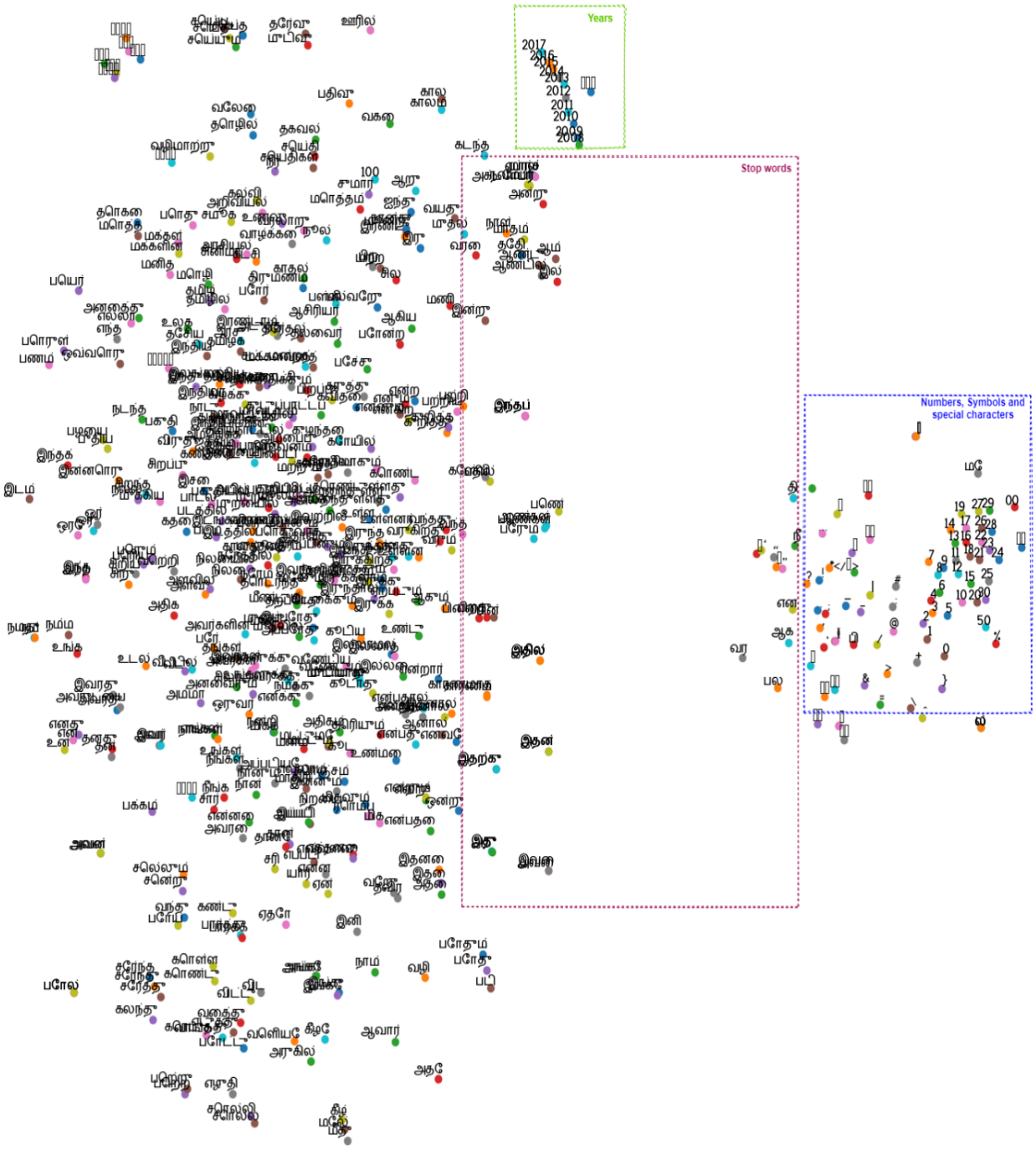


Figure 3.3: Visualization of Tamil dataset using t-SNE [72]

We wanted to know how the words are arranged in the vector space to understand our dataset. We used t-SNE for exploration and visualizing our dataset. We randomly selected 500 words from each 300-dimensional embedding layer and visualized them on a two-dimensional plate using t-SNE. The word vector model visualization for Tamil and Sinhala languages are shown in Figures 3.3 and 3.4 (respectively).

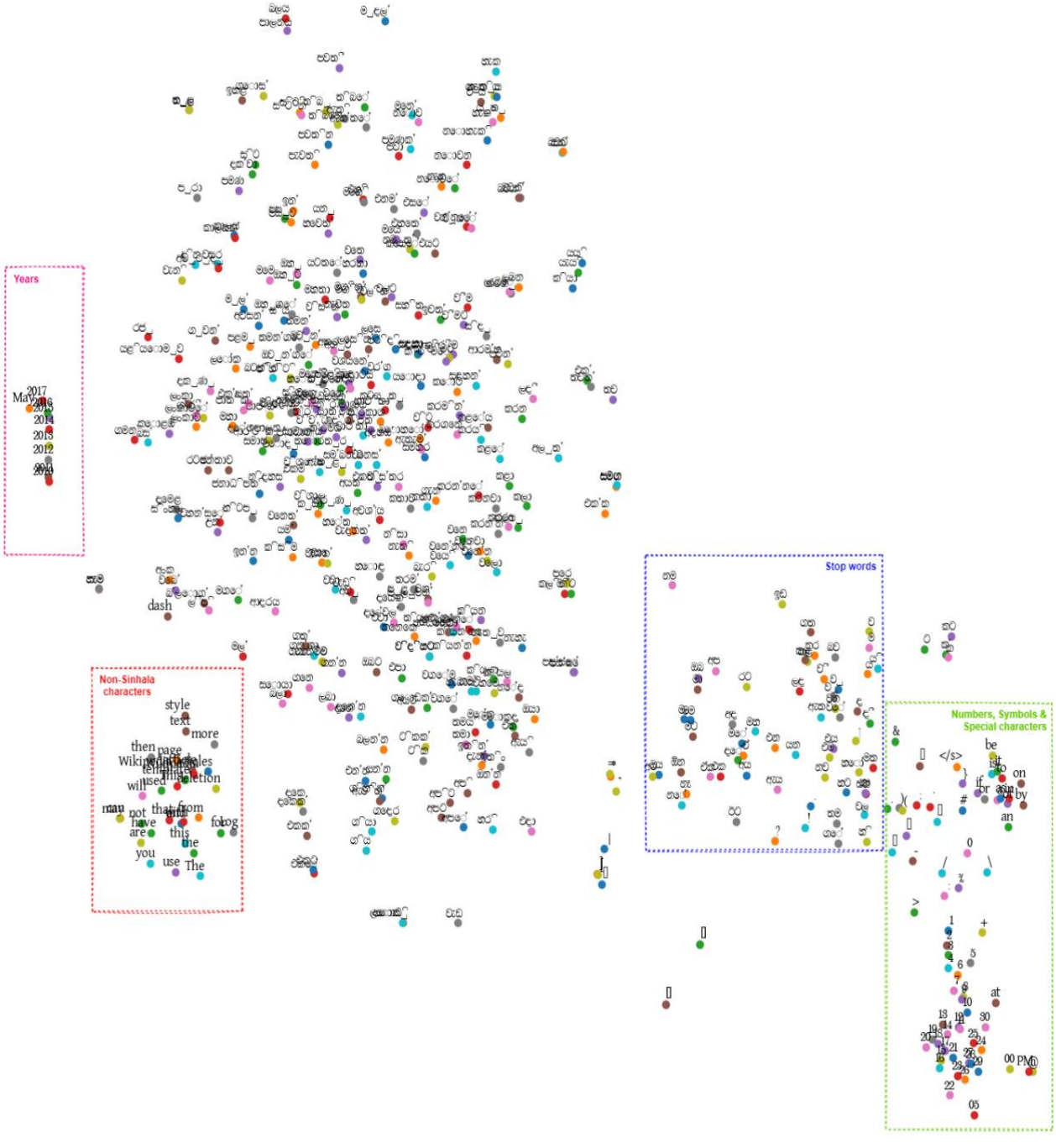


Figure 3.4: Visualization of Sinhala dataset using t-SNE [72]

The visualization shows that our embedding models have noisy, incorrect data such as words that are mapped together with numbers, symbols, special characters, and stop words in vector space. It gives us insightful information as to what necessary preprocessing steps need to be taken in order to enhance the data quality in the embedding layer.

3.3 Preprocessing

Preprocessing is a crucial task in any text-processing pipeline that needs to deal with noisy, incorrect data. Proper preprocessing techniques (shown in Figure 3.5) have to be in-place to eliminate unnecessary information and improve the quality of our datasets before feeding into the embedding layer for word vectorization.

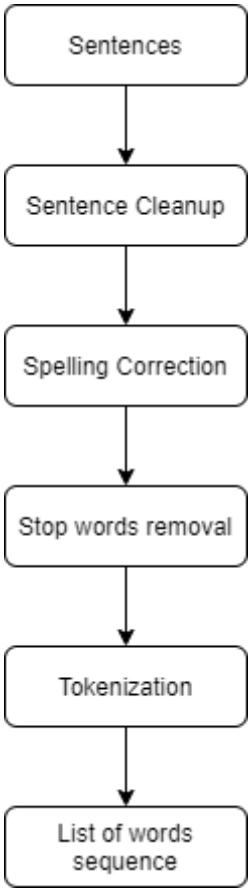


Figure 3.5: Preprocessing steps for short sentence dataset

3.3.1 Sentence cleanup

Our primary focus is to measure the similarity between monolingual short sentences, focusing on Tamil and Sinhala languages. Hence, when building the fastText model, we removed all Non-Sinhala and Non-Tamil characters. In this process, we used regular expressions to filter out non-Tamil, non-Sinhala characters that include numbers, punctuations, question marks, and quotation marks, symbols, and special characters.

- Numbers
- Symbols

- Punctuations marks
- Question mark
- Special characters.
- Strip whitespace

3.3.2 Stop words removal

In any language, words such as articles, pronouns, and certain verbs are frequently treated as stop words. It does not deliver any unique information that can be used to get the context or true meaning of the sentence. Stop words can be eliminated without any harmful impact to the final model. We removed stop words since they contributed no unique information to our model. We used the NLTK library and provided lists of stop words for Sinhala [73] (shown in Table 3.6) and Tamil [74] (shown in Table 3.7) languages for stop word removal. Short sentences were divided into a list of words and then removed if it existed in the provided stop words list.

The following list of Sinhala stop words was used in preprocessing for Sinhala short sentences and was created as a by-product of "Word Embedding Evaluation for Sinhala" research done by Lakmal et al. [73].

Table 3.6: List of Sinhala stop-words

සහ	නිසා	අතර	සහ	පවා	ඔන්න
සමග	නිසාවෙන්	විසින්	දක්වා	ද	මෙන්න
සමඟ	බවට	සමග	ට	හෝ	උදෙසා
අහා	බව	පිළිබඳව	ගේ	වත්	පිණිස
ආහ්	බවෙන්	පිළිබඳ	එ	විනා	සඳහා
ආ	නම්	තුළ	ක	හැර	අරබයා
ඕහෝ	වැඩි	බව	ක්	මිස	නිසා
අනේ	සිට	වැනි	බවත්	මුත්	එනිසා
අඳෝ	දී	මහ	බවද	කිම	එබැවින්
අපොයි	මහා	මෙම	මත	කිම්	බැවින්
අපෝ	මහ	මෙහි	ඇතුලු	ඇයි	හෙයින්
අයියෝ	පමණ	මේ	ඇතුළු	මන්ද	සේක්
ආයි	පමණින්	වෙත	මෙසේ	හෙවත්	සේක
ඌයි	පමන	වෙතින්	වඩා	නොහොත්	ගැන
චී	වන	වෙතට	වඩාත්ම	පතා	අනුව

විහ්	විට	වෙනුවෙන්	නිති	පාසා	පරිදි
වික්	විටින්	වෙනුවට	නිතිත්	ගානෙ	විට
හෝ	මේ	වෙන	නිතොර	තව	තෙක්
දෝ	මෙලෙස	ගැන	නිතර	ඉතා	මෙතෙක්
දෝහෝ	මෙයින්	නෑ	ඉක්බිති	බොහෝ	මේතාක්
මෙන්	ඇති	අනුව	දැන්	වහා	තුරු
සේ	ලෙස	නව	යලි	සෙද	තුරා
වැනි	සිදු	පිළිබඳ	පුන	සැනින්	තුරාවට
බඳු	වශයෙන්	විශේෂ	ඉතින්	හනික	තුලින්
වන්	යන	දැනට	සිට	එම්බා	නමුත්
අයුරු	සඳහා	එහෙන්	සිටන්	එම්බල	එනමුත්
අයුරින්	මගින්	මෙහෙන්	පටන්	බොල	වස්
ලෙස	හෝ	එහේ	තෙක්	නම්	මෙන්
වැඩි	ඉතා	මෙහේ	දක්වා	වනාහි	ලෙස
ශ්‍රී	ඒ	ම	සා	කලී	පරිදි
හා	එම	තවත්	තාක්	ඉඳුරා	එහෙත්
ය	ද	තව	තුවක්	අන්න	

The following list of Tamil stop words were used in preprocessing for Tamil short sentences and was created by the TamilNLP[1] team [74] using Wikipedia Tamil text corpus[2] using TXM Corpus Analysis Tool.

Table 3.7: List of Tamil stop-words

ஒரு	போன்ற	அவள்	இந்தப்	இதனால்	எனவும்
என்று	எந்த	நீ	எனும்	அவை	எனப்படும்
மற்றும்	வந்து	ஆகிய	மேல்	அதே	எனினும்
இந்த	இதன்	இருந்தது	பின்	ஏன்	அடுத்த
இது	அது	உள்ளன	சேர்ந்த	முறை	இதனை
என்ற	அவன்	வந்த	ஆகியோர்	யார்	இதை
பேர்	தான்	இருந்த	எனக்கு	என்பதை	கொள்ள
என்பது	பலரும்	மிகவும்	இன்னும்	எல்லாம்	இந்தத்
பல	என்னும்	இங்கு	அந்தப்	மட்டுமே	இதற்கு

[1] https://github.com/AshokR/TamilNLP/wiki/Stopwords
[2] https://github.com/AshokR/TamilNLP/wiki/Wikipedia-Tamil-Text-Corpus

ஆகும்	மேலும்	மீது	அன்று	இங்கே	அதனால்
அல்லது	பின்னர்	ஓர்	ஒரே	அங்கே	தவிர
அவர்	கொண்ட	இவை	மிக	இடம்	போல
நான்	இருக்கும்	இந்தக்	அங்கு	இடத்தில்	வரையில்
உள்ள	தனது	பற்றி	பல்வேறு	அதில்	சற்று
அந்த	உள்ளது	வரும்	விட்டு	நாம்	எனக்
இவர்	போது	வேறு	பெரும்	அதற்கு	என்
என	என்றும்	இரு	அதை	எனவே	வரை
முதல்	அதன்	இதில்	பற்றிய	பிற	மட்டும்
என்ன	தன்	போல்	உன்	சிறு	கொண்டு
இருந்து	பிறகு	இப்போது	அதிக	மற்ற	வேண்டும்
சில	அவர்கள்	அவரது	அந்தக்	விட	

3.3.3 Spelling correction

Spelling correction is the preliminary task of detecting and correcting spelling errors or misspelling in the dataset in the preprocessing phase. It is a crucial task in any major NLP pipeline, especially in text-processing tasks that need to deal with noisy, incorrect data in the dataset.

Mainly there are two types of spelling correction that are performed in our dataset in order to correct spelling errors and improve the sentence quality in the datasets.

- Traditional spell correction
- Contextual spell correction

3.3.3.1 Traditional spell correction

Traditional spell correction is a process of identifying and correcting the spelling errors and misspelled words in the sentence. We used the ***tamilspellchecker*** python library for spelling correction for Tamil[3].

Example:
This library returns the correct word or list of possible correct words for given spelling error word or misspelled word.

[3] https://pypi.org/project/tamilspellchecker/

```
>>> from TamilSpellingAutoCorrect import TamilSpellingAutoCorrect
>>> spellchecker  = TamilSpellingAutoCorrect("tamil_bloom_filter.txt","tamilwordlist.txt")
>>> from_spell_checker_list = spellchecker.tamil_correct_spelling("செலுதட்துகின்றார்")
>>> print(from_spell_checker_list)
['செலுத்துகின்றார்']

>>> from TamilSpellingAutoCorrect import TamilSpellingAutoCorrect
>>> spellchecker  = TamilSpellingAutoCorrect("tamil_bloom_filter.txt","tamilwordlist.txt")
>>> from_spell_checker_list = spellchecker.tamil_correct_spelling("செற்றைச்")
>>> print(from_spell_checker_list)
['சோற்றைச்', 'செற்றைச்', 'செற்றைப்', 'நெற்றைச்', 'செற்றுச்', 'செற்றச்', 'சேற்றைச்', 'சாற்றைச்']
```

This library is unable to correct certain words due to the incompleteness of this tool. The accuracy of the ***Tamilwordchecker*** library depends on the number of unique words in the wordlist. To improve the accuracy, we need to add more unique words to the wordlist.

```
>>> from TamilSpellingAutoCorrect import TamilSpellingAutoCorrect
>>> spellchecker  = TamilSpellingAutoCorrect("tamil_bloom_filter.txt","tamilwordlist.txt")
>>> from_spell_checker_list = spellchecker.tamil_correct_spelling("துள்ளுகின்றார்")
>>> print(from_spell_checker_list)
[]
```

3.3.3.2 Contextual spell correction

Contextual spell correction is the process of identifying and correcting contextual spelling errors or the use of an incorrect word, even though it is a valid word in a particular sentence or context. Traditional spell correction checks and corrects misspelled words, but they fail to distinguish words that are erroneously placed in a sentence. Due to the unavailability or incompleteness of contextual spelling correction tools, we manually analyzed and performed contextual spelling correction in our datasets.

Example in Tamil sentence:

Before correction:

- வாத்து தன் ***இலங்கையை*** விரித்தபடி காணப்படுகின்றது
  (The duck is seen spreading its Sri Lanka)

After correction:

- வாத்து தன் ***இறக்கையை*** விரித்தபடி காணப்படுகின்றது
  (The duck is seen spreading its wings)

Example in Sinhala sentence:

Before correction:

- කමිසයක් පැළඳ නොමැති තරුණයා ස්කේටිං ක්‍රීඩාව ***වෙයි***
  (Shirtless man becomes skating)

After correction:

- කමිසයක් පැළඳ නොමැති තරුණයා ස්කේටිං ක්‍රීඩාව ***කරයි***
  (A shirtless man is skating)

3.3.4 Tokenization

Tokenization splits the raw text into individual words or terms called tokens. These tokens assist in identifying the context of text and improving the NLP models. Through evaluating the series of words, tokenization assists in reading the context of the language. We used the Python split() function to tokenize each sentence into vectors of words.

After performing all the aforementioned preprocessing actions, data quality improved due to the abolition of noise and avoidable information.

3.4 Architecture and Implementation

We experimented with cosine similarity as a baseline technique and measured the similarity between short sentences. And then we used our proposed Siamese networks approach for measuring the similarity between short sentences.

3.4.1 Experiment with Siamese networks

We adopted the MaLSTM model by Mueller et al. [12] and implemented a sentence similarity measurement method using Deep Learning techniques. Since this methodology is mostly language-independent, it can be utilized for Tamil and Sinhala languages with suitable preprocessing techniques and word embedding models.

MaLSTM has been trained on short sentence pairs to learn a highly structured space of sentence representations that captures rich semantics. The simplified architecture of our MaLSTM model is shown in Figure 3.6.

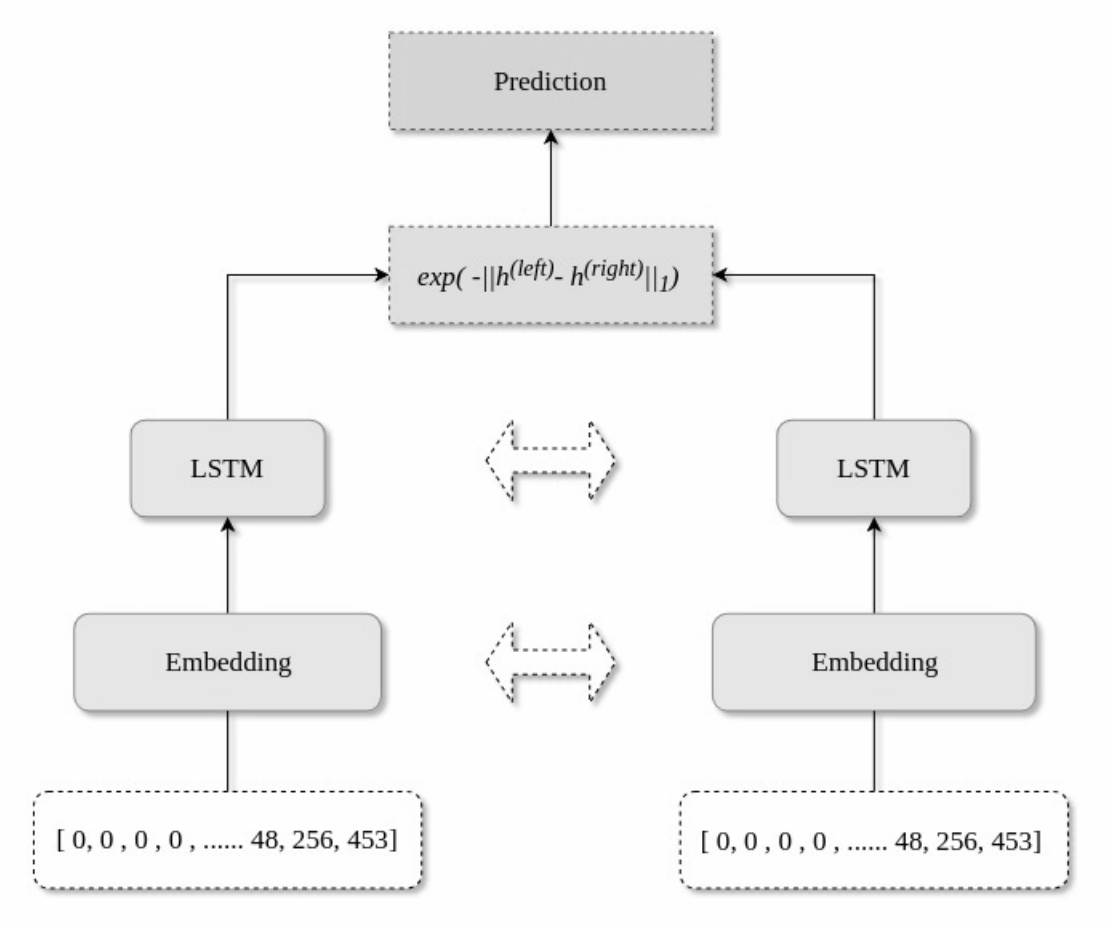


Figure 3.6: Simplified architecture of MaLSTM model

The overall goal of MaLSTM is to assess a pair of short sentences to determine whether they are semantically similar or not. As shown in Figure 3.7, our MaLSTM model consists of a Siamese network that has two identical LSTM networks: $LSTM_{left}$ and $LSTM_{right}$ with shared weights. Each LSTM network is responsible for processing the sentences and updating the hidden states. Sentences are passed as word vector representations and return a hidden state encapsulating the sentence's semantic meaning. The zero-padded sequences of word indices are given as input to the LSTM network. These inputs are vectors of fixed-length where the first zero is disregarded, and non-zero is the index that uniquely identifies the word.

The fastText [63] word-embedding model is used to map words to vectors. Subsequently, the Manhattan similarity function is used for comparing the final LSTM networks' hidden states and output a similarity score. The process is shown in Figure 3.7. Our model comprises four main layers: preprocessing layer, word-embedding layer, hidden layer, and sentence similarity measurement layer.

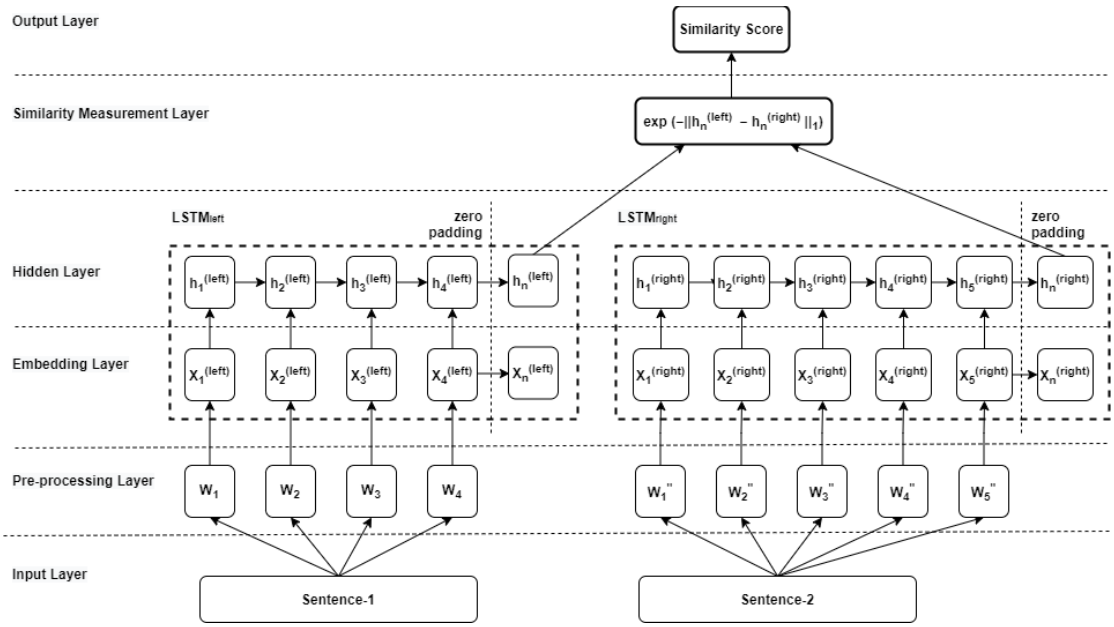


Figure 3.7: System for short sentence similarity measurement [12]

3.4.2 Experiment with Cosine similarity

Cosine similarity is used as a baseline technique for measuring similarity between pairs of short sentences in both Sinhala and Tamil languages. The cosine similarity score is compared with manually annotated similarity score, previous hybrid approaches, and finally with Siamese Networks based approach.

3.5 Embedding Layer

Deep Learning models understand input in the form of vectors. In order to input our dataset into Deep Learning models, every sentence in the dataset needs to be tokenized into a list of words and then vectorized.

The Embedding layer accepts sentence pairs as word sequences from the pre-processing layer and vectorizes words into vector representation. We utilized fastText as the word-embedding model, which builds word vectors using the skip-gram approach, with each word represented as a string of n-grams characters. The fastText model is used to learn word embeddings as the average of its character embeddings, which accounts for the morphological makeup of the word, demonstrating its efficacy in Tamil and Sinhala languages. Word vectors are allocated to even words with no lexical match based on their characters or subunits [63, 75].

Zero Padding

Sentences are in different lengths in a sentence dataset. However, deep neural networks require inputs of the same size. Zero-padding is done for this purpose. Zero-padding is a simple process where zeros are added to the end of a sentence to expand its length. Zero-padding is done in a pair of short sentences and makes sure that sentences have equal lengths before passing them into the deep neural networks.

3.6 Hidden Layer

LSTM networks are good at handling variable-length sequential data such as sentences. LSTM updates its internal states such as input vector x_t, memory state c_t, and hidden state h_t by performing calculation using the equations (3.2) – (3.7):

$$i_t = sigmoid(W_i x_t + U_i h_{t-1} + b_i) \quad (3.2)$$

$$f_t = sigmoid\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (3.3)$$

$$\tilde{c}_t = tanh\ (W_c x_t + U_c h_{t-1} + b_c) \quad (3.4)$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \quad (3.5)$$

$$o_t = sigmoid(W_o x_t + U_o h_{t-1} + b_o) \quad (3.6)$$

$$h_t = o_t \odot tanh(c_t) \quad (3.7)$$

where, i_t: input, o_t: output, and f_t: forget gates, and memory state $\boldsymbol{c_t}$ at t,: time, respectively. The LSTM weight matrices parameters are W_k, U_k and the bias vector is b_k for each k in $\{i, f, c, o\}$ and an element-wise product of matrices is denoted by $\odot$.

The LSTM algorithm learns a mapping from a space of variable length sequences and encodes the input sequences into a fixed-dimension hidden state representation. Expressed differently, each sentence is represented as a sequence of words (e.g., sentence1 is represented by w_1, w_2, w_3), and words vector sequence (e.g., sentence1 is represented by x_1, x_2, x_3) is input into the LSTM network. It is updated at each index sequence, its hidden state. The final hidden state of LSTM for each sentence is an n-dimensional vector, which holds the semantic meaning of the sentence. The most important feature of the Siamese architecture is its ability to share weights across the two LSTM sub-networks, which reduces the total number of parameters, and the tendency of over-fitting in turn.

3.7 Sentence similarity measurement layer

Similarity between words is measured in metrics in vector space using various distance measurements such as Manhattan distance, Euclidean distance, Cosine distance, and Hamming distance. When the data has high-dimensionality, Manhattan distance is usually preferable to the Euclidean distance, while distance between unqualified variables is calculated using the Hamming distance, and the cosine distance is mainly used to determine the similarity between two data points.

The final hidden states of two LSTM networks are compared using the Manhattan distance function as given in Equation (3.8). Alternatives such as Euclidean or cosine distance can also be used, but Mueller et al. [11] used the Manhattan distance for English, and it marginally outperforms other sensible alternative similarity functions.

$$Sim_{manhattan} = exp\left(-\|h_n^{left} - h_n^{right}\|_1\right) \qquad (3.8)$$

Manhattan similarity function outputs the similarity scores as values between 0 and 1, due to the exponent of a negative in function.

4. SYSTEM EVALUATION AND RESULTS

This chapter explores the experiment results comparing the performance of different features and similarity measurement algorithms. Furthermore, it compares experiment results with existing short sentence similarity measurement research for Tamil and Sinhala languages [2, 3].

4.1 Baseline experiment

Kadupitiya et al. [2] and Anutharsha et al. [3] carried out sentence similarity measurement research for short phrases for the Sinhala and Tamil languages (respectively). These research works use unsupervised hybrid approach techniques that combine knowledge based and corpus based similarity measures (called semantic similarity) and similarity measures using word-order information [2, 3]. These are the only existing short sentence similarity measurement research in Sinhala and Tamil languages.

Table 4.1 summarizes the results of Anutharsha et al. [3], and Kadupitiya et al. [2]. We used the same dataset in our experiments for direct comparison of results from both approaches.

Table 4.1: Baseline experiment results

Language	**Pearson correlation coefficient**	**Mean Squared Error**
Sinhala	0.832	0.145
Tamil	0.815	0.195

4.2 Effects of preprocessing

Preprocessing is an important step for text processing tasks in NLP. It transforms text into a more digestible form so that Deep Learning algorithms can perform better. We decided to conduct a few experiments to understand and analyze the effects of preprocessing in a short sentences' dataset with our model.

- Effects of cleanup
- Effects of stop word removal
- Effects of spell correction

4.3.1 Effects of cleanup

We identified the list of commonly occurring punctuations, numbers, symbols, and special characters while skimming our sentence dataset. Punctuations mostly carry specific information in the sentences while others add noise to the data. Even though full stop, comma, question mark, exclamation mark, and colons are important punctuations in sentences, introducing noise when building the fastText model, we decided to remove all non-Sinhala/non-Tamil characters. We conducted a few experiments to understand the behavior of sentence cleanup. The following datasets were used to analyze the importance of sentence cleanup.

- Without sentence cleanup in Tamil dataset
- With sentence cleanup in Tamil dataset
- Without sentence cleanup in Sinhala dataset
- With sentence cleanup in Sinhala dataset

Table 4.2: Sentence cleanup experiment results

Language	**Without sentence cleanup ***		**With sentence cleanup**	
	Pearson correlation coefficient	**Mean Squared Error**	**Pearson correlation coefficient**	**Mean Squared Error**
Tamil	0.823	0.196	0.841	0.187
Sinhala	0.834	0.189	0.862	0.175

* Performed all preprocessing steps except sentence cleanup.

4.3.2 Effects of stop word removal

Stop word removal is one of the most commonly used preprocessing steps in NLP text-processing applications. Stop words are usually eliminated from sentences before Deep Learning models are trained since they are commonly uttered and hence do not contribute any unique information that can be utilized to calculate sentence similarity. Also, by removing stop words, the dataset size decreases, and less time is taken to train the model. Stop word removal can potentially help in improving performance because there are fewer and only significant tokens left. Thus, the classification exactitude could be enhanced. The idea is simply removing the words that occur commonly across all the documents in the corpus.

We provided lists of hand-curated stop words for Sinhala [73] (shown in Table 3.6) and Tamil [74] (shown in Table 3.7) languages. Short sentences were divided into a list of words and then removed if it exists in the provided stop words list.

We conducted a few experiments to understand the behavioral effects of stop words removal. The following datasets were used to analyze the importance of stop word removal in the sentence datasets.

- Without stop words removal in Tamil dataset
- With stop words removal in Tamil dataset
- Without stop words removal in Sinhala dataset
- With stop words removal in Sinhala dataset

Table 4.3: Stop words removal experiment results

Language	**Without stop word removal ***		**With stop words removal**	
	Pearson correlation coefficient	**Mean Squared Error**	**Pearson correlation coefficient**	**Mean Squared Error**
Tamil	0.813	0.209	0.841	0.187
Sinhala	0.827	0.193	0.862	0.175

* Performed all preprocessing steps except stop words removal in sentences.

4.3.3 Effects of spell correction

Spelling correction techniques are used for text processing to achieve better results. We decided to conduct a few experiments to understand the behavior of these spelling corrections. The following datasets were used for testing the importance of spelling correction:

- Without doing spelling correction in Tamil dataset
- With the spell correction in Tamil dataset
- Without doing spelling correction in Sinhala dataset
- With the spell correction in Sinhala dataset

We identified the following spelling errors or misspelled words (shown in Table 4.4) while skimming the Tamil dataset. Similarly, the following spelling errors or

misspelled words (shown in Table 4.5) were identified while skimming the Sinhala dataset.

Table 4.4: Spelling errors or misspelling words in Tamil dataset

Spelling errors word	**Correct spelling words**	**English Translation**
செலுதட்துகின்றார்	செலுத்துகின்றார்	Pay, Drive
துரத்துகிறகனது	துரத்துகின்றது	Chasing
துள்ளுகுின்றார்	துள்ளுகின்றார்	Leaps
ஓடட	ஓட்ட	Run
இயறகையை	இயற்கையை	Nature
ஒருவுர்	ஒருவர்	A person
செற்றைச்	சேற்றைச்	Mud
வீர்ர்	வீரர்	Player
மடடை	மட்டை	Bat
வ்லகூப்பாட்டில்	Unknown	Unknown

* English Translation some words might not be the same meaning as in Tamil

Table 4.5: Spelling errors or misspelled words in Sinhala dataset

Spelling errors word	**Correct spelling word**	**English Translation**
වින්සුරුවෙකු	විනිසුරුවෙකු	Judge or umpire in sport
පැහනි	පැහැති	Used to describe color.
පුදගලයෙක්	පුද්ගලයෙක්	A person
ගමින් ලරන	ගමන් කරන	Traveling
දමමින්	දමමින්	Placing / Putting
නෙවක්	නැවක්	A ship
අරමින්	කරමින්	Doing
රෝදා	රෝද	Wheels
ක්‍රීඩිකාව කගේ	ක්‍රීඩිකාවකගේ	Sportswoman`s.

පනියි	පනියි	Jumps

* English Translation some words might not be the same meaning as in Sinhala

We experimented with the above scenarios with our model and compared the results as shown in Table 4.6.

Table 4.6: Sentence spelling correction experiment results

Language	**Without spelling correction ***		**With spelling correction**	
	Pearson correlation coefficient	**Mean Squared Error**	**Pearson correlation coefficient**	**Mean Squared Error**
Tamil	0.8452	0.187	0.841	0.185
Sinhala	0.8636	0.175	0.862	0.172

* Performed all preprocessing steps except spelling correction in sentences.

Observed that performance gain with the spell correction was comparatively small and it did not improve the accuracy much. This behavior shows that datasets have fewer spelling errors or misspelled words. Reason might be the datasets for Tamil and Sinhala were constructed using a well-organized team (5 participants & 3 judges) and had a rare chance of spelling errors in these datasets.

4.3 Model evaluation

We split datasets into three sections for training, validation, and testing, where 80% of the dataset is used for training, and 10% is used for validation, and the remaining 10% used for testing. In the training phase, system performance was evaluated based on the accuracy (Pearson correlation coefficient) and loss function (Mean Squared Error) against the manually annotated score. This approach measures the similarity of short sentences based on the gold relatedness score (shown in Table 3.1), same way as Mueller et al [12] used in their approach.

We tried out various parameters (shown in Table 4.7) and repeated testing by changing values until our model achieved a better performance with a combination of these parameters with optimistic values.

Table 4.7: MaLSTM model parameters

Parameters	**Values**
Number of hidden states of LSTM cells	50, 100
Activation function of LSTM cells	Sigmoid
Model optimizer	Adam with learning rate 0.001
Batch size	16, 32
Number of epochs	50, 100

We used Intel® Core™ i5 Processors 7200U, CPU @ 2.50GHz 2.70 GHz, and 16GB RAM for training. As a result, for Sinhala (shown in Figure 4.2), it has reached 91.44% training accuracy & 86.79% validation accuracy after 50 epochs. For Tamil (shown in Figure 4.1), it has reached 86.53% training accuracy & 84.90% validation accuracy after 100 epochs.

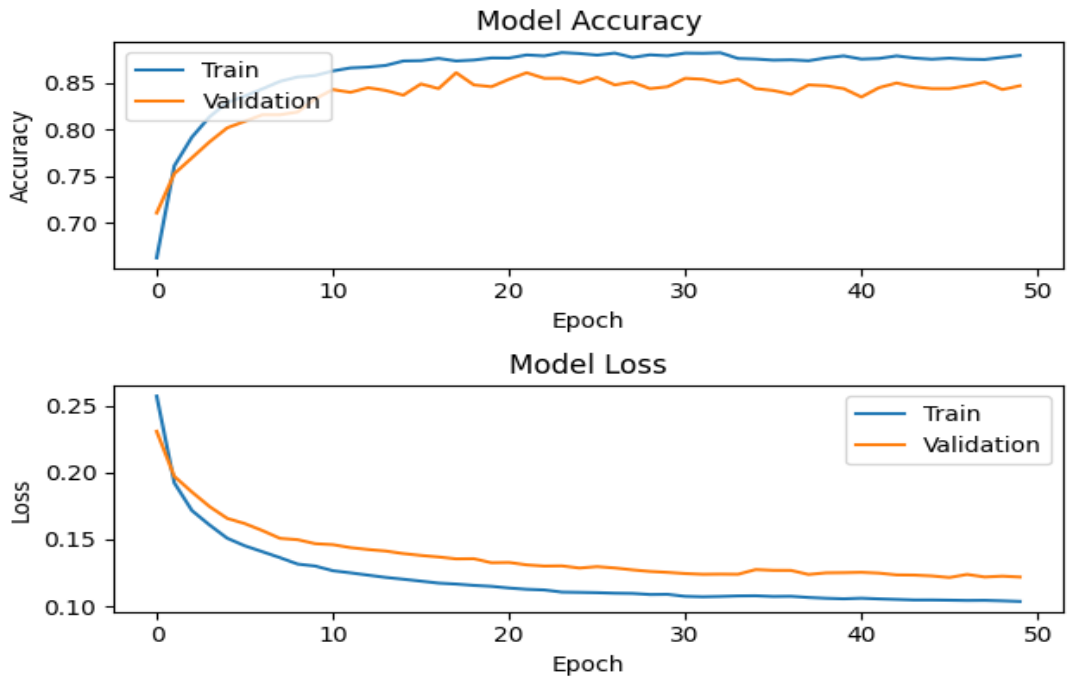


Figure 4.1: Model training and validation performance for Tamil dataset

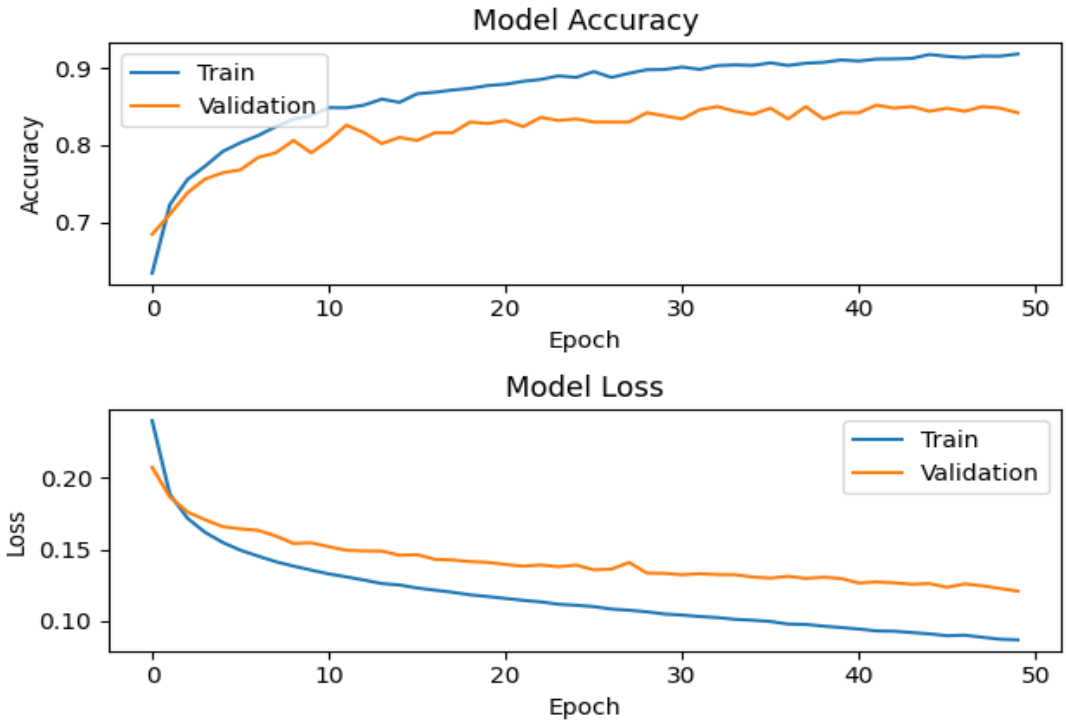


Figure 4.2: Model training, validation performance for Sinhala dataset

Table 4.8: Tamil - Short sentence similarity score comparison

Sentences	**Manual Similarity Score**	**Hybrid Approach [2] Score**	**Cosine Similarity Score**	**Our Approach Score**
A: ஓட்டப்பந்தய காரொன்று சிறிய வளைவொன்றில் செல்கின்றது (A racing car is going into a small curve) B: ஓட்டப்பந்தய காரொன்று வளைவொன்றில் செல்கின்றது (A racing car is going into a curve)	0.91	0.82	0.96	0.94
A: வீரர் ஒருவர் மலையில் ஏறுகின்றார் (A player climbs the mountain) B: ஒருவர் கயிற்றினைப்பிடித்தவாறு மலையில் ஏறுகின்றார் (A person climbs the mountain holding a rope)	0.55	0.45	0.67	0.47

A: கால்ப்பந்தாட்டத்தில் வீரரொருவர் பந்தை அடிக்கின்றார் (A player hits the ball in football) B: கால்பந்து வீரர்கள் பந்தை ஒருவரிடம் இருந்து ஒருவர் பறிக்க நினைக்கின்றனர் (Football players try to snatch the ball from one another)	0.20	0.02	0.73	0.12
A: பனி சறுக்கல் வீரர் பறந்து சாகசம் காட்டுகின்றார் (A Snowboarder fly and shows adventure) B: பனி மலையில் சறுக்கியவண்ணம் ஒருவர் செல்கிறார் (A person is sliding on a snowy mountain)	0.48	0.41	0.57	0.51
A: நான்கு சிறுவர்கள் கூடைப்பந்தாட்ட விளையாட்டில் ஈடுபடுகின்றனர் (Four children play basketball game) B: கூடைப்பந்தாட்டத்தில் பிள்ளைகள் ஈடுபடுகின்றனர் (Children are engaged in basketball)	0.62	0.48	0.77	0.53

4.4 Results summary

We evaluated our system with a 10% short sentence datasets and reported results in Pearson correlation coefficient (Pearson's r) and Mean Squared Error (SME). Table 4.8 shows a comparison of similarities between manually annotated score, previous hybrid approach [3], cosine similarity and our approach for Tamil short sentence pairs.

As shown, our approach is able to measure the similarity score that is closer to the manually annotated score and outperforms the cosine similarity technique and previous hybrid approach for Tamil. We obtained Pearson correlation coefficient of 0.841 and SME of 0.187 (shown in Figure 4.4).

This result shows that the Pearson correlation coefficient is 3.07% higher than what was reported by Anutharsha et al. [3].

Table 4.9: Sinhala - Short sentence similarity score comparison

Sentences	**Manual Similarity Score**	**Hybrid Approach [2] Score**	**Cosine Similarity Score**	**Our Approach Score**
A: කබරයෝ දෙදෙනෙක් වතුර කඩිත්තක අරගලයක නිරත වෙති (two monitors fights in a stream of water) B: කිඹුලන් දෙදෙකු වැවක සිටියි (two crocodiles in a lake)	0.67	0.55	0.39	0.71
A: රතු පැහැති මෝටර් රථයක් වේගයෙන් ධාවනය වේ (a red car drives fast) B: මිනිසෙක් යතුරු පැදියක් ධාවනය කරයි (A man drives a motor bike)	0.43	0.51	0.37	0.48
A: මෝටර් රථ යාන්ත්‍රිකයෙක් කාර් එකක උවසුන වෙනට බටයකින් වතුර පහරක් එල්ල කරයි (A motor mechanic aims water to the bonnet using a hose) B: මිනිසෙක් ඔසවා ඇති මෝටර් රථයක් සෝදමින් සිටියි (A man washes a car that is lifted up)	0.97	0.83	0.38	0.94
A: යතුරු පැදි ධාවන තරඟයක යෙදෙන ක්‍රීඩකයෙක් (An athlete who plays in a motor bike racing competition) B: යතුරු පැදි ධාවකයෙක් වමට වංගුවක හැරෙයි (motorcycle rider turns left on a bend)	0.87	0.81	0.67	0.93
A: ක්‍රීඩකයෙක් ඔහුගේ පිත්ත ඔසවයි (An athlete lifts his bat)	0.77	0.92	0.45	0.82

B: ක්‍රීඩකයෙකු බේස් බෝල් ක්‍රීඩාවේ යෙදී සිටී (An athlete plays baseball)				

Similarly, for Sinhala 10% of the dataset was used for testing. As shown in Table 4.9, our approach is able to measure the similarity score that is closer to the manually annotated score and outperforms the previous hybrid approach for Sinhala. Our approach obtained an SME of 0.175 and Pearson correlation coefficient of 0.862 (shown in Figure 4.3). This result shows that the Pearson correlation coefficient is 3.61% higher than the result reported by Kadupitiya et al. [2].

Table 4.10: Performance comparison using Pearson correlation coefficient.

Language	Hybrid Approaches [2, 3]	Monolingual Sentence Similarity Measurement using Siamese Neural Networks for Sinhala and Tamil Languages	
		Cosine Similarity	Siamese Networks
Sinhala	0.832	0.217	**0.862**
Tamil	0.815	0.091	**0.841**

As can be seen in Figure 4.3 and 4.4, our approach is mostly able to predict a similarity score that is closer to the manually annotated score for both Sinhala and Tamil. But in certain circumstances, it was unable to predict a similarity score close to the manual score. This may be due to the influence of loan words. It is briefly discussed in the following error analysis section.

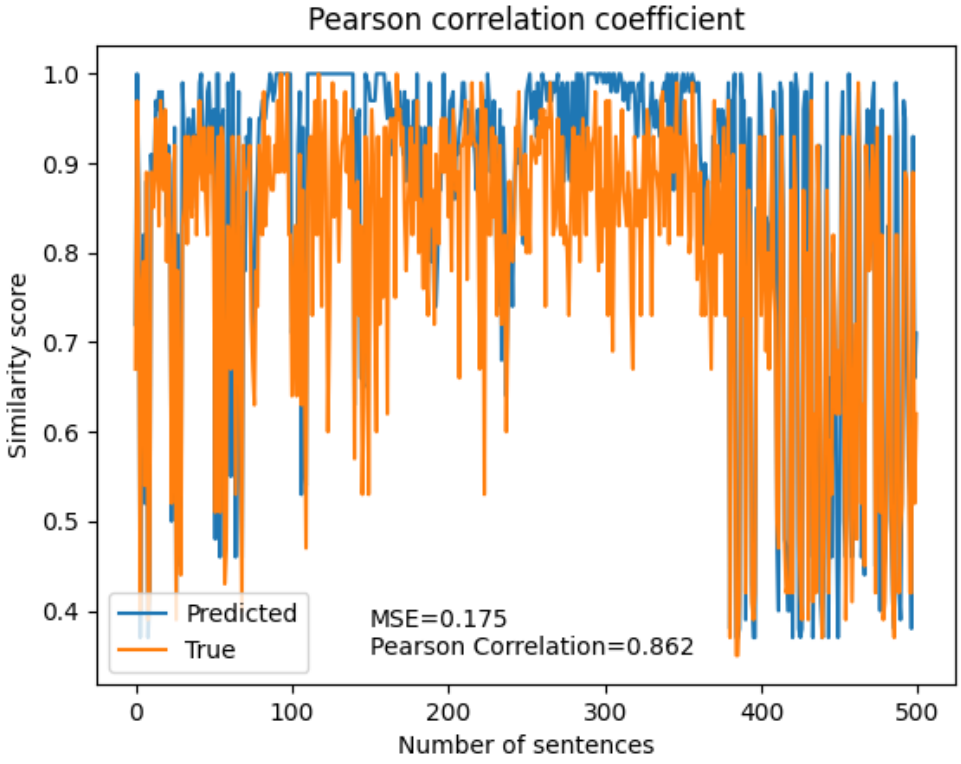


Figure 4.3: Model evaluation performance for Sinhala.

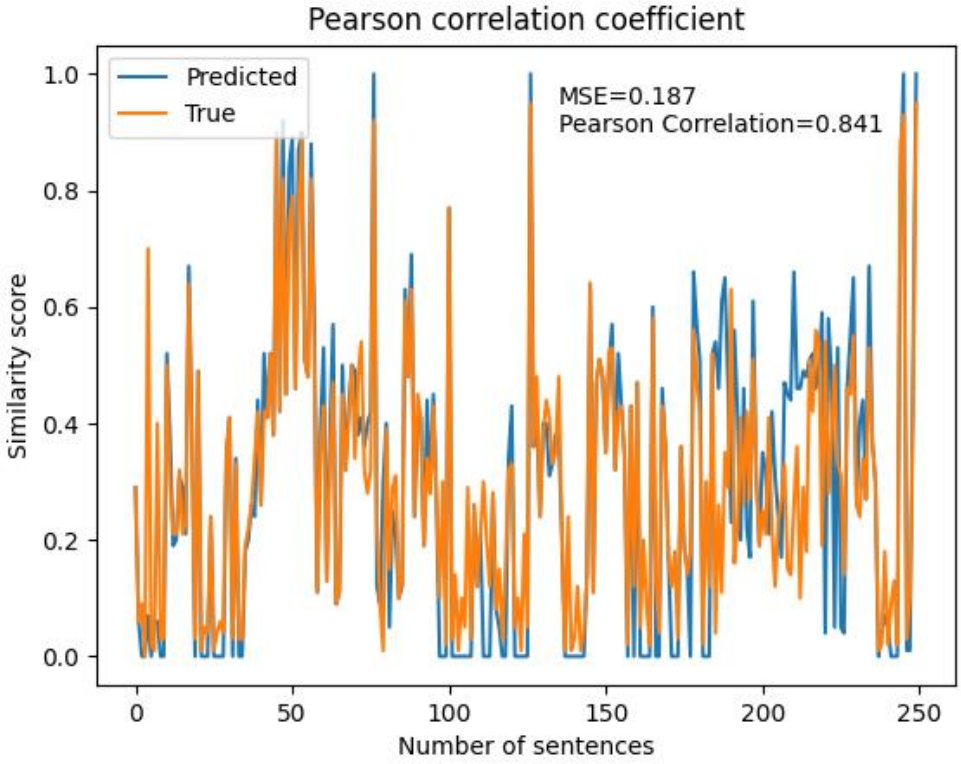


Figure 4.4: Model evaluation performance for Tamil.

4.5 Error Analysis

A loanword is a word that has been borrowed from one language and used in another without being translated. Sri Lanka was colonized by the Portuguese, Dutch, and English and as a consequence, a significant number of loanwords from Portuguese, Dutch, and English have been borrowed for use in both languages. The majority of these loanwords is used in the text contents on the internet and became part of the Tamil/ Sinhala language.

Table 4.11: Loanwords in Tamil dataset

Tamil loanwords	**Tamil word**	**English Meaning**
சைக்கிள்	மிதிவண்டி, ஈருளி, துவிச்சக்கரவண்டி	Bicycle
ஜிம்னாஸ்த்திக்	சீருடற்பயிற்சி	Gymnastics
டென்னிஸ்	தட்டுப்பந்து, வரிப்பந்தாட்டம்	Tennis
டிவ்	முழுக்கு, (நீரில்)மூழ்கு, முக்குளிப்பு	Dive
பற்மின்ரன்	பூப்பந்தாட்டம், பூப்பந்து	Badminton
பேஸ்பால்	எல்லை, தளக்கட்டுப் பந்தாட்டம்	Baseball
ஸ்கேட்டிங்	சறுக்கு, பனிச்சறுக்கு	Skating
கித்தாரை	நரம்பள்ள இசைக்கருவி	Guitar

Some sentences in our datasets were constructed using loanwords that are not the actual Tamil/ Sinhala words, but are directly borrowed from other languages (Ex: English). Following Table 4.11 and Table 4.12 shows a list of loanwords from the dataset for both Sinhala and Tamil languages (respectively).

Table 4.12: Loanwords in Sinhala dataset

Sinhala loanwords	**Sinhala word**	**English Meaning**
ස්කේටිං	අයිස් මත ලිස්සා යාම	Skating
ජිම්නාස්ටික්	සරඹ පිළිබඳ වූ	Gymnastic

බාස්කට් බෝල්	පැසිපන්දු	Basketball
ෆුට්බෝල්	පාපන්දු	Football
ජැකැට්ටුවක්	බාච්චුව	A Jacket
පිරමීඩයක්	චතුර්භුජාංශ ස්තූපය	A Pyramid
පැරෂුට්	වායුච්ඡත්‍රය	Parachute
බොක්සිං	ගුස්ති	Boxing

'சைக்கிள்' is a loanword for '*Bicycle'* whereas actual Tamil words are '*மிதிவண்டி'* or '*ஈருளி'* or '*துவிச்சக்கரவண்டி'*. Similarly *'ෆුට්බෝල්'* is a loanword for Football whereas the actual word is *'පාපන්දු'.*

Since loanwords are treated as the actual word and used in the same context and carry the same meaning in the sentences, we are expected to have the same similarity score when we measure the similarity between actual word and the loanword. The similarity score has to be equal or mostly equal to 1 but in a real scenario, it is not.

We checked the similarity between the actual word and loanword using both cosine similarity and our siamese network based approach. Our model was able to calculate a similarity score slightly better than the cosine similarity function.

The following example shows a similarity calculation between the actual word and the loanword in the Tamil language using cosine similarity function.

```
>>> similarity = fasttext.similarity('சைக்கிள்', 'துவிச்சக்கரவண்டி')
>>> print(similarity)
0.532845
>>>
>>> similarity = fasttext.similarity('சைக்கிள்', 'மிதிவண்டி')
>>> print(similarity)
0.6419646
>>>
>>> similarity = fasttext.similarity('சைக்கிள்', 'ஈருளி')
>>> print(similarity)
0.33304918
```

The following example shows a similarity calculation between the actual word and the loanword in the Sinhala language using cosine similarity function.

```
>>>
>>> similarity =  word2vec.similarity('ෆුට්බෝල්', 'පාපන්දු')
>>> print(similarity)
0.54812344
>>>
```

Loanword problem was not considered in the preprocessing and no cleanup or modifications were done for the loanwords in the previous hybrid approaches for Tamil and Sinhala languages. We need to use the same dataset to compare the performance of both approaches. Hence, we did not consider them into the preprocessing steps for clean up or replace these loanwords with actual words.

5. CONCLUSION

We implemented the pioneering Deep Learning based system for estimating the semantic similarity of short sentences in the Tamil and Sinhala languages. We obtained the best results that could beat the cosine similarity and previous research based on a hybrid approach for the Sinhala, and Tamil languages.

The previous hybrid similarity calculation approaches for Tamil and Sinhala languages depended on its individual techniques and it required NLP resources for similarity measurement. Unlike the previous hybrid approaches, our approach is independent of the language and domain. Thus, our method outperforms the hybrid method used in previous research without the use of lexical resources such as WordNet or dictionaries.

We observed that our system obtained the best results when we used the fastText model. It could be due to the following feature of the fastText model.

- The fastText model ideally deals with highly inflectional languages such as Sinhala and Tamil. It is able to identify inflectional variants of words and allows the embeddings to understand suffixes and prefixes.
- The fastText model worked well with the rare words, and it was able to get vector representation of the words that were not found in the model dictionary during training.
- The lexical database used in hybrid approaches is limited to one to one mappings of similar words. The fastText model was able to accurately identify similar words, even words with different lexical forms.

Also, we observed that Manhattan distance performed better than conventional distance metrics because this metric calculates the sum of the absolute differences in each feature, rather than their squares. Manhattan distance is most preferable when dealing with high dimensional data.

Our approach is able to identify the similarity between a loanword and an actual word better than cosine similarity, but still accuracy is considerably low, which needs to improve in future work.

Despite the fact that numerals, full stops, commas, and question marks have a distinct meaning in sentences, our findings did not support this. As a result, all characters other than Sinhala and Tamil were removed. After conducting the sequence of preprocessing processes, we were able to get the best results by removing extraneous information and therefore improving the data quality of sentences.

Our system achieved 3.07% better results than the previous hybrid approach for Tamil and 3.61% better results than the previous hybrid approach for Sinhala. Both system performances were evaluated based on Pearson correlation coefficient and means squared error.

In addition, we have created a dataset for the Tamil language that contains 500 short sentences and 2500 short sentence pairs. Each short sentence pair is manually annotated with similarity scores by human judges. We have contributed this Tamil short sentence dataset for the community, which can be used for further text analysis.

5.1 Future works

There are several possible ways to enhance the current system performance. The following list of possible future enhancements and improvements can be experimented with the discussed model.

- Contextualized word-embedding models (Ex: BERT, ELMo, etc.) can be used to obtain additional syntactic and semantic information of the languages.
- Current dataset contains limited numbers of sentence pairs and is limited to certain domains, when dataset size increases with a variety of data, it will enable many new avenues.
- Different word embedding methods and combinations of word embedding and traditional features can be experimented.

REFERENCES

[1] Y. Li, Z. A. Bandar, and D. Mclean, "An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources." IEEE Transactions on Knowledge Data Engineering, vol.[2] 15, no. 4, pp. 871–882, 2003.

[2] JCS Kadupitiya, Surangika Ranathunga, and Gihan Dias. "Sinhala short sentence similarity calculation using corpus-based and knowledge-based similarity measures". In Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSLLP 2016), pp. 44–53, 2016.

[3] Anutharsha Selvarasa, Nilasini Thirunavukkarasu, Niveathika Rajendran, Chinthoorie Yogalingam, Surangika Ranathunga, and Gihan Dias. "Short Tamil sentence similarity calculation using knowledge-based and corpus-based similarity measures." In Engineering Research Conference (MERCon), 2017 Moratuwa, pp. 443-448. IEEE, 2017.

[4] Li, Yuhua, Zuhair Bandar, David McLean, and James O'shea. "A Method for Measuring Sentence Similarity and its Application to Conversational Agents." In FLAIRS Conference, pp. 820-825. 2004.

[5] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics." IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 8, pp. 1138–1150, 2006.

[6] Zhao, J., Zhu, T. T., & Lan, M. Ecnu: "One stone two birds: Ensemble of heterogeneous measures for semantic relatedness and textual entailment". Proceedings of the SemEval, pp. 271-277, 2014.

[7] He, Hua, Kevin Gimpel, and Jimmy Lin. "Multi-perspective sentence similarity modeling with convolutional neural networks." In Proceedings of the 2015 conference on empirical methods in natural language processing, pp. 1576-1586. 2015.

[8] Kiros, Ryan, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. "Skip-thought vectors." arXiv preprint arXiv:1506.06726. 2015.

[9] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks". In Association for Computational Linguistics (ACL). pp 1556-1566. 2015.

[10] He, Hua, and Jimmy Lin. "Pairwise word interaction modeling with deep neural networks for semantic similarity measurement." In Proceedings of the 2016 conference of the north American chapter of the Association for Computational Linguistics: human language technologies, pp. 937-948. 2016.

[11] Xie, Jun, Bo Chen, Xinglong Gu, Fengmei Liang, and Xinying Xu. "Self-attention-based BiLSTM model for short text fine-grained sentiment classification." IEEE Access vol. 7: pp. 180558-180570. 2019.

[12] Jonas Mueller and Aditya Thyagarajan. "Siamese Recurrent Architectures for Learning Sentence Similarity". In Proceedings of the 30th AAAI Conference on

Artificial Intelligence, February 12--17, Phoenix, Arizona, USA. pp. 2786-2792. 2016.

[13] Chandrasekaran, Dhivya, and Vijay Mago. "Evolution of Semantic Similarity - A Survey." arXiv preprint arXiv:2004.13820. 2020.

[14] Gomaa, W. H. and Fahmy, A. A, "A Survey of Text Similarity Approaches". International Journal of Computer Applications 68(13), pp. 13-18, 2013.

[15] Allison, Lloyd, and Trevor I. Dix. "A bit-string longest-common-subsequence algorithm." Information Processing Letters 23, no. 5: pp. 305-310. 1986.

[16] Levenshtein, Vladimir I. "Binary codes capable of correcting deletions, insertions, and reversals." In Soviet physics doklady, vol. 10, no. 8, pp. 707-710. 1966.

[17] Hall, Patrick AV, and Geoff R. Dowling. "Approximate string matching." ACM computing surveys (CSUR) 12, no. 4: pp. 381-402. 1980.

[18] Ratcliff, John W., and David E. Metzener. "Pattern-matching-the gestalt approach." Dr Dobbs Journal, Issue 13, no. 7: pp. 46. 1988.

[19] Krause, Eugene F. "Taxicab geometry." The Mathematics Teacher 66, no. 8: pp. 695-706. 1973.

[20] Wegner, Peter. "A technique for counting ones in a binary computer." Communications of the ACM 3, no. 5: pp 322. 1960.

[21] Danielsson, Per-Erik. "Euclidean distance mapping." Computer Graphics and image processing 14, no. 3: pp. 227-248. 1980.

[22] Jaccard, Paul. "Étude comparative de la distribution florale dans une portion des Alpes et des Jura." Bull Soc Vaudoise Sci Nat 37: pp. 547-579. 1901.

[23] Dice, Lee R. "Measures of the amount of ecologic association between species." Ecology 26, no. 3: pp. 297-302. 1945.

[24] Burgess, Curt, Kay Livesay, and Kevin Lund. "Explorations in context space: Words, sentences, discourse." Discourse Processes 25, no. 2-3: pp. 211-257. 1998.

[25] Foltz, Peter W., Walter Kintsch, and Thomas K. Landauer. "The measurement of textual coherence with latent semantic analysis." Discourse processes 25, no. 2-3: pp. 285-307. 1998.

[26] Turney, Peter D. "Mining the web for synonyms: PMI-IR versus LSA on TOEFL." In European conference on machine learning, Springer, Berlin, Heidelberg, pp. 491-502. 2001.

[27] Mihalcea, Rada, Courtney Corley, and Carlo Strapparava. "Corpus-based and knowledge-based measures of text semantic similarity." In Proceedings of the American Association for Artificial Intelligence. (Boston, MA), vol. 6, no. 2006, pp. 775-780. 2006.

[28] Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38, no. 11, pp. 39-41, 1995.

[29] Rajendran, S., S. Arulmozhi, B. Kumara Shanmugam, S. Baskaran, and S. Thiagarajan. "Tamil wordnet." In Proceedings of the first international global WordNet conference. Mysore, vol. 152, pp. 271-274. 2002.

[30] L. Arulmozhi, PT. Pattabhi R. K. Rao, Sobha, "A Hybrid POS Tagger for a Relatively Free Word Order Language." Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages, pp. 79-85, 2006.

[31] Welgama, V., Herath, D. L., Liyanage, C., Udalamatta, N., Weerasinghe, R., & Jayawardana, T. "Towards a Sinhala wordnet". In Proceedings of the Conference on Human Language Technology for Development, 2011

[32] Wijesiri, Indeewari, Malaka Gallage, Buddhika Gunathilaka, Madhuranga Lakjeewa, Daya Wimalasuriya, Gihan Dias, Rohini Paranavithana, and Nisansa De Silva. "Building a wordnet for sinhala." In Proceedings of the Seventh Global WordNet Conference, pp. 100-108. 2014.

[33] Weerasinghe, Ruvan, Dulip Herath, and Viraj Welgama. "Corpus-based Sinhala lexicon." In Proceedings of the 7th Workshop on Asian Language Resources (ALR7), pp. 17-23. 2009.

[34] Resnik, Philip. "Using information content to evaluate semantic similarity in a taxonomy." In Proceedings of the 14th International Joint Conference on AI. arXiv preprint cmp-lg/9511007. 1995.

[35] Lin, Dekang. "An information-theoretic definition of similarity." In Proceedings of the International Conference on Machine Learning, vol. 98, no. 1998, pp. 296-304. 1998.

[36] Jiang, J and Conrath, D. "Semantic similarity based on corpus statistics and lexical taxonomy". In Proceedings of the International Conference on Research in Computational Linguistics. arXiv preprint cmp-lg/9709008. 1997.

[37] Leacock, C and Chodorow, M. "WordNet: An electronic lexical database". MIT Press, Chapter Combining local context and WordNet similarity for word sense identification, pp. 265–283. 1998.

[38] Wu, Zhibiao, and Martha Palmer. "Verb semantics and lexical selection." In Proceedings of the Annual Meeting Association for Computational Linguistics. arXiv preprint cmp-lg/9406033. 1994.

[39] Lesk, Michael. "Automatic sense disambiguation using machine readable dictionaries: how to tell a pinecone from an ice cream cone." In Proceedings of the 5th annual international conference on Systems documentation, pp. 24-26. 1986.

[40] Islam, Aminul, and Diana Inkpen. "Semantic text similarity using corpus-based word similarity and string similarity." ACM Transactions on Knowledge Discovery from Data (TKDD) 2, no. 2: pp. 1-25. 2008.

[41] Allison, Lloyd, and Trevor I. Dix. "A bit-string longest-common-subsequence algorithm." Information Processing Letters 23, no. 5: pp. 305-310. 1986.

[42] Pawar, Atish, and Vijay Mago. "Calculating the similarity between words and sentences using a lexical database and corpus statistics." IEEE Transactions on Knowledge and data Engineering. arXiv preprint arXiv:1802.05667. 2018.

[43] Lintean, Mihai C., and Vasile Rus. "Measuring Semantic Similarity in Short Texts through Greedy Pairing and Word Semantics." In Flairs conference, pp. 244-249. 2012.

[44] Ferreira, Rafael, Rafael Dueire Lins, Steven J. Simske, Fred Freitas, and Marcelo Riss. "Assessing sentence similarity through lexical, syntactic and semantic analysis." Computer Speech & Language 39: pp. 1-28. 2016.

[45] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation vol. 9, no. 8: pp. 1735-1780. 1997.

[46] Graves, Alex, and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." Neural networks vol. 18, no. 5-6 (2005): pp. 602-610. 2005.

[47] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078. 2014.

[48] Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. "Signature verification using a" siamese" time delay neural network." Advances in neural information processing systems: pp. 737-737. 1994.

[49] Xie, Niantao, Sujian Li, and Jinglin Zhao. "ERCNN: Enhanced Recurrent Convolutional Neural Networks for Learning Sentence Similarity." In 18th China National Conference on Chinese Computational Linguistics, pp. 119-130. 2019.

[50] Yin, Wenpeng, Hinrich Schütze, Bing Xiang, and Bowen Zhou. "ABCNN: Attention-based convolutional neural network for modeling sentence pairs." Transactions of the Association for Computational Linguistics 4: pp. 259-272. 2016.

[51] Liang, Zhiyao, and Jian Liu. "Sentence Similarity Measurement with Convolutional Neural Networks Using Semantic and Syntactic Features." Computers, Materials & Continua, vol.63, no.2: pp. 943-957. 2020.

[52] Chen, Qin, Qinmin Hu, Jimmy Xiangji Huang, and Liang He. "CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1. pp.9 2018.

[53] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing vol. 45, no. 11: pp. 2673-2681. 1997.

[54] Chen, Qian, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. "Enhanced lstm for natural language inference." arXiv:1609.06038. 2016.

[55] Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781. 2013a.

[56] Ichida, Alexandre Yukio, Felipe Meneguzzi, and Duncan D. Ruiz. "Measuring semantic similarity between sentences using a siamese neural network." In 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-7. IEEE, 2018.

[57] Ranasinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov. "Semantic textual similarity with Siamese neural networks." In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), pp. 1004-1011. 2019.

[58] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In Advances in neural information processing systems, pp. 3111-3119. 2013b.

[59] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.

[60] T. Kenter and M. de Rijke. "Short text similarity with word embeddings". In CIKM, ACM, pp 1411-1420. 2015.

[61] McCann, Bryan, James Bradbury, Caiming Xiong, and Richard Socher. "Learned in translation: Contextualized word vectors." In NIPS. arXiv preprint arXiv:1708.00107. 2017.

[62] Mikolov, Tomáš, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." In Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 746-751. 2013.

[63] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information". Transactions of the Association for Computational Linguistics 5 (2017), pp. 135-146. 2017.

[64] Ranashinghe, Tharindu, Constantin Orasan, and Ruslan Mitkov. "Enhancing unsupervised sentence similarity methods with deep contextualised word representations." In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), 2019.

[65] Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." In NAACL. 2018a. arXiv preprint arXiv:1802.05365. 2018.

[66] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." OpenAI blog vol. 1, no. 8: pp. 9. 2019

[67] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805. 2018.

[68] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. "Semi-supervised sequence modeling with cross-view training". In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 1914 - 1925. 2018.

[69] Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." In ACL. Association for Computational Linguistics. arXiv preprint arXiv:1801.06146. 2018.

[70] Marco, Marelli, Bentivogli Luisa, Baroni Marco, Bernardi Raffaella, Menini Stefano, and Zamparelli Roberto. "SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment." In Proc. SemEval, pp. 1-8. 2014.

[71] Marelli, Marco, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. "A SICK cure for the evaluation of compositional distributional semantic models." In Lrec, pp. 216-223. 2014.

[72] Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9, no. 11, pp. 2579-2605, 2008.

[73] Lakmal, Dimuthu, Surangika Ranathunga, Saman Peramuna, and Indu Herath. "Word embedding evaluation for sinhala." In Proceedings of the 12th Language Resources and Evaluation Conference, pp. 1874-1881. 2020.

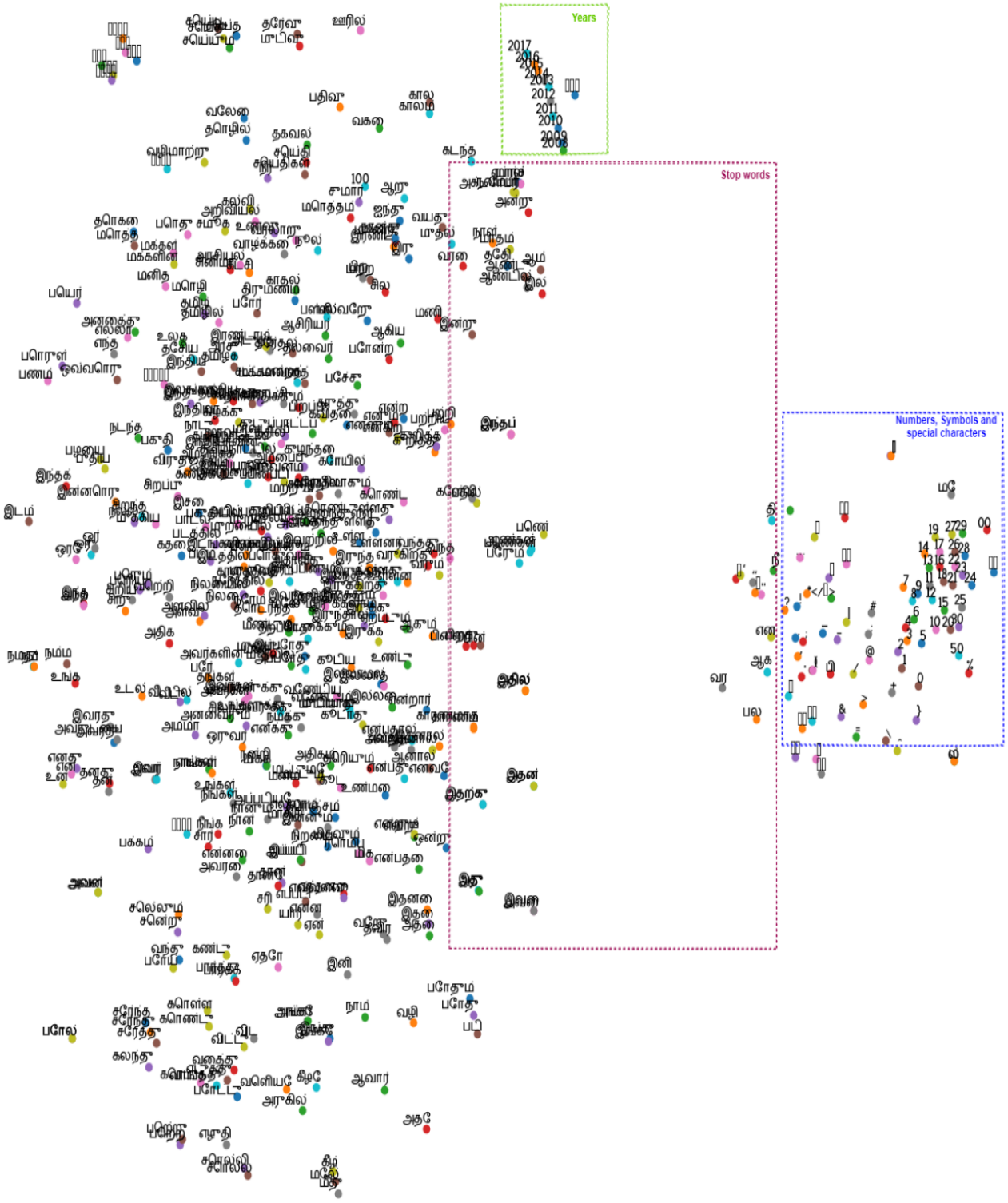
[74] "Tamil Stopwords list", TamilNLP, 2018. [Online]. Available: https://github.com/AshokR/TamilNLP/wiki/Stopwords [Accessed: Jul.26 2018].

[75] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, "Bag of Tricks for Efficient Text Classification". arXiv preprint 1607.01759, 2016.

[76] Neculoiu, Paul, Maarten Versteegh, and Mihai Rotaru. "Learning text similarity with siamese recurrent networks." In Proceedings of the 1st Workshop on Representation Learning for NLP, pp. 148-157. 2016.

APPENDIX-I

Visualization of Tamil word-embedding model using t-SNE algorithm



APPENDIX-II

Visualization of Sinhala word-embedding model using t-SNE algorithm