DEVELOPING A CODE-MIXED SENTIMENT ANALYSIS MODEL FOR XITSONGA-ENGLISH MUSIC REVIEW

by

BLESSING NKUNA

DISSERTATION

Submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

FACULTY OF SCIENCE AND AGRICULTURE

(School of Mathematical and Computer Sciences)

at the

UNIVERSITY OF LIMPOPO

Supervisor: Prof TI Modipa

Co-Supervisor: Mr PS Ramalepe

2025

DECLARATION OF AUTHORSHIP

I **Nkuna Blessing** declare that this study is my own original work and all information extracted from other sources is acknowledged as such by means of complete references. I further affirm that I have not submitted this work to any other university for any other degree or examination.

Signature:  Date: 10 March 2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to everyone whose support, guidance, and encouragement made the completion of this dissertation possible. Their support and guidance were invaluable throughout this journey.

My heartfelt gratitude goes to my family for their consistent encouragement and patience, which gave me the resilience to overcome challenges along this path.

I owe special thanks to my advisors, Dr. TI Modipa and Mr. PS Ramalepe, whose expertise, feedback, and unwavering support greatly influenced the quality and focus of this dissertation.

I extend my heartfelt gratitude to Mkateko Ndlovu, Altah Khosa, Hlamulo Chauke, and Ndzalo Ndumiso Ubisi for their invaluable help in annotating the data. Their contributions were instrumental in the successful completion of this research, and I am truly appreciative of their support throughout the journey.

Finally, I sincerely appreciate everyone who offered direct or indirect support throughout this journey. Your confidence in my abilities has been profoundly motivating.

ABSTRACT

Sentiment analysis is an essential natural language processing technique for monitoring online discussions about brands, products, and services. Traditionally focused on monolingual data, sentiment analysis has now expanded to include code-mixed texts, reflecting the growing use of multiple languages within single sentences on social media. This dissertation addresses the gap in sentiment analysis for code-mixed data by developing a Long Short-Term Memory (LSTM) classifier for Xitsonga-English comments extracted from YouTube music reviews. This research aims to design and implement a sentiment analysis model tailored for Xitsonga-English code-mixed texts, evaluating its performance against traditional monolingual sentiment analysis methods. This includes collecting a substantial dataset of Xitsonga-English comments, determining their polarity, developing an LSTM classifier, and assessing its accuracy, precision, recall, and F1-score. Data collection involved scraping 1 998 Xitsonga-English comments from a Xitsonga YouTube channel, cleaning and tokenizing the comments for analysis.

Sentiments were defined and categorized into positive, negative, and neutral classes based on specific criteria, with dictionaries developed for both Xitsonga and English lexicons. These lexicons were used to label the comments, facilitating the creation of training data for the LSTM model. Additionally, a word embedding matrix was developed using Word2Vec, capturing semantic similarities between words. The LSTM classifier's architecture included embedding layers initialized with pre-trained word embeddings, two LSTM layers for sequence processing, and a dense output layer for sentiment classification. Despite efforts to address overfitting through regularization and model adjustments, the final LSTM model did not perform as expected on the validation and test datasets, highlighting challenges in generalizing sentiment classification for the collected dataset. To address this, a stacking classifier combining Random Forest, Support Vector Machine, Gradient Boosting, and Logistic Regression was developed and compared with the LSTM model. The stacking classifier showed better generalization on unseen data, indicating its robustness for sentiment analysis tasks in code-mixed contexts.

The results highlight the challenges and potential solutions in developing robust sentiment analysis models for code-mixed languages, contributing valuable insights to the domain of natural language processing.

Keywords – Sentiment Analysis, Code-Mixed, Polarity, Annotated, Long short-term memory Classifier, Stacking Classifier.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1. Problem statement

Sentiment analysis, a technique within natural language processing (NLP), is used to track and interpret online discussions about an organization's brand, products, and services by extracting meaningful insights from the source content (Chakravarthi *et al.,* 2021). Traditionally, Businesses and institutions leveraged sentiment analysis to assess and understand the emotions expressed in product reviews. Sentiment analysis has gradually expanded to include the analysis of social media content. (Mäntylä *et al.,* 2018). Sentiment analysis can be applied to both monolingual and code-mixed reviews.

A sentence that incorporates words or phrases from two distinct languages is called a code-mixed sentence. The use of multiple languages within a single sentence or post has become a widespread practice among social media users. While many existing studies have focused on analysing sentiments from monolingual data, there is less research on sentiment analysis of mixed-language text data (Srinivasan and Subalalitha, 2021). Monolingual refers to the use of only one language in discourse.

Code-mixing introduces unique challenges for sentiment analysis. For example, a sentence like "This song is lit, ndza ku rhandza mhani" (Xitsonga-English: "This song is lit, I love you mom") combines English and Xitsonga in a way that creates ambiguities in syntax, grammar, and semantics. Monolingual sentiment analysis models, which are typically trained on clean, single-language datasets, struggle to interpret such hybrid text. This highlights the need for specialized models capable of understanding and processing code-mixed data.

Dutta et al. (2021) noted that messages, reviews, or tweets containing multiple languages, such as English-Hindi, English-Chinese, English-Tamil, and Xitsonga-English, have fewer sentiment analysis models developed for them. They further highlighted the limited exploration of code-mixed sentiment analysis, which is mainly attributed to the absence of

adequately annotated datasets. This is particularly true for low-resource languages like Xitsonga, where the lack of datasets and models has left a significant gap in research.

Few studies have developed a long short-term memory (LSTM) classifier for code-mixed sentiment analysis on South African languages. This study proposes an LSTM classifier for Xitsonga-English code-mixed music comments collected from YouTube reviews. The LSTM architecture was chosen due to its ability to retain long-term 'memory,' enabling it to understand longer contexts. By focusing on Xitsonga-English code-mixed data, this study addresses a critical gap in sentiment analysis research for low-resource languages and provides a foundation for future work in this area.

2. Motivation

The prevalence of code-mixing in social media platforms has increased substantially, making sentiment analysis classifiers for code-mixed data increasingly necessary. While sentiment analysis models for monolingual data have existed for some time, applying them to code-mixed data can be challenging (Chakravarthi, Priyadharshini, *et al.,* 2020). Mabokela and Schlippe (2022) introduced the first SAfriSenti corpus, which includes English, Sepedi, and Setswana. The corpus consists of over forty thousand tweets, with 36.6% being code-mixed. In this study, sentiment lexicons for Sepedi and Setswana were developed and incorporated into sentiment taggers using morphemes to indicate sentiment categories. However, the research did not involve the development of models for training or testing the dataset, nor did it include an evaluation of model performance, as no models were developed.

Muhammad *et al.* (2023) proposed a sentiment analysis model for Xitsonga-English code-mixed data. Their model was developed for Xitsonga that is spoken in Mozambique, which differs from South African Xitsonga. Additionally, they created a code-mixed sentiment analysis corpus encompassing 13 other African languages, including Amharic, Algerian Arabic, Hausa, Igbo, Kinyarwanda, Moroccan Arabic, Mozambican Portuguese, Nigerian Pidgin, Oromo, Swahili, Tigrinya, Twi, and Yorùbá. The dataset was collected using the Twitter Academic API. The study's results demonstrated that AfroXLMR-large outperformed AfroXLMR-base, XLM-R-large, and XLM-T-base by more than 2.5 points in

F1-score. However, their work focused on Mozambican Xitsonga, which differs from South African Xitsonga in vocabulary and usage. This highlights the need for region-specific models and datasets tailored to South African Xitsonga.

The most popular approaches for sentiment analysis include Convolutional Neural Networks (CNNs), Bidirectional Gated Recurrent Units (BiGRUs), and Long Short-Term Memory (LSTM) networks. CNNs are deep learning algorithms that perform automatic feature extraction without requiring explicit feature engineering (Tembhurne and Diwan, 2021). BiGRU is a sequence processing method that utilizes two GRUs, with one analysing the input from start to finish and the other from finish to start. LSTM, a complex sub-field of deep learning, aims to emulate human brain function and uncover hidden connections in sequential data (Gers *et al.,* 2003).

LSTM was selected for this research because of its ability to effectively retain and leverage memory. Long Short-Term Memory Networks (LSTMs) are capable of autonomously learning important features from raw input data, making them especially useful for processing complex and high-dimensional datasets. They can uncover meaningful representations, and extract features most relevant to the task at hand (Smagulova and James, 2019). In contrast, traditional machine learning models often rely on manual feature engineering, which is both time-consuming and may fail to capture all pertinent information in the data. Furthermore, the LSTM algorithm demonstrates strong generalization capabilities, enabling it to perform well on slightly different data from the training data. This versatility makes LSTMs especially suitable for real-world scenarios where sentiment analysis must be applied to diverse text inputs.

2.1 Aim

The aim of this study was to develop a code-mixed sentiment analysis model for Xitsonga-English music reviews using long-term short memory.

2.2 Objectives

The objectives of this study are to:

i. Collect Xitsonga-English code-mixed music reviews.
ii. Determine the polarity of Xitsonga-English code-mixed comments.

iii. Develop a sentiment classifier using long-term short memory.
iv. Evaluate the performance of the classifier.

2.3 Research question

The study will attempt to answer the following questions:

i. How to collect and analyse code-mixed music comments?
ii. How to develop a sentiment classifier using a deep learning algorithm?
iii. What are the effective methods for evaluating sentiment analysis classifier?

2.4 Research gap

This study contributed to the field of sentiment analysis by addressing the following gaps:

i. There was a scarcity of annotated datasets for Xitsonga-English code-mixed text, particularly in the context of music reviews. This study addressed this gap by collecting and annotating a dataset of Xitsonga-English code-mixed music reviews from YouTube, providing a valuable resource for future research.
ii. Existing sentiment analysis models were primarily designed for high-resource languages, leaving low-resource languages like Xitsonga underexplored. By developing an LSTM-based sentiment classifier for Xitsonga-English code-mixed data, this study demonstrated the feasibility of applying advanced deep learning techniques to low-resource languages.
iii. While some studies had explored Xitsonga-English code-mixing, they focused on Mozambican Xitsonga, which differed from South African Xitsonga. This study focused on South African Xitsonga, providing a region-specific model. This focus ensured that the model was tailored to the linguistic and cultural nuances of South African Xitsonga, making it more relevant and effective for local applications.

3. Scientific contribution

The LSTM classifier that the study developed is the first one developed on South African indigenous languages specifically on Xitsonga-English code-mixed data. This classifier is used to classify new Xitsonga-English comments' sentiments without going through each comment to check their sentiments. The study contributes to the development of sentiment

analysis classifiers using South African code-mixed dataset. The study produced Xitsonga-English code-mixed dataset for future sentiment analysis studies.

4. Dissertation outline

- Chapter 2: Literature Review – An extensive examination of previous studies on sentiment analysis, code-mixed data, Xitsonga-English context, classification models, and data augmentation techniques.
- Chapter 3: Methodology – An explanation of the research design, including data collection, cleaning, pre-processing, sentiment analysis, model development, and data augmentation.
- Chapter 4: Results – An in-depth analysis of the performance of the classifiers, highlighting misclassification patterns and examining the effects of data augmentation on model accuracy.
- Chapter 5: Conclusion – Summary of findings and future research.

Chapter 2: Literature Review

2.1 Introduction

This section reviews research that has advanced the creation of sentiment analysis models for code-mixed datasets. Individuals often use one or more languages when communicating or commenting on social media posts. Conducting sentiment analysis on code-mixed datasets facilitates the examination of product reviews, advertisements, and social trends (Choudhary *et al.,* 2018). The widespread use of multiple languages on social media platforms has sparked growing attention towards the research of sentiment analysis for code-mixed datasets (Ahmad *et al.,* 2022).

Xitsonga, one of South Africa's 12 official languages, is spoken by approximately 2.2 million people (Alexander, 2023). This language is distributed across a broad area in the southeast of Southern Africa, including Zimbabwe and southern Mozambique, where it is also commonly known as Xichangana. In daily conversations or when expressing their views, Xitsonga speakers frequently mix their language with English (Zerbian, 2009).

2.2 Cultural significance of Xitsonga

Xitsonga is part of the Bantu language family, which belongs to the larger Niger-Congo group. It has different dialects like Tswa and Ronga, which are very similar and easy to understand for Xitsonga speakers (Mabaso, 2014). Xitsonga has many vowel and consonant sounds, like other Bantu languages, giving it a unique sound (Ziervogel et al., 1971). For the Tsonga people, Xitsonga is an important part of their identity and culture. The language is used in many cultural activities like music, dance, and storytelling, and it plays a key role in traditional ceremonies and rituals (Mtetwa, 2005). These practices help keep the language alive in the community. One interesting aspect of Xitsonga is the mixing of languages, especially with English. Many people switch between Xitsonga and English in their conversations, showing how multilingual societies adapt (Setati, 2002). This helps people communicate more easily in different situations.

However, fewer people speak it as their first language, and there aren't enough resources for teaching it. Despite this, there are efforts through education and cultural programs to promote and preserve Xitsonga so that it continues to thrive (Nkondo, 2013).

2.3 Monolingual and code-mixed language processing in sentiment analysis

Understanding both monolingual and code-mixed language processing is essential for advancing sentiment analysis in multilingual and linguistically diverse contexts. The following sections provide insights into how sentiment analysis is conducted in both monolingual and code-mixed language contexts.

2.3.1 Monolingual language processing in sentiment analysis

Monolingual data consists of text or content written in a single language and is frequently employed to train and develop NLP models tailored to that specific language (Wenzek *et al.,* 2019). Monolingual sentiment analysis involves extracting emotions or sentiments from data written in a single language. Research in this area has been more prevalent than in code-mixed sentiment analysis, with most studies concentrating on English texts (Singh and Lefever, 2020).

2.3.2 Code-mixed language processing in sentiment analysis

Code-mixed text involves the blending of two or more languages within a single sentence or discourse (Woon and Ho, 2007). Fewer studies have explored code-mixed sentiment analysis, as much of the research has focused on the English language. This limited focus is largely attributed to the lack of suitable code-mixed datasets (Srinivasan and Subalalitha, 2021).

2.4 Annotation process for code-mixed sentiment analysis data

Using YouTube Comment Scraper, Hande *et al.* (2020) gathered code-mixed English-Kannada comments from YouTube. Following Chakravarthi *et al.* (2020), at least three annotators annotated each sentence using the following schema: Mixed-feelings, neutral, positive, negative, and not in the intended language. Annotators' annotations were gathered via a Google form. Gender and educational background data were gathered in

order to determine the diversity of the annotators. Next, guidelines for annotating the comments were provided to them.

Ameer et al. (2022) manually selected data for English and Roman Urdu code-mixed SMS messages from the existing benchmark SMS-AP-18 corpus. To streamline the annotation process, 12,000 code-mixed SMS messages were carefully chosen from a total of 84,694 messages. The proposed corpus was annotated manually by three annotators following predefined guidelines. Each annotator was a native Urdu speaker fluent in English, with formal education and extensive experience in text annotation.

Chakravarthi *et al.* (2020) utilized a YouTube scraper to gather code-mixed Malayalam-English comments from YouTube in order to create an annotated corpus. They followed (Mohammad, 2016) methodology for annotation, having at least three annotators identify each sentence using one of the following categories: Positive, Negative, Mixed-Feelings, Neutral, and Not in Intended Language. They used Google forms with a limit of 100 sentences per form, and they collected the annotator's email so they could only annotate once. Two annotators annotated each sentence; a third annotator annotated the sentence in the event of a controversy. Two more annotators annotated the sentences if the three annotators were at odds.

Using data collected from Twitter, Mandal et al. (2018) developed a Bengali-English Code-Mixed Corpus for Sentiment Analysis. They implemented two primary data annotation systems: one for sentiment tagging and another for language tagging. The language tagging system was built using a curated list of Bengali words derived from their study, alongside a collection of English words sourced from open internet resources. The aforementioned lists of English and Bengali words were used to create the N-Grams-Bi-Grams and Tri-Grams dictionaries at the character level. The Linear Support Vector Machine (LSVM) was trained to create a supervised language tagger. The language tagging algorithm looks up the target token in the lexicons first and then assigns the relevant tag if it is found. Given that the target token is not found in the lexicon, then the tag will be assigned using the supervised tagger. In order to determine the sentiments of a tweet, they employed a rule-based technique known as sentiment tagging. They examined the three items: Emoticon, Feeling, and Hashtags. Yimam *et al.* (2020)

developed a code-mixed corpus of tweets in Amharic and English that were gathered via the Twitter API. Three individuals annotated the collected tweets.

2.5 Levels of granularity in sentiment analysis

Sentiment analysis can be conducted at various levels, offering different degrees of granularity to derive meaningful insights from textual data. Each level examines a distinct scope, ranging from analysing an entire document to focusing on specific entities or aspects within the text. This section explores the four most commonly used levels of sentiment analysis: document-level, sentence-level, aspect-level, and entity-level, each providing a unique perspective for understanding sentiments in text.

2.5.1 Document-level sentiment analysis

The objective of document-level sentiment classification is to assess a user's overall opinion about a product (Dou, 2017). At this level, the task involves classifying whether the sentiment expressed in an entire document is positive or negative (Joshi and Itkat, 2014). However, as noted by Yessenalina *et al.* (2010), a significant challenge in document-level sentiment classification lies in the fact that certain sections of a document have a greater influence on identifying its general sentiment than others.

2.5.2 Sentence-level sentiment analysis

Sentence-level sentiment classification focuses on identifying subjective sentences within a document that contains both objective and subjective content. Once subjective sentences are detected, their sentiment orientation is determined (Kumar and Sebastian, 2012). This approach analyses each sentence individually to ascertain its polarity, providing a sentiment polarity for each statement (Shirsat *et al*., 2018). Sentence-level sentiment analysis has attracted growing interest because of its complexity and its essential role in facilitating opinion analysis tasks (Yang and Cardie, 2014).

2.5.3 Aspect-level sentiment analysis

Aspect-level sentiment analysis begins with identifying pieces of text that represent features of a product (Kolkur *et al.,* 2015). This approach involves first detecting the aspects of the target entity and then assigning a sentiment polarity to each aspect (Steinberger *et al.,* 2014). Aspect-level analysis typically concentrates on the development of different types of deep learning models to enhance performance (Zhang and Qian, 2020). As a fine-grained sentiment analysis task, it provides detailed and comprehensive results (Wang *et al.,* 2016).

2.5.4 Entity-level sentiment analysis

Entity-level analysis focuses on identifying the sentiment linked to named entities within a given text input, which are typically application-dependent (Fu *et al.,* 2019). Feature extraction at this level often utilizes the SentiWordNet lexicon, applied to parsed text to determine the polarity of descriptive words (Sweeney, 2017). This approach is particularly useful for applications requiring detailed sentiment insights, such as analysing product reviews or customer feedback (Cambria *et al.,* 2017).

For this study, sentence-level sentiment analysis was chosen. This method was selected because the analysis concentrates on identifying the sentiment polarity of each sentence in the dataset. By treating each sentence as an independent unit, the approach ensures that sentiments are accurately captured without being influenced by the context of surrounding sentences. This level of granularity aligns with the study's objective of analysing user comments, which are often brief and self-contained.

2.6 Code-mixed sentiment analysis approaches

Sentiment analysis for code-mixed datasets can be performed using the following three primary approaches: Lexical-based**,** Machine-learning**,** and Neural networks**.** Each approach is effective in different contexts and can be applied to various levels of textual analysis such as document, phrase, and aspect levels. This subsection provides a detailed comparison of the methodologies used in prior studies, including feature extraction techniques, machine learning algorithms, and performance metrics. Table 2.1 summarizes the findings from the literature:

Table 2.1 Summary of code-mixed sentiment analysis approaches

Study	Dataset Source	Language Pair	Feature Extraction	Algorithms Used	Performance Metrics	Key Findings
Medhat et al. (2014)	Twitter, Facebook, WhatsApp	English-Punjabi	5-gram and trigram algorithms	Custom n-gram-based classifiers	Accuracy, Precision, Recall, F1-score	Trigram technique outperformed 5-gram in accuracy and F1-score; both performed equally in precision and recall.
Garain et al. (2020)	Twitter	Hindi-English	Text preprocessing, tokenization	Support Vector Regression (SVR)	Recall, Accuracy, F1-score	SVR performed best on positive tweets based on recall, accuracy, and F1-score.
Kanwar et al. (2020)	YouTube comments	Malayalam-English, Tamil-English	Text preprocessing, tokenization	Logistic Regression	Precision, Recall, F1-score	Logistic Regression performed well in classifying positive comments for both languages based on precision, recall, and F1-score.
Smagulova and James (2019)	Not specified	Xitsonga-English	LSTM embedding and encoding layers	LSTM Network	Recall, Accuracy, Precision, F1-score	LSTM outperformed non-neural network methods by capturing the true meaning of input strings.
Tho et al. (2021)	Twitter API	Javanese-Indonesian	Lexicon-based features (SentiNetWord, VADER)	Lexicon-based models (SentiNetWord, VADER)	Accuracy, Precision, Recall, F1-score	Both VADER and SentiNetWord performed well in identifying negative tweets but struggled with positive and neutral tweets.
Chakravarthi et al. (2020)	YouTube comments	Tamil-English	Text preprocessing, tokenization	BERT-Multilingual, Naive Bayes, Decision Tree, Random Forest, SVM, 1DConv-LSTM	F1-score, Precision, Recall	K-Nearest Neighbor and 1DConv-LSTM excelled in classifying positive comments; Naive Bayes performed best for negative comments. Random Forest and Logistic Regression had high F1-scores for positive comments.
Sultan et al. (2020)	Patwa et al. (2020) dataset	Spanish-English	TF-IDF (word + char level)	SVM, Multinomial Naive Bayes (MNB), Logistic Regression (LR), XLM-RoBERTa	F1-score	XLM-RoBERTa achieved the highest F1-score.
Balouchzahi et al. (2021)	Dravidian-CodeMix-FIRE2021	Kannada-English, Malayalam-English, Tamil-English	Char n-grams, syllable n-grams, TF-IDF vectorization	Logistic Regression (LR), Multi-Layer Perceptron (MLP), Linear SVM (LSVM)	F1-score	LR and LSVM performed best for different feature sets and language pairs. Char n-grams contributed most to high performance.
Bharathi and Samyuktha (2021)	Dravidian-CodeMix-FIRE2021	Malayalam-English, Kannada-English,	Count vectorization, TF-IDF, multilingual	Logistic Regression (LR), Multilayer Perceptron (MLP), Naive	F1-score	Compared multiple classifiers; results varied based on feature extraction methods.

		Tamil-English	transformer embeddings	Bayes (NB), KNN, Random Forest (RF)		
Sabty et al. (2019)	Twitter, interviews, ANERCorp	Arabic-English	Word2Vec (skip-gram, CBOW), character-level embeddings	Deep neural networks	F1-score	Poor performance due to mismatch between monolingual training data and code-mixed test data.
Singh et al. (2018)	Twitter API	English-Hindi	Tokenization, POS tagging	LSTM, Conditional Random Field (CRF)	Accuracy	CRF outperformed LSTM for POS tagging.
Anbukkarasi (2020)	Dravidian-CodeMix-FIRE2020	Tamil-English	Text preprocessing, tokenization	Bi-LSTM	Precision, Recall, F1-score	Performed well on validation data but poorly on testing data.
Reddy et al. (2019)	Twitter API	Hindi-English	Word2Vec embeddings, character-level features	CNN, Bi-LSTM, attention-based Bi-LSTM	Accuracy	Attention-based Bi-LSTM outperformed CNN and standard Bi-LSTM for humour recognition.
Lal et al. (2019)	Prabhu et al. (2016) dataset	Hindi-English	Character-level representations, subword embeddings	CNN, dual Bi-LSTM, feature network	F1-score	Subword-LSTM outperformed Char-LSTM and traditional ML models like SVM and Naive Bayes.
Patra et al. (2018)	Twitter API	Hindi-English, Bengali-English	Glove embeddings, TF-IDF word n-grams	SVM, Naive Bayes, CNN, Bi-LSTM	F1-score, Precision, Recall	N-gram-based model achieved the highest F1-score.
Sabri et al. (2021)	Twitter API	Persian-English	Multilingual BERT embeddings, translation of non-Persian words	Ensemble Bi-LSTM, Naive Bayes, Random Forest	F1-score	Ensemble Bi-LSTM outperformed Naive Bayes and Random Forest.
Javdan et al. (2020)	Public datasets	Hindi-English, Spanish-English	Tokenization, n-grams	NBSVM (Naive Bayes-SVM), Bi-GRU-CNN, Bi-LSTM-CNN	F1-score	NBSVM outperformed other classifiers across both datasets.
Patwa et al. (2020)	Patra et al. (2018), Solorio et al. (2014)	Hindi-English, Spanish-English	Tokenization, embeddings	Naive Bayes, Logistic Regression, Random Forest, SVM, CNN, LSTM, Bi-LSTM, BERT	F1-score	CNN performed best for Hindi-English; BERT achieved superior results for Spanish-English.

2.6.1 Lexical based approach to sentiment analysis

A sentiment lexicon consists of a list of words that have been assigned a polarity and strength (Darwich *et al.*, 2019). One of the simplest classifiers for identifying sentiments is lexicon-based sentiment analysis, which assigns a score based on the comparison between the lemmatized tokens in the text under study and the lemmas in the lexicon (Ohman, 2021). The dictionary-based approach, which is based on dictionary words (i.e., WordNet), and the corpus-based approach, which employs corpus data, are the two categories of the lexicon-based approach. It may also be further subdivided into statistical and semantic approaches (Karachi, 2018). At the document, phrase, and aspect levels, the Lexicon-based method can be applied. The following are tools that can be used for lexicon-based sentiment analysis: SentiWord, VADER, SenticNet and TextBlob.

SentiWordNet

SentiWordNet is a lexical resource where each item is a "synset," or collection of lemma-PoS pairs with similar meanings (Esuli *et al.,* 2006). The numerical scores positive(s) and negative(s), which span from zero to one, are linked to each synset s (Guerini *et al.,* 2013). This scoring system enables SentiWordNet to effectively capture subtle nuances in sentiment, making it a valuable tool in various sentiment analysis applications (Baccianella et al., 2010).

VADER

VADER is used to determine the polarity of texts and classify them in multiclass sentiment analysis (Chiny *et al.,* 2021). It has proven particularly effective at handling social media messages, movie reviews, and product reviews by combining a sentiment lexicon with a list of lexical elements, each labelled as either positive or negative based on their semantic orientation (Bonta *et al.,* 2019). Overall, VADER's versatility in analysing diverse textual sources makes it an invaluable tool for sentiment analysis tasks in contemporary applications (Hutto and Gilbert, 2014).

TextBlob

TextBlob is a Python library that is used to process sentiments for textual data; it also provides scores for polarity and subjectivity of each word in the text, where polarity is used to identify the sentiment and subjectivity usually refers to personal opinions (Mas Diyasa *et al.*, 2021). This combination of features makes TextBlob a valuable tool for sentiment analysis, allowing users to easily extract insights from text data (Loria, 2018).

SenticNet

SenticNet is a lexicon approach that looks at opinion polarity and affective information where common sense knowledge concepts are excluded (Cambria *et al.,* 2012). By focusing on the emotional context of words, SenticNet provides a unique perspective on sentiment analysis, making it a valuable resource for understanding nuanced sentiments in text (Cambria et al., 2013).

Applications of sentiment analysis using lexicon-based approach

Tho *et al.* (2021) studied code-mixed sentiment analysis of Javanese and Indonesian languages using a Lexicon-based methodology. Their study's objective was to compare SentiNetWord and VADER, two lexicon models. To gather the code-mixed tweets, they used the Twitter API. When Vader and SentiNetWord models were compared with hand labeling, both performed well in recognizing negative tweets, but they were unable to produce the same outcomes for positive and neutral tweets.

2.6.2 Machine-learning approach to sentiment analysis

Machine learning algorithms, as described by Ain *et al.* (2017), focus on extracting phrases and aspect-level features. These algorithms encompass a range of classifiers, including Naive Bayes, Support Vector Machines, Random Forest, Decision Trees, K-Nearest Neighbors, and Logistic Regression, all of which are commonly used in sentiment analysis tasks (Joshi *et al.,* 2016).

Naive Bayes

Naive Bayes (NB) is an approach that relies on the Bayes Theorem, assumes feature independence, and is useful for spam detection and text categorization (Baid *et al.,* 2017). As a supervised learning and statistical approach for classification, the Bayesian Classification computes the probability for hypotheses and is resistant to noise in the input data (Learned-miller, 2011). The simplicity and efficacy of Naive Bayes make it the preferred method for sentiment analysis (Kolchyna *et al.*, 2015).

Support Vector Machine

Supervised models called support vector machines (SVM) are employed in data analysis, regression analysis, and text classification (Shanmugavadivel *et al.*, 2022). According to Usman *et al.* (2016), SVM is reliable on big feature spaces and efficient on text categorization tasks. Given these advantages, SVM has become a widely adopted method in various applications requiring effective classification and analysis of high-dimensional data (Zhang et al., 2019).

Random Forest

Random forests build multi-altitude decision trees during the input phase, and as an output, they produce a large number of decision trees (Gupte *et al.*, 2014). Using random samples of training data, Random Forests generate decision trees that improve performance, decrease overfitting, and decrease model variance overall (Nayak and Natarajan, 2016). Three essential elements comprise the Random Forest method: First, a prediction tree is created via bootstrapping sampling; next, each decision tree makes predictions using a random predictor; and last, predictions are combined by Random Forest using the average for regression or majority votes for classification to make predictions (Asian *et al.*, 2022). Sentiment analysis can benefit from Random Forest's various attributes (Khanvilkar and Vora, 2018).

Decision Trees

The core concept of this method, applicable to both regression and classification, involves constructing a tree incrementally while segmenting the dataset into progressively smaller subsets (Guia *et al.,* 2019). The information gain parameter and the Gini index are commonly used to determine which attribute should be selected to further partition the dataset (Ahuja *et al.,* 2019). Decision trees, as symbolic classifiers, offer two main advantages: first, they can reduce the number of features needed, and second, their structure provides an alternative summary of the patterns identified in the training data (Wakade *et al.,* 2012). Overall, decision trees are favored for their intuitive and interpretable nature, making them a widely used tool in various applications (Bühlmann & Yu, 2002).

K-Nearest Neighbors

The k-Nearest Neighbors (k-NN) method is often used as a classification algorithm due to its memory-based learning approach and its simplicity, which makes it easy to understand and apply at every stage (Damarta *et al.,* 2021). The k-NN algorithm classifies new data by referring to the 'K' nearest data points (Salma and Silfianti, 2021). Its simplicity and effectiveness contribute to its widespread use in various applications, positioning it as a favoured approach for handling various analytical tasks among practitioners (Zhang, 2016).

Logistic Regression

Logistic regression is employed to predict outcomes based on multiple independent variables. (Tyagi and Sharma, 2018). The process involves taking the logarithm of a collection of weighted features derived from the input, which are then linearly combined by the logistic regression classifier (Dara *et al.,* 2017). This method is widely recognized for its effectiveness in binary classification tasks and is commonly applied in various fields, including healthcare and finance, to predict probabilities of outcomes (Ali *et al*., 2019).

Chakravarthi et al. (2020) developed a Tamil-English code-mixed sentiment analysis dataset. They collected YouTube comments from movie trailers using a YouTube comment scraper program for their study. A variety of machine learning methods were used to categorize the sentiments of these code-mixed Tamil-English YouTube comments, providing a baseline for further research. The techniques included BERT-Multilingual, Naive Bayes, Decision Tree, Random Forest, K-Nearest Neighbor, Support Vector Machine, 1DConv-LSTM, DME, and CDME. Additionally, the performance of these models was evaluated using F1-score, Precision, and Recall. K-Nearest Neighbor and 1DConv-LSTM did excellent in categorizing positive remarks when precision was used, but Naive Bayes did best when classifying negative comments. Regarding neutral remarks, Random Forest fared better than the other classifiers. They found that while Decision Tree worked well in classifying both neutral and negative comments, Naive Bayes and Support Vector Machine fared well in classifying positive remarks. Random Forest and Logistic Regression used the F1-score to classify the positive comments as having a high score; Decision Tree performed well for both the neutral and negative remarks.

Sultan *et al.*, (2020) carried out sentiment analysis on Spanish-English code-mixed data using machine learning. They made use of a supplementary dataset from Patwa *et al.*, (2020). They fitted two TF-IDF vectorizers, one for word-level and the other for character-level after pre-processing their dataset. Following that process, the vectorizers were merged into a unified output, which was subsequently input into machine-learning models. Three distinct machine-learning models were tested: Support Vector Machine (SVM), Multinomial Naïve Bayes (MNB), and Logistic Regression (LR). The XLM-RoBERTa base model, which they optimized, had the greatest F1-Score.

Kannada-English, Malayalam-English, and Tamil-English code-mixed sentiment analysis was examined in another study by Balouchzahi et al. (2021). The secondary data used is from Dravidian-CodeMix-FIRE2021, which consists of user comments. Char sequences, Char and syllables were extracted as sub-word level features; the n-gram generator was used to accept a list of those char sequences as an input, and it then generated the

corresponding n-grams. These n-grams were then vectorized using TF-IDF vectorizer in order to train the classifiers. Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Linear Support Vector Machines (LSVM) are the machine-learning classifiers. Char + LSVM, Char seq. + LR, Syllable + LR for Tamil-English; Char + LR, Char seq. + LSVM, Syllable + LR for Malayalam-English; and Char + LSVM, Char seq. + LSVM, Syllable + LR for Kannada-English were the three classifiers combined with the three word characteristics. They evaluated their paired classifiers and found that for different feature sets and language pairs, LR and LSVM performed better than each other. They found that char n-grams, followed by syllable n-grams, were responsible for most of the high performances.

Using the identical data used by Balouchzahi et al. (2021) and provided by Dravidian-CodeMix-FIRE2021 for Malayalam, Kannada, and Tamil, Bharathi and Samyuktha, (2021) conducted a code-mixed sentiment analysis. The two steps of their system architecture were the classification phase and the feature extraction phase. Count vectorization, TF-IDF vectorization, multilingual transformer based, and other classifiers including logistic regression, multilayer perceptron, Naive Bayes, K-Nearest Neighbor, and Random Forest classifier were compared during the feature extraction step.

2.6.3 Neural networks approach to sentiment analysis

Neural network algorithms include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long-Short-Term Memory Networks (LSTMs), Word2vec and Gated Recurrent Units (GRUs),

Convolutional Neural Networks (CNNs)

According to Tembhurne and Diwan, (2021) CNNs are a type of deep learning algorithm that primarily carry out automatic feature extraction without the need for explicit feature engineering. CNN neurons process input, propagate it further, and take in information just like neurons in regular neural networks (Rani and Kumar, 2019). Using values from a CNN's buried layer as features rather than as a classifier may be the best method for achieving improved accuracy (Poria *et al.,* 2015).

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are specialized neural networks built to process and analyse sequences of data, where the length of the input can vary, thanks to their recurrent hidden state, where the activation depends on the previous time step (Wang *et al.,* 2016). Contemporary RNNs, utilizing memory cells, are capable of detecting connections between data points that are far apart in the input sequence (Nistor *et al*., 2021). These networks are particularly beneficial for tasks like natural language processing, time series forecasting, and other areas that deal with sequential or time-dependent data (Mikolov *et al.,* 2010).

Long-Short-Term Memory Networks (LSTMs)

Long Short-Term Memory (LSTM) is a specialized type of neural network designed to learn long-range dependencies in text sequences. It employs memory units, or gates, that control the flow of information throughout the network (Patel and Tiwari, 2019). LSTM is a variant of the Recurrent Neural Network (RNN) architecture, specifically developed to 'retain' values obtained previously for a designated period (Muhammad *et al.,* 2021). This distinctive capability enables LSTMs to effectively overcome challenges in sequence prediction tasks, making them particularly valuable in applications such as language modeling, machine translation, and speech recognition (Hochreiter and Schmidhuber, 1997).

Gated Recurrent Units (GRUs)

Gated Recurrent Unit (GRU) is a neural network approach featuring gating mechanisms that manage the flow of information within the unit, without depending on separately defined memory cells (Sachin *et al.,* 2020). This streamlined design enables GRUs to effectively model sequential data while reducing computational complexity, making them a preferred choice for applications such as natural language processing and time series forecasting (Chung *et al.,* 2014).

Sabty et al. (2019) introduced the first annotated code-mixed sentiment dataset for an Arabic-English corpus. They also developed word embeddings and deep neural networks specifically for Arabic-English code-mixed text. Data was collected from three primary sources:

1. Informal interviews transcribed into a speech corpus.
2. Twitter via the streaming API.
3. The Arabic ANERCorp dataset for Named Entity Recognition (NER), which involved translating part of an annotated Arabic NER dataset.

Given the multilingual nature of the corpus, two word embedding models were utilized to cover both languages. The Word2Vec (W2V) technique was employed to develop and save the Arabic word embedding model, which was then integrated into the NER system. W2V incorporated methods such as skip-gram, deep learning, and continuous Bag-of-Words. An independent dataset of Arabic newswire articles was used to train the W2V model. However, due to the mismatch between training data (monolingual) and testing data (code-mixed), the resulting systems performed poorly, achieving a low F1-score.

Singh et al. (2018) presented an original language-tagged and POS-tagged dataset of English-Hindi code-mixed tweets, alongside a POS-tagging model trained on the dataset. Tweets were collected using Twitter's streaming API and annotated by two bilingual speakers fluent in Hindi and English. LSTM Recurrent Neural Networks and Conditional Random Field (CRF) models were employed for POS tagging, with the CRF model demonstrating superior performance.

Anbukkarasi (2020) focused on sentiment analysis of YouTube comments written in Tamil-English code-mixed language. They gathered comments from the Dravidian-CodeMix-FIRE2020 dataset. Using a Bi-LSTM classifier, they categorized comments into five sentiment categories: positive, negative, neutral, mixed-feelings, and non-Tamil. While the classifier performed well on validation data, its performance was less satisfactory on testing data.

Reddy et al. (2019) conducted a study on humour recognition in Hindi-English code-mixed tweets. Using Twitter's API, they built a corpus to train multilingual word embeddings. Word2Vec was employed to train the bilingual embedding model by experimenting with parameters like embedding size, window length, and negative sampling. Three deep learning models were developed:

1. A CNN-based model with four parallel 1D convolutional layers for feature extraction.
2. A Bi-LSTM network for extracting compositional semantics from bilingual text.
3. A Bi-LSTM network with an attention mechanism to highlight significant words for humour detection.

The attention-based Bi-LSTM model outperformed the others in accuracy.

Lal et al. (2019) utilized a dataset provided by Prabhu et al. (2016) to perform sentiment analysis on Hindi-English code-mixed data. Their model comprised three key components:

1. A CNN to generate sub-word-level representations.
2. A dual encoder network with two Bi-LSTMs to capture sentence-level sentiment information.
3. A feature network based on surface features and monolingual sentence vector representations to enhance classification accuracy.

Given the challenges of using word embeddings for code-mixed data, they opted for character-level representations. Subword-LSTM outperformed both Char-LSTM and traditional machine-learning models such as Support Vector Machine, Multinomial Naïve Bayes, and Lexicon-Based Naïve Bayes.

2.6.4 Combining machine learning and neural networks for sentiment analysis

Applications of sentiment analysis using both neural network and machine-learning approach

Patra et al. (2018) conducted a sentiment analysis study on code-mixed Hindi-English and Bengali-English data collected from Twitter using the Twitter API. They developed baseline systems using Glove word embeddings, TF-IDF word n-grams, sentiment lexicon-based models, Support Vector Machine (SVM), Naive Bayes, Document Term Matrix, Convolutional Neural Networks (CNN), and Bi-LSTM. These systems were evaluated using F1-score, recall, and precision. The n-gram-based model achieved the highest F1-score, emphasizing the efficiency of deep learning models for various NLP applications.

Sabri *et al.* (2021) compiled a collection of 3,640 code-mixed Persian-English tweets annotated for sentiment polarity, collected via the Twitter API. Their approach involved identifying non-Persian words, translating them using Yandex, and generating embeddings with a pre-trained multilingual BERT model. An ensemble model comprising three Bi-LSTM networks was evaluated against Naive Bayes and Random Forest classifiers. The combined model surpassed the others in sentiment classification.

Javdan *et al*. (2020) performed sentiment analysis on Hindi-English and Spanish-English code-mixed datasets. They proposed machine learning (NBSVM: Naive Bayes-SVM) and deep learning models (Bi-GRU-CNN and Bi-LSTM-CNN). Results demonstrated that the NBSVM model outperformed other classifiers across both datasets.

Patwa *et al*. (2020) examined sentiment analysis on Hindi-English and Spanish-English texts. Hindi data was sourced from Patra *et al.* (2018), and Spanish data from Solorio *et al.* (2014). The study involved 61 teams for Hindi-English and 28 for Spanish-English, with classifiers including Naive Bayes, Logistic Regression, Random Forest, SVM, CNN, LSTM, Bi-LSTM, and BERT. CNN performed best for Hindi-English, while BERT achieved superior results for Spanish-English.

Jhanwar and Das (2018) analysed Hindi-English code-mixed sentiment using Facebook comments from a dataset by Joshi *et al.* (2016). They introduced a combined model that integrates word n-gram-based Multinomial Naive Bayes (MNB) and character-trigram-based LSTM. MNB excelled in handling rare keywords, while LSTM performed better for longer sentences, effectively capturing sequential patterns.

2.7 Sentiment Analysis Metrics

Sentiment analysis relies on various metrics to quantify and evaluate textual emotions. These metrics can be broadly categorized into sentiment quantification metrics (such as polarity, intensity, and granularity) and performance evaluation metrics (such as accuracy, precision, recall, and F1-score).

2.7.1 Sentiment Quantification Metrics

Sentiment quantification metrics are used to measure the nature and strength of emotions expressed in text. Among these, sentiment polarity, sentiment intensity, and classification granularity are the most commonly employed.

Sentiment polarity

Sentiment polarity refers to the direction of sentiment, typically classified as positive, negative, or neutral. It provides a straightforward measure of the overall sentiment expressed in a text. Research by Tian *et al*. (2018) demonstrates that polarity scores alone serve as a strong foundation for sentiment classification, effectively capturing the overall direction of sentiment without the added complexity of intensity variations.

Sentiment intensity

Sentiment intensity measures the degree of positivity or negativity in a text. While it aims to quantify the strength of sentiment, it can struggle with contextual nuances, making it less reliable in certain cases. Akhtar *et al.* (2020) highlighted the limitations of sentiment intensity-based tools, noting that their sensitivity to subtle sentiment shifts can introduce ambiguity.

Classification granularity

Classification granularity refers to the level of detail in sentiment classification, ranging from binary (positive/negative) to multi-class (e.g., very positive, positive, neutral,

negative, very negative). While finer granularity can provide more nuanced insights, it often increases complexity and may reduce interpretability.

Given these findings, sentiment polarity scores emerge as a more effective approach for sentiment classification, particularly in real-world applications where clear distinctions between positive, negative, and neutral sentiments are essential. While sentiment intensity and classification granularity offer additional insights, their limitations highlight the importance of polarity-based models in achieving robust and interpretable sentiment analysis.

2.7.2 Performance Evaluation Metrics

The performance of sentiment classification can be assessed using accuracy, precision, recall, F1-score and confusion matrix (Kharde and Sonawane, 2016).

Accuracy

Accuracy is a key metric in evaluating classification models, representing the percentage of cases correctly identified. It is derived by comparing the number of correctly identified positive and negative instances to the total number of cases, which encompasses both the correct (true positives and true negatives) and incorrect (false positives and false negatives) predictions (Ahuja *et al*., 2019). Understanding accuracy is essential, as it provides crucial information regarding the consistency and performance of classification models across different applications (Kumar *et al*., 2016). Its formula is given by:

$$\text{Accuracy} = \frac{(True\ Positive + True\ Negative)}{(True\ Positive + True\ Negative + False\ Positive + False\ Negative)}$$

Precision

Precision, a metric for evaluating the strength of a model's predictions, is expressed as the proportion of true positive predictions relative to the total number of samples identified

as positive. In a model with 100% precision, all predicted positive samples are truly positive (Wankhade, Rao, and Kulkarni, 2022). This metric is especially important in situations where false positives carry significant consequences, as it helps ensure the accuracy of positive predictions (Sokolova and Lapalme, 2009). Its equation is given by:

$$\text{Precision} = \frac{(True\ Positive)}{(True\ Positive + False\ Positive)}$$

Recall

Recall is calculated by dividing the number of correctly identified positive instances by the total number of actual positive instances in the dataset (Ribeiro *et al.,* 2016). This metric is critical for assessing model performance in scenarios where identifying all relevant instances is paramount, such as in medical diagnosis or fraud detection (Davis and Goadrich, 2006). Its equation is giving by:

$$\text{Recall} = \frac{(True\ Positive)}{(True\ Positive + False\ Negative)}$$

2.7.4 F1-Score in sentiment classification

The F1-Score is a combined metric that represents the balance between precision and recall by calculating their harmonic mean. It provides a single value to assess the model's effectiveness, particularly useful in situations where class distribution is skewed, this metric is essential for evaluating models where both false positives and false negatives have significant consequences (Saito and Rehmsmeier, 2015). It is calculated by:

$$\text{F1-Score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

Confusion matrix

This evaluation technique assesses correctness by leveraging statistical measures, including true positives, true negatives, false positives, and false negatives (Navin and Balaji, 2016). Examining these metrics offers crucial information about how well classification models are performing, enabling practitioners to make targeted improvements and adjustments across applications such as healthcare diagnostics and fraud detection (Powers, 2011).

Application of evaluation metrics in code-mixed sentiment analysis

Medhat *et al*. (2014) assessed a study they conducted utilizing F1-score, accuracy, precision, and recall in code-mixed English-Punjabi sentiment analysis. In order to categorize the code-mixed sentences they gathered from Twitter, Facebook, and WhatsApp, they created 5-gram and trigram algorithms. While both algorithms produced the same results for precision and recall, the trigram technique outperformed the 5-gram in terms of accuracy and F1-score.

A sentiment analysis study on tweets with mixed Hindi and English codes was conducted by Garain *et al.* (2020). They evaluated the Support Vector Regression algorithm that they developed using recall, accuracy, and F1-score. On the positive tweets, the three evaluation metrics that were employed fared the best.

Following the application of logistic regression to a code-mixed sentiment analysis study involving YouTube comments in Malayalam-English and Tamil-English, Kanwar *et al.* (2020) employed precision, recall, and F1-score to assess the generated classification. Taking into account all of the evaluation factors, their classifier did a good job of categorizing positive comments for both code-mixed languages.

The LSTM network is used in this study to develop a sentiment analysis classifier for Xitsonga-English code-mixed words due to its efficacy in aiding with knowledge memorization. When it comes to non-neural network classification methods, certain words are trained as distinct inputs and have no real meaning in a phrase. As a result, the

methods forecast classes based on statistics rather than meaning (Smagulova and James, 2019). When the right LSTM embedding and encoding layers are applied, the model can determine the true meaning contained in the input string and produce the most correct output class. The study uses recall, accuracy, precision, and F1-score for evaluation.

2.8 Conclusions

This chapter reviewed studies on sentiment analysis for code-mixed datasets, focusing on their challenges and opportunities. It began by discussing the cultural significance of Xitsonga in communication, identity, and multilingual settings. The chapter highlighted the differences between monolingual and code-mixed language processing, noting the limited research on code-mixed datasets because of the absence of labelled data. It also explored the different methods used to create high-quality code-mixed corpora in various languages. These insights set the stage for understanding sentiment analysis approaches and guide the study's focus on addressing gaps in code-mixed sentiment analysis, especially for underrepresented languages like Xitsonga.

Chapter 3: Methodology

3.1 Introduction

This section outlines the comprehensive approach adopted for this study, encompassing the stages from data collection to the development and analysis of sentiment analysis classifiers. We begin by describing the data collection process from a Xitsonga YouTube channel, followed by an overview of the necessary data cleaning and preparation steps for effective analysis. Next, we discuss the data pre-processing and tokenization techniques applied to the collected comments. Finally, we delve into the sentiment analysis of the comments, covering monolingual and code-mixed comments, along with a comparative analysis of both comment types.

The approach used to analyse sentiments is then discussed, with a focus on defining and categorizing sentiments as well as identifying key sentiment-defining words. The development of a specialized dictionary for Xitsonga-English sentiment analysis is introduced, detailing its scope, purpose, and the process of gathering sentimental words and phrases. The classification of sentiments is elaborated upon, including the user profile of annotators, classification instructions, and guidelines.

A discussion of the lexicon and data annotation is followed by an overview of data visualization techniques. The chapter concludes with the development of sentiment classifiers, beginning with the LSTM classifier, including the tools used, word embedding development, and the overall model architecture. The final section covers the stacked classifier and data augmentation techniques used to enhance the performance of sentiment classifiers. Figure 3.1 illustrates a process flow for sentiment analysis, starting with data collection and moving through stages such as data analysis, sentiment definition, and dictionary development. It also includes data annotation, visualization, and the development of classifiers. After developing the classifiers, they are trained and then evaluated to assess their performance.

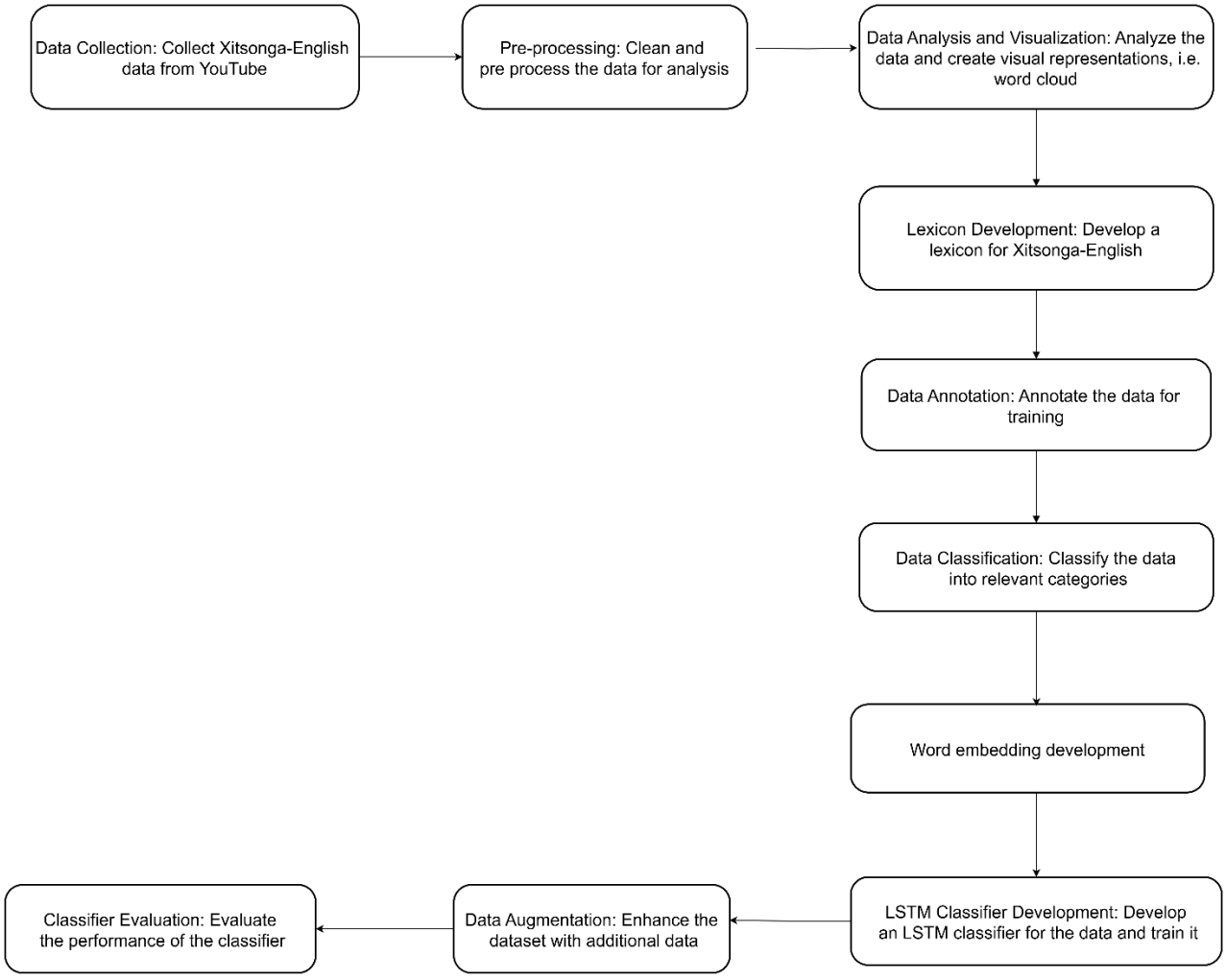

Figure 3.1 Detailed pipeline of the study's methodological process and workflow

3.2 Data collection from Xitsonga YouTube channel

The data for this study was collected from a Xitsonga YouTube channel that primarily features content related to Xitsonga songs, music festivals, and artist interviews. The data collection focused on the comments section of the channel. To ensure privacy, potential identifiers were removed from the dataset. Comments were collected utilizing a web scraping tool called YouTube Scraper. The tool extracted comments by inputting the URL of each music-related post. The data collection period ran from July 2023 to October 2023, resulting in a total of 989 Xitsonga-English code-mixed comments, 444 Xitsonga-only comments, and 565 English-only comments.

3.2.1 Data cleaning and preparation for analysis

After collecting the comments, the next step was data cleaning. This process encompassed multiple steps to standardize and prepare the data for effective use. These steps involved the removal of numbers, emojis, punctuation, and stop words, along with the conversion of uppercase text to lowercase for uniformity. The comments were then split into individual words or tokens, facilitating easier analysis of the text. By completing these steps, the data was properly prepared for more effective and accurate analysis in the following stages.

3.2.2 Data pre-processing and tokenization

Once the data was cleaned, the process moved to tokenization, carried out with the Python-based Natural Language Toolkit (NLTK). This process divided the text into manageable units known as tokens, which could represent individual words, phrases, or characters. Tokenization was a vital step in structuring the textual data, enabling effective analysis. The tokenization process included splitting the cleaned text into individual words, a key step for many natural language processing (NLP) tasks. Additionally, the text was broken down into individual sentences to enhance comprehension of the comments' context and structure. Throughout the tokenization process, the context of the words was preserved to avoid distorting phrases and idiomatic expressions. Table 3.1 shows the tokenized comments and the number of words each have, the words are coloured to show which ones are Xitsonga and which ones are English; the English ones are in yellow and the Xitsonga ones are in green. The comments presented by Table 3.1 were sampled from the collected comments, only the first 9 comments were taken.

Table 3.1 Tokenized comments with corresponding word count for each comment.

Comments	Number of words
made, miss, home, proud, say, tsonga	6
thank, loads, boti, helping, preserve, culture, keep, good, work, inkomu	10

kulelo, mina, everything, song, reminds, boy, hood, ka, bungeni	9
sesi, khombo, etlelani, hiku, rhula	5
maluleke, swa, saseka, swinene	4
ani, valavula, niku, minga, fambi, ntswelani, nhwe, mi, ala, mifamba, miti, sola, xisolo, xa, yini, nakha, na, cina	18
keep, good, work, going, support, till, end, time	8
talent	1
feel, like, reach, dead, end, song, always, make, realise, end, still, far, passed, many, challenges, life, lost, love, ones	19

3.3 Data analysis and visualisation of sentiment comments

In this study, we analysed a total of 1,998 comments comprising 21,554 words. These comments were categorized into two types: code-mixed and monolingual. The subsequent sections provide an in-depth analysis of both categories.

3.3.1 Analysis of monolingual comments

For Xitsonga comments, the dataset comprises 4,543 words, with sentence lengths varying from as few as 2 words to as many as 52 words. On average, Xitsonga sentences contain 10 words, with a standard deviation of 6.06. This indicates that while the average sentence length is relatively long, there is considerable variation in sentence length, suggesting diverse sentence structures and complexity within the Xitsonga comments.

The analysis of English comments reveals a total word count of 4,867, with sentence lengths varying from as few as 1 word to as many as 58 words. The average sentence length in English is 8 words, with a standard deviation of 5.98. When compared to the Xitsonga sentences, with an average sentence length of 10 words and a standard deviation of 6.0, the difference is minimal. This suggests that, while there is a slight

tendency for English sentences to be shorter, both languages exhibit similar variability in sentence lengths. This is illustrated in Table 3.2.

Table 3.2 The number of English and Xitsonga monolingual words per sentence

Language	**Number of words**	**Min words per sentence**	**Max words per sentence**	**Average words per sentence**	**Standard deviation**
Xitsonga	4543	2	52	10	6.06
English	4867	1	58	8	5.98

3.3.2 Analysis of code-mixed Xitsonga-English comments

For the code-mixed comments, Table 3.3 reveals how the two languages blend within sentences. The code-mixed comments have a total of 8,506 Xitsonga words. The shortest Xitsonga words in a sentence are just 2 words, but some can be as long as 80 words. On average, a sentence has 7 Xitsonga words, though there's a lot of variation, with a standard deviation of 8.10. For the English parts of the code-mixed sentences, there are 3,638 words. There are sentences with 2 English words, but the longest ones are only 34 words. On average, English sentences within this code-mixed context are shorter, with 4 words per sentence on average, and they show less variation in length, as the standard deviation is lower at 3.10. This means that in the code-mixed dataset, Xitsonga words tend to be longer and more varied, while the English portions are shorter and more consistent.

Table 3.3 The number of Xitsonga-English code-mixed sentences

Language	**Number of words**	**Min words per sentence**	**Max words per sentence**	**Average words per sentence**	**Standard deviation**
Xitsonga	8506	2	80	7	8.10
English	3638	2	34	4	3.10

3.3.3 Comparative analysis of monolingual and code-mixed comments

The visual analysis of the comments data highlights clear distinctions between monolingual and code-mixed comments. As illustrated in Figure 3.2, monolingual Xitsonga comments exhibit longer average sentence lengths compared to monolingual English comments. However, code-mixed comments reveal a notable shift: Xitsonga sentences are shorter on average, while English sentences are significantly shorter and more consistent. This trend suggests that users tend to favour Xitsonga over English when writing code-mixed Xitsonga-English comments. Figure 3.2 captures three key dimensions: (1) the average number of words per sentence for monolingual Xitsonga and English comments, as well as for each language within code-mixed comments; (2) sentence length variability across monolingual Xitsonga and English comments, alongside the individual contributions of both languages in code-mixed comments; and (3) the total word count for Xitsonga and English monolingual comments, as well as for each language within code-mixed comments.

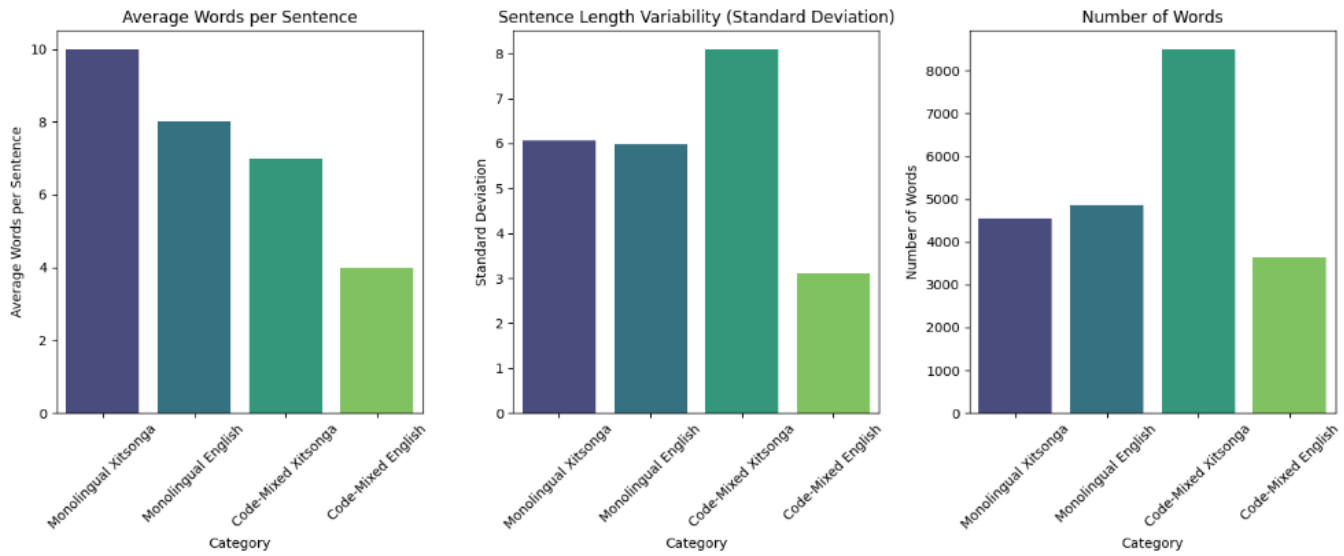

Figure 3.2 Comparative analysis of monolingual versus code-mixed comments

The bar graphs presented by Figure 3.3 illustrate the frequency distribution of sentence lengths for English, Xitsonga, and Xitsonga-English code-mixed. Each graph depicts the number of sentences with a specific length, measured in the number of words. For each dataset, the x-axis represents the sentence length, ranging dynamically from 0 to the maximum sentence length observed in the data: 80 words for Xitsonga-English code-mixed (Xitsonga part), 34 words for Xitsonga-English code-mixed (English part), 52 words

for Xitsonga monolingual, and 58 words for English monolingual sentences**;** while the y-axis shows the frequency of sentences corresponding to each length. Again, when looking at Figure 3.3, one can notice the positive skewness in the sentence length distribution. This positive skewness indicates that the majority of sentences are short, while a few significantly longer sentences extend the distribution's tail to the right. In a right-skewed distribution, the mean typically exceeds both the mode and the median, as the longer sentences pull the mean upwards. For the English monolingual comments, where most sentences are around 5-10 words (the mode), the mean is likely elevated to around 10-15 words due to the influence of the longer sentences, and the standard deviation reflects this variability, capturing the spread of sentence lengths. The Xitsonga monolingual comments show a less skew, with fewer long sentences, resulting in a mean that is closer to the mode (approximately 8-12 words) and a slightly lower standard deviation. In contrast, the Xitsonga-English code-mixed comments show the most significant right skew, characterized by a sharp peak at shorter sentences and a long tail of much longer ones. This leads to a higher mean, likely between 15-20 words, and the largest standard deviation among the three datasets, indicating that sentence lengths in the code-mixed dataset are the most variable. These visualizations provide a comparative overview of sentence length distributions, allowing for an easy comparison of text complexity and structure across the different languages.

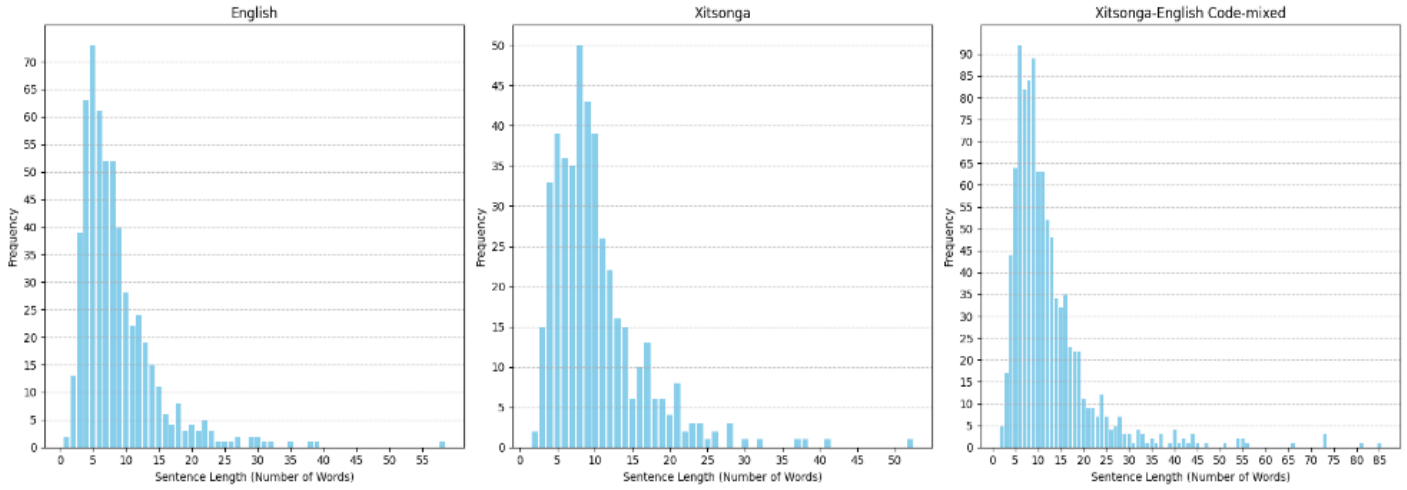

Figure 3.3 Frequency distribution of sentence lengths for English, Xitsonga monolingual and Xitsonga-English code-mixed.

3.3.4 Xitsonga-English word frequency analysis

In Figure 3.4, the dataset was visualized using a word cloud to emphasize the most common words in both Xitsonga and English. The most common words in the dataset, such as "song," "boti," "thank," "beautiful," "music," and "culture," show a strong connection to music and cultural appreciation. "Song" and "music" suggest that users enjoy and appreciate the music content. The word "boti," meaning "brother" in Xitsonga, could be referring to different artists that users are talking about. "Thank" shows that users are expressing gratitude for the content or experiences. "Beautiful" indicates that users find the content pleasing. Lastly, "culture" highlights the importance of cultural values, with many users expressing pride in their heritage. These words suggest that users are mostly sharing positive feelings about music, culture, and their experiences. The prominence of these words reflects the sentiments in the dataset, revealing what users find most important, impactful, or worth mentioning in their comments. Figure 3.5 presents the distribution of comments categorized into positive, negative, and neutral sentiments using a bar graph. The graph shows that there are 1004 positive comments, 705 neutral comments, and 289 negative comments. This distribution reflects predominantly positive comments, with a notable number of neutral comments and a smaller portion of negative feedback. The bar graph clearly illustrates the varying frequencies of each sentiment category within the dataset, reinforcing the observation of a generally favourable sentiment among users. In Figure 3.6, the pie chart visualizes the percentage distribution of the comments based on sentiment. Positive comments make up 50.3% of the total, neutral comments 35.3%, and negative comments 14.5%. The pie chart provides a straightforward visual representation of the sentiment distribution, highlighting that positive feedback is predominant. This distribution aligns with the word cloud and bar graph, showing that users largely view the subject positively, with a substantial portion of neutral feedback and a smaller amount of negative responses.

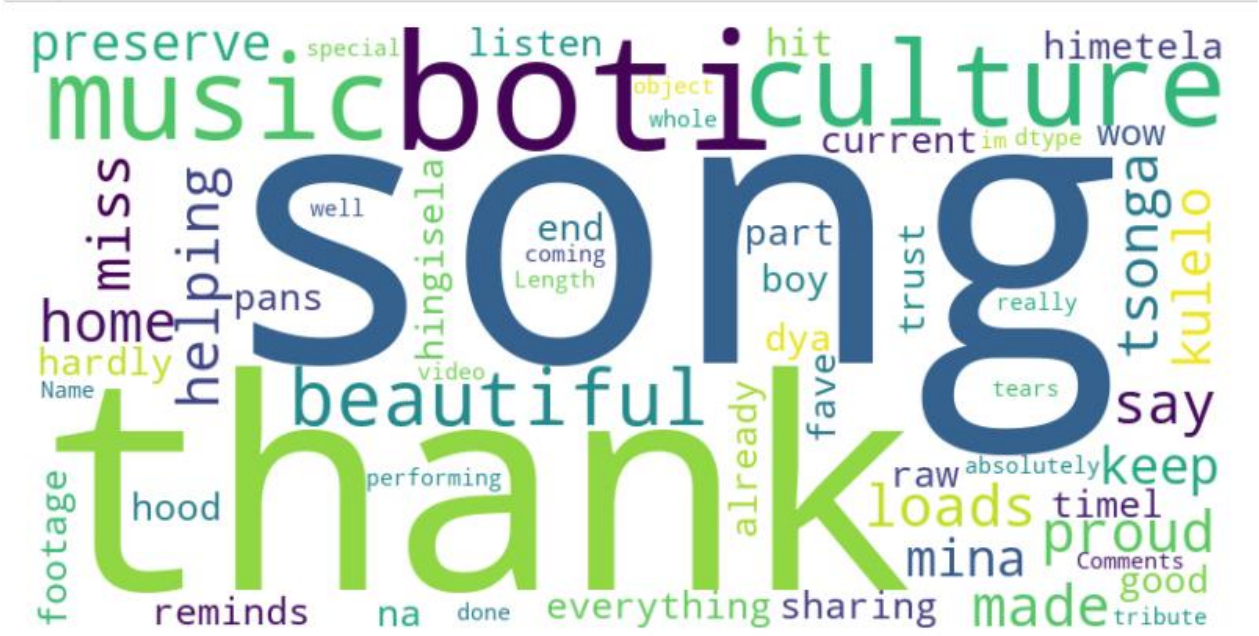

Figure 3.4 Xitsonga-English word cloud representing the most frequently used words

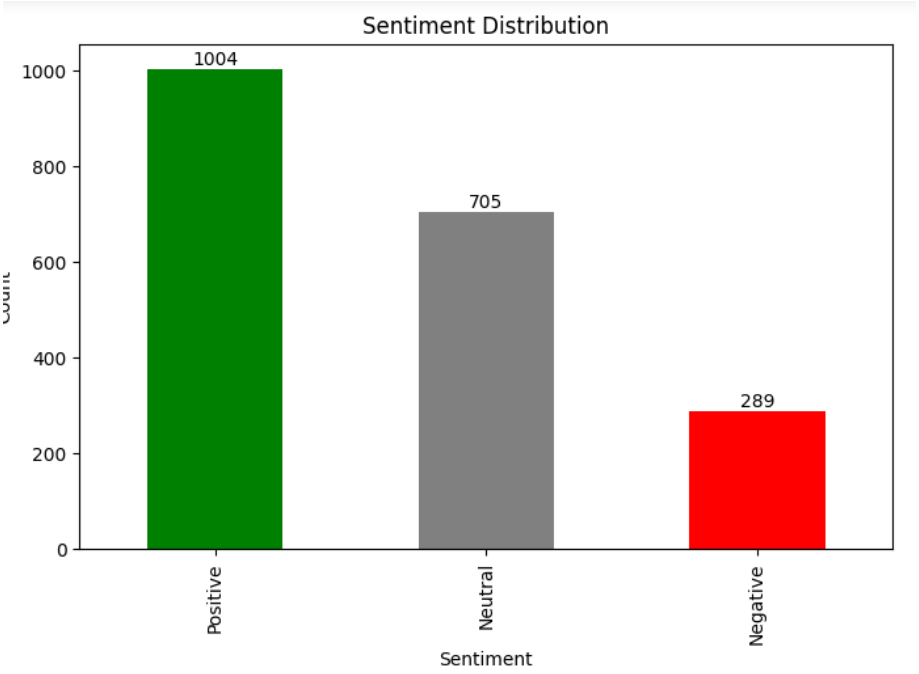

Figure 3.5 Bar graph illustrating sentiment distribution across comments

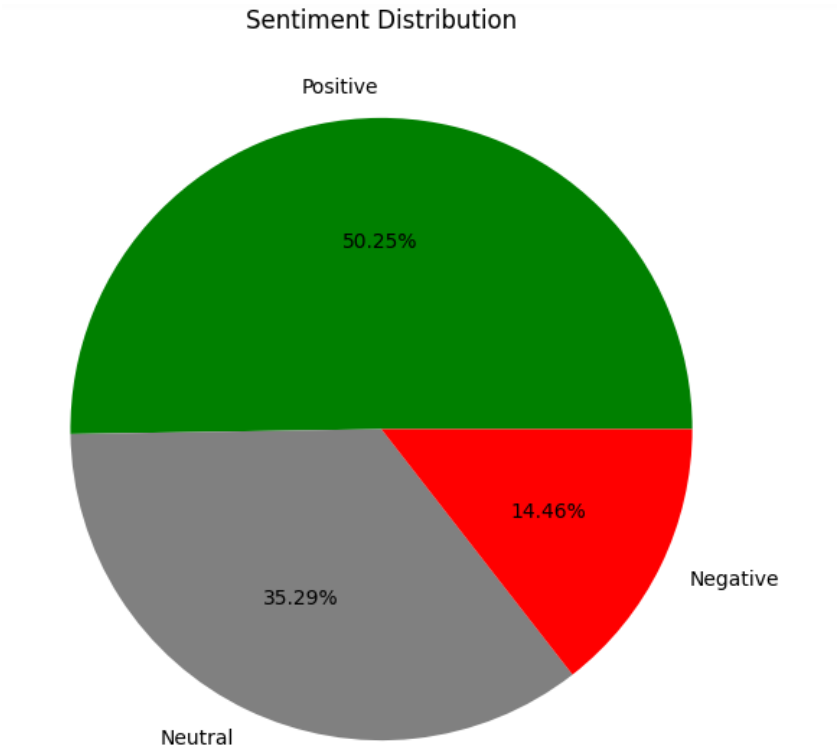

Figure 3.6 Pie chart illustrating sentiment distribution across comments

3.4 Analysing sentiments in the collected comments

This section discusses the sentiment definition as well as the words that defined the sentiments. The sentiments under consideration are positive, negative and neutral. The words used to define the sentiments are both in Xitsonga and English.

3.4.1 Defining and categorizing sentiments in the comments

In this section, we categorize comments into three main sentiment groups: positive, negative, and neutral. Each category is defined based on specific emotional and evaluative criteria. Positive comments encompass a range of uplifting and affirming sentiments, while negative comments include expressions of dissatisfaction and conflict. Neutral comments, on the other hand, are characterized by the absence of emotional or judgmental language, serving as impartial statements.

- Defining positive comments

We defined positive comments by using the following sentiments: (i) Joy and Happiness, (ii) Optimism; this includes good outcomes, improvements, and brighter futures, (iii)

Gratitude and Appreciation, (iv) Hope, (v) Kindness and Compassion, (vi) Achievement and Success, (vii) Praise and Compliments, (viii) Positivity in Language, (ix) Satisfaction.

- Defining negative comments

The negative comments were defined as following: (i) Sadness and Despair, (ii) Problems and Failures, (iii) Anger and Frustration, (iv) Criticism and Disapproval, (v) Conflicts and Tension, (vi) Regrets, (vii) Complaints, (viii) Challenges, (Iv) Negativity in Language.

- Defining neutral comments

The comments that did not add any emotional or judgemental language were classified as neutral.

3.4.2 Words that defines the sentiments

In the analysis of sentiments, it is crucial to identify and understand the words that convey various emotional tones. This section explores sentiment-laden vocabulary across two languages: English and Xitsonga.

- English words
  - ➢ Positive words: love, happy, excited, proud, amazing, and great.
  - ➢ Negative words: hate, sad, disappointed, hurt, sorry, and bad.
  - ➢ Neutral: song, album, video, performing, and tradition.

- Xitsonga words
  - ➢ Positive words: nyanyula (excite), rhanda (love), sasekile (beautiful), khensa (thank), nandika (nice), nyungubyisa (proud), nyikiwile (gifted), namba (good), tsakisa (exciting), tsokombela (sweet).
  - ➢ Negative: rila (cry), tika (tough), xengaku (deceive), mavabyi (sickness), nsele (cruel), dlaya (*kill*), sola (regret), chava (scared), xihlawuhlawu (discrimination).
  - ➢ Neutral: xidzumba (drum), tlambhi (artist), tshika (let go).

3.5 Developing a dictionary and lexicon for Xitsonga-English sentiment analysis and data annotation

The development of a specialized dictionary is a crucial component of this research, aimed at enhancing sentiment analysis capabilities for Xitsonga-English texts. This dictionary is designed to address the need for labelled data that can effectively train deep learning models. The primary goal is to create a valuable resource that bridges the linguistic gap between Xitsonga and English by providing a list of common words.

3.5.1 Scope and purpose of the specialized dictionary

- Goal: The purpose of creating this dictionary is to provide labelled data to train deep learning models for sentiment analysis tasks in Xitsonga-English texts.
- Target audience: This dictionary is intended for Xitsonga native speakers and individuals with an understanding of English.
- Common words: The dictionary contains a compilation of frequently used words in both Xitsonga and English, with their corresponding translations and sentiment labels. Table 3.4 and 3.5 show the commonly used words for English and Xitsonga.

Table 3.4 Part-of-speech labels for common English words

	English common words			
	Nouns	**Verbs**	**Adjectives**	**Adverbs**
1	Song	Dance	Good	Never
2	Music	Listen	Best	Much
3	Love	Play	Great	Unnecessarily
4	Shame	Perform	Proud	Really
5	Video	Watch	Pain	Absolutely
6	Artist	Thank	Nice	Lovely
7	Problem	Sing	Awesome	Very

Table 3.5 Part-of-speech labels for common Xitsonga words.

Xitsonga common words				
	Nouns	**Verbs**	**Adjectives**	**Adverbs**
1	Rhandza	Khensa	Fana	Ngofu
2	Risimu	Nyika	Nyungubyisa	Hakunene
3	Xikwembu	Tshika	Saseka	Henhla
4	Nyiko	Yimbelela	Nandziha	Kambe
5	Gaza	Kombela	Kahle	Hansi
6	Vaseketeri	Suka	Leswinene	Swinene
7	Vanga	Sukela	Biha	Switiva

3.5.2 Gather sentimental words and phrases

Collect vocabulary: After the process of tokenization is performed, sentimental words and phrases are collected. As it has been described when defining the sentiments, guidelines were used to decide which words are positive, negative or neutral. Table 3.6 presents a selection of sentimental words categorized as positive, neutral, and negative for both English and Xitsonga. It is important to note that these are not direct translations but rather different words that convey similar sentiments in each language. Table 3.7 shows sentimental phrases for Xitsonga while Table 3.8 is for English.

Table 3.6 Sentimental words categorized for both English and Xitsonga

	Positive		**Neutral**		**Negative**	
	English	**Xitsonga**	**English**	**Xitsonga**	**English**	**Xitsonga**
1	Love	Nandziha	Song	Nyika	Pain	Rila

2	Good	Leswinene	Music	Hansi	Problem	Nsele
3	Proud	Swinene	Artist	Risimu	Hate	Sola
4	Great	Nyungubyisa	Perform	Gaza	Disappointed	Chava
5	Awesome	Kahle	Play	Vaseketeri	Never	Xihlawuhlawu
6	Best	Rhandza	Listen	Henhla	Sad	Biha

Table 3.7 Sentimental phrases in Xitsonga

Xitsonga phrases		
Positive	**Neutral**	**Negative**
nyiko ya nwina a hlamarisa barhule mi nyikiwile eka dokodela hi ta mi rhandza hi la ku nga heli ku	ani tshembi kuri u yingisela kanwe ntsena nghoma leyi	yi twala kahle marha mipfa mi jila ku famba na marito
swakahle swiberiwa mandla ha khensa	hembe ya thomsa yini hleketisa xikale	mkhanselara isungule adya mali ya mpfula sweswi mfumu wunike swisiwana swakudya va dyini
leswinene swi beriwa mavoko ra namba risimu leri	nileku laveni ka risimu lero loko mihi vona hitshamile emindyangwini hitshama naswo tala riri yini vito	mipfa mi vitana naha papa hina ka mi tshika xihlawuhlawu

Table 3.8 Sentimental phrases for English

English phrases		
Positive	**Neutral**	**Negative**
It's the energy for me, love way you uplifts your culture, love your fashion sense and your mentality is so positive filled with life	You can't stop us neither wanna see us	The video doesnt play sound but on the main speaker, earpiece phone and tv theres sound
Best ever lead guitar I would pay to listen	I don't sleep without watching these videos	When I listen to this song I cry, stepfather may your soul rest peace
I cherish the song very much even though I dont understand the language, lyrics are interesting	I can't remember the song truly speaking	The music video is low and bad

3.5.3 Developing a lexicon and data annotation

The Valence Aware Dictionary and sEntiment Reasoner (VADER) is a sentiment analysis tool built in Python that uses a combination of lexicons and rules to process social media language, including slang, emoticons, abbreviations, and emojis (Pano and Kashef, 2020). VADER leverages a predefined dictionary of words and rules to assess the sentiment of a text (Bose et al., 2021). Due to its rule-based nature and lexical capabilities, VADER was particularly effective for classifying English comments.

The VADER sentiment analysis tool was first installed through the Natural Language Toolkit (NLTK) library. The SentimentIntensityAnalyzer library was then imported to evaluate the intensity or strength of sentiment in the text. An instance of the analyser class was initialized using the SentimentIntensityAnalyzer. All the English comments were

stored in a variable named comments, which were subsequently analysed using the analyser instance.

To determine the sentiment polarity, the polarity scores method was employed to compute the polarity and compound scores of each comment. Based on the compound score, the intensity thresholds were defined as follows:

- Comments with a compound score between 0.5 and 1.0 were classified as *positive*.
- Comments with a compound score between -0.5 and -1.0 were classified as *negative*.
- Comments with a compound score ranging from -0.5 to 0.5 were classified as neutral.

This approach facilitated an efficient and structured analysis of English comments.

With the sentiment words that were given to the annotators and those that they discovered during annotation, it was easier to develop a Xitsonga lexicon. Each word depending on their sentiment was assigned a polarity score, this was done considering the fact that all words with positive sentiments had to have a score between 0.5 and 1, and also a negative lexicon was assigned a score between -0.5 and 1. We then took the lexicon and put it into a VADER code together with the comments, the code checked if a word from the lexicon is within any of the comments and if yes then the comments are labelled given the polarity score of the word. To achieve that we first separated the comments into those that contain the sentiments and those that do not, again this was done on the code-mixed and Xitsonga comments using English and Xitsonga sentiments. For the code-mixed data, we developed lexicons for both Xitsonga and English, and labelling of comments followed the same process as described earlier. The remaining Xitsonga and code-mixed comments that had no sentimental words were again given to the volunteers to classify, this is where most of the neutral comments saw light.

Table 3.9 shows some of the words that were used to build a Xitsonga dictionary with their assigned polarity. Each word in the corpus was first analysed for its meaning within the context of the language. For instance, words like "rhanda," which means "love," and "nandziha," which can be interpreted as "enjoy" or "like," were understood to convey

positive sentiments. Conversely, words like "biha," meaning "bad," and "nsele," meaning "filth" or "rudeness," were recognized as conveying negative sentiments. The sentiment of each word was further refined based on how strongly it conveys a positive or negative emotion. For example, "rhanda" (love) conveys a pronounced positive sentiment, hence it was assigned a high positive polarity score of 0.9. On the other hand, "biha" (bad) expresses a strong negative sentiment, resulting in a high negative polarity score of -0.7.

Table 3.9 Sentimental words with assigned polarity scores

Word	Assigned polarity scored
rhanda	0.9
leswinene	0.7
nandziha	0.8
saseka	0.6
biha	- 0.7
nsele	- 0.8
sukela	- 0.6
tsokombela	0.9
rila	- 0.8
nyungubyisa	0.6

3.6 Classification of sentiments

This section outlines the approach followed to assign sentiment labels.

1. User profile of annotators and validators

Three annotators and one volunteer for validation were identified, all the annotators were students and proficient in both Xitsonga and English, including fluency, comprehension, and writing ability. They all have national matric certificates with Xitsonga as a first

language and English as a second language. The fourth person, the volunteer was brought in as a validator of the annotated sentiments.

2. Classification instructions and guidelines

This section outlines the methodology used for the classification of comments together with detailed information on the classification categories and the instructions given to the annotators.

- Classification categories: The annotators were given three labels to classify the comments; this includes positive, negative and neutral.
- Instructions and Guidelines: From the gathered comments, 300 were chosen at random; 100 of them are code-mixed, 100 are Xitsonga alone, and the remaining 100 are in English. An Excel spreadsheet including two columns—one for the comments and the other for the labels—was used to compile these comments. Before each annotator received the comments, each of them were given the instructions. The criteria were given, as well as the meanings of the labels. Users also received examples of how to name these feelings as well as examples of words that can be used to make neutral, positive, or negative comments. The sentiment of a written statement can also be inferred from its tone and emotions, so they also had to take it into account.

3. Sentiment Annotation Analysis

Table 3.10 shows the analysis of sentiment annotations by three annotators. The validator resolved disagreements and provided the final sentiment label for each comment.

Table 3.10 Sentiment annotation of comments by multiple annotators and a validator

Comments	**Annotator 1**	**Annotator 2**	**Annotator 3**	**Validator**
mpfumawulo lowu wuni nyanyule ku hinda mpimo	positive	positive	Positive	Positive
whats name song	neutral	positive	neutral	neutral

nita rila hi vutlhari anga ndzisani ya mina	negative	negative	neutral	negative
ive got much love admiration woman keep winning sho madjozi	positive	positive	positive	positive
proud ndzi ti rhandza swinene tinghoma ta nwina	positive	positive	positive	positive

4. Inter-Annotator Agreement

To quantify the reliability of the annotations, Fleiss' Kappa (κ) was calculated. This measure assesses the degree of agreement among multiple annotators, correcting for agreement that occurs by chance. Fleiss' Kappa is particularly suitable for this study, as it involves three annotators and categorical data (positive, negative, and neutral).

Fleiss' Kappa is calculated using the following formula:

$K = \frac{\bar{p}-\bar{p}_e}{1-\bar{p}_e}$

Where:

$\bar{p}$: The observed agreement among annotators.

$\bar{p}_e$: The expected agreement by chance.

Calculation of Fleiss' Kappa

Before calculating the inter-annotator agreement using Fleiss' Kappa, several steps were taken to organize and analyse the sentiment annotations. These steps are systematically presented by Table 3.11 and 3.12, which ensure a logical flow and clarity in the analysis process. Table 3.11 provides a detailed view of the sentiment labels assigned by each of the three annotators for the selected comments. This table serves as the foundation for understanding the initial classifications and identifying areas of agreement or disagreement among the annotators.

Table 3.11: Sentiment annotations by annotators

Comment ID	Annotator 1	Annotator 2	Annotator 3
1	Positive	Positive	Positive
2	Neutral	Positive	Neutral
3	Negative	Negative	Neutral
4	Positive	Positive	Positive
5	Positive	Positive	Positive

Table 3.12 summarizes the level of consensus among the annotators by counting how many annotators assigned each sentiment label (Positive, Neutral, or Negative) to every comment. This table is crucial for calculating Fleiss' Kappa, as it quantifies the observed agreement and forms the basis for further statistical analysis.

Table 3.12: Agreement counts for sentiment annotations

Comment ID	Positive	Neutral	Negative
1	3	0	0
2	1	2	0
3	0	1	2
4	3	0	0
5	3	0	0

Observed Agreement ($\overline{p}$):

$\overline{p} = \frac{1+ 0.333+ 0.333+1+1}{5} \approx 0.733$

Expected Agreement ($\overline{p}_e$):

$\overline{p}_e = 0.667^2 + 0.2^2 + 0.133^2 \approx 0.503$

Fleiss' Kappa:

$K = \frac{0.733-0.503}{1-0.503} \approx 0.463$

The use of Fleiss' Kappa to evaluate inter-annotator agreement provided a clear and quantifiable measure of the consistency of the sentiment annotations. The moderate agreement ($\kappa \approx 0.463$) suggests that while the annotators were generally aligned in their understanding and application of the sentiment labels, there were some disagreements that required resolution by the validator.

3.7 Developing the LSTM classifier

In this section, the focus is on the development of a Long Short-Term Memory (LSTM) classifier, a specialized type of recurrent neural network (RNN) designed for sequence prediction tasks. The LSTM model excels at understanding and retaining contextual relationships within sequential data.

For this study, the LSTM classifier was employed to analyse and classify textual data by leveraging the context embedded in sequences of words. Because of its architecture, the model can identify intricate linkages and patterns in the data, including long-term dependencies that are often critical for accurate classification. This capability makes the LSTM classifier an ideal choice for sentiment analysis tasks, where understanding the flow and meaning of word sequences significantly impacts model performance.

3.7.1 Tools used to develop the LSTM classifier

To develop the LSTM classifier for the Xitsonga-English code-mixed comments, we used the Python programming language within the Jupyter Notebook environment. Before starting the classifier development, we imported necessary modules. We used Pandas for tasks like importing, analysing, cleaning, exploring, and manipulating the data. We also imported other Python libraries required for the study, as listed in Table 3.13. The comments data we loaded into the system had two columns: one for the comments and another for the sentiments. Since TensorFlow (the tool we used to build the classifier) cannot directly process text, we added a new column to encode the sentiments

numerically—positive comments were marked with 1, neutral with 0, and negative with -1.

Table 3.13 List of Python libraries utilized in the study

Library	Usage
pandas	Data analysis and manipulation
numpy	Numerical computing
tensorflow	Deep learning framework
sklearn.model_selection	Dividing data into test and train sets
imblearn.over_sampling	Handling imbalanced data
gensim.models	word embedding using Word2Vec
sklearn.preprocessing	Label encoding for target variable
tensorflow.keras.models	Creating sequential model
tensorflow.keras.layers	Adding layers to neural network
tensorflow.keras.optimizers	Optimizing model's weights during training
tensorflow.keras.callbacks	Call-backs during training (e.g. early stopping)
keras.preprocessing.sequence	Pre-processing text data (e.g. padding sequence)

3.7.2 Developing the word embedding

To start, the neural network model's embedding layer was initialized using an embedding matrix. The Word2Vec model was used to create word embeddings, which are dense vector representations of words in a continuous vector space. The following parameters were used to develop the Word2Vec embeddings:

- Sentences: This parameter represents a list of sentences (or sequences of words) used to train the Word2Vec model. In this study, the sentences were extracted from the *Comments* column in the training dataset.

- Vector **s**ize: This describes the word vectors' (embeddings') dimensionality. Every word was represented by a 200-dimensional dense vector.
- Window: The maximum separation between the current and forecasted words in a sentence is known as the context window size for word embedding learning. For this study, it was fixed at 5.
- Min count: This parameter determines the minimum frequency of words to be included in the vocabulary. Words occurring fewer than two times were ignored by setting this value to 2.
- Workers: This indicates how many threads will be used to train the Word2Vec model in parallel. It was set to 4 for this study.

The provided sentences were then used to train the Word2Vec model, which produced word embeddings. In order to capture semantic similarities and correlations between words based on their co-occurrences in the training data, every word in the vocabulary was mapped to a dense vector with 200 dimensions.

3.7.3 Developing the LSTM classifier

Subsets of the dataset were separated for testing, validation, and training. The LSTM classifier was trained using the training subset, which made up 70% of the original data. During training, the classifier's performance was assessed and hyperparameters were adjusted using the validation subset, which comprised 15% of the data. After the training and validation stages were finished, the testing subset, which also made up 15% of the data, was set aside for evaluating the trained classifier's ultimate performance. Pre-processing procedures were completed before the classifier was developed, with an emphasis on the following crucial tasks to get the data ready for analysis:

Tokenization and padding

A tokenizer was initialized with num_words=5789, limiting the vocabulary size to 5789 unique words, and an out-of-vocabulary token <OOV> was defined for words not included in the vocabulary. The tokenizer was fitted on the training dataset comments to build the vocabulary. Text comments in the training, validation, and testing datasets were then converted into sequences of integers using this tokenizer. To ensure uniform input length,

these sequences were padded to a fixed length of 12 tokens (max_len=12) using the pad_sequences function.

Label encoding

Sentiment labels in the datasets were converted into numerical values using the LabelEncoder. The label encoder was fitted to the training data, and the encoded labels were applied to the training, validation, and testing datasets.

Handling imbalanced data

Using the Synthetic Minority Over-sampling Technique (SMOTE), the training dataset's class imbalance was addressed. Instead of creating duplicates, SMOTE blended pre-existing samples to create synthetic samples for underrepresented classes. By guaranteeing a more equitable distribution of classes in the training data, this enhanced the classifier's capacity to generalize and lessened prejudice towards the majority class.

3.7.3.1 Model architecture

The LSTM classifier architecture for sentiment analysis includes an embedding layer initialized with pre-trained word embeddings, enabling the model to represent words as dense vectors. Two LSTM layers come next, which are essential for identifying contextual dependencies and temporal correlations in the text. In order to classify sentiments into three categories, the model ends with a dense output layer that uses a softmax activation function. This architecture is tailored to effectively process code-mixed Xitsonga-English comments by combining the strengths of word embeddings and sequence learning.

Embedding layer (parameters)

- input_dim: Specifies the size of the vocabulary, which is set to 5789.
- output_dim: Defines the dimension of the dense embedding. In this case, it's set to word2vec_model.vector_size, which is the size of word vectors in the pre-trained Word2Vec model.
- input_length: Length of input sequences, which is set to 12.

- weights: Pre-trained word embeddings loaded from embedding_matrix. This parameter initializes the embedding layer with pre-trained word embeddings, making them non-trainable by default.

LSTM layers

- Two LSTM layers were added to the classifier. The first LSTM layer has 256 units and returns sequences. It also has a dropout of 0.3 to prevent overfitting. The second LSTM layer has 128 units and doesn't return sequences, indicating it's the last LSTM layer. It also has a dropout of 0.3.
- return_sequences: Determines whether to return the full sequence of outputs for each timestep. Setting it to true allows stacking LSTM layers.
- dropout: The percentage of units that must be dropped in order to transform the inputs linearly.
- recurrent_dropout: The fraction of units that must be dropped in order to alter the recurrent state linearly.

Dense (output) layer:

- With three units denoting the number of classes, the dense layer functions as the classifier's output layer.
- units: Specifies the dimensionality of the output space, which is set to 3 for multi-class classification.
- activation: The activation function applied to the output layer. In the case of our study, it's set to 'softmax' for multi-class classification to output probabilities for each class. Table 3.14 shows the layers of the classifier together with their parameters.

Table 3.14 Configuration of layers and parameters for LSTM classifier

Layer	Parameters
Embedding	• input_dim: 5789 • output_dim: word2vec_model.vector_size • input_length: 12 • weights: embedding_matrix

1st LSTM layer	• units: 256 • return_sequences: True • dropout: 0.3 • recurrent_dropout: 0.3
2nd LSTM layer	• units: 128 • dropout: 0.3 • recurrent_dropout: 0.3
Dense (Output layer)	• units: 3 • activation: 'softmax' • kernel_regularizer: 'l2'

3.7.3.2 Developing the classifier

The classifier was compiled for training with several arguments that define how it should be trained. The following are the key arguments:

Optimizer

The model in this study was trained using the Adam optimizer. Adam's flexible learning rate, which enhances stability and convergence speed, makes it popular. To guarantee gradual updates during training, avoid overshooting, and facilitate effective learning, the learning rate was set at 0.0001.

Loss function

The loss function selected for this multi-class classification task is sparse_categorical_crossentropy. This loss function is suitable when the target labels are integers, as it efficiently handles the comparison between predicted class probabilities and true class labels in a multi-class setting. The loss function is defined by the formula:

$$Loss = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(p^{(i)}, y^{(i)})$$

Where n is the number of samples, $y^{(i)}$ is the true class label for the i^{th} sample, and $(p^{(i)}, y^{(i)})$ is the predicted probability of the true class $y^{(i)}$ for the i^{th} sample.

3.7.3.3 Training the LSTM classifier

Train_labels_resampled and train_sequences_resampled were preprocessed training sequences that we used. For the complete training dataset, 30 epochs (iterations) were used. There were 32 samples in each gradient update. The classifier's weights are updated depending on the gradients computed on the training dataset, which is split into batches of size 32 in each epoch. The classifier's exposure to the full training data is determined by the number of epochs; 30 epochs is a popular beginning point that frequently offers a fair trade-off between training time and classifier performance. The batch size of 32 was selected as it is a commonly used value that provides a balance between computational efficiency and memory constraints. The validation data used to monitor the classifier's performance during training consists of preprocessed validation sequences (val_sequences) and corresponding encoded labels (val_labels_encoded).

3.8 Developing the stacked classifier

The study went on to developing a second classifier to compare with the LSTM that has been developed. The study developed a stacking classifier that combined the four most commonly used machine learning classifiers. Random Forest, Support Vector Machine, Gradient Boosting, and Logistic Regression are the four classifiers that were integrated. First, we loaded the dataset and imported various libraries necessary to create the classifier. A random state was set to 42, this was to ensure reproducibility.

TF-IDF vectorizer

The text input was transformed into numerical characteristics appropriate for model training using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. The max_features parameter was set to 5000 when the vectorizer was first started, which helped restrict the number of features and lower the data's dimensionality. The training data was subjected to the fit_transform approach, wherein the transform phase transformed the training text data into a sparse matrix of TF-IDF features, while the fit step learnt the vocabulary and calculated the IDF values for words. The TF-IDF score of a word is represented by each column in this matrix, whereas each row corresponds to a document.

Base classifiers

Three base classifiers—the Random Forest Classifier, Support Vector Classifier (SVC), and Gradient Boosting Classifier—were used for the classification task. To guarantee reproducibility, the Random Forest and Gradient Boosting classifiers each employed 100 trees and a random state of 42. The regularization parameter, which regulates the trade-off between model complexity and training error, was set to 1 for the Support Vector Classifier, which employed the Radial Basis Function (RBF) kernel. The probability parameter was set to true in all base classifiers to allow for probability predictions, which were later used as input features for the meta-model. The base classifiers' outputs were combined to form the final predictions through a stacking ensemble technique. Table 3.15 provides the specific parameters for each of these base classifiers.

Table 3.15 Detailed parameters for base classifier and their configurations

Classifier	**Parameter**	**Value**	**Description**
Random Forest Classifier	n_estimators	100	Number of trees in the forest.
Random Forest Classifier	random_state	42	Seed for the random number generator to ensure reproducibility.
Support Vector Classifier (SVC)	kernel	RBF	Specifies the kernel type to be used in the algorithm
Support Vector Classifier (SVC)	C	1	Regularization parameter, controls trade-off between margin size and classification error.
Support Vector Classifier (SVC)	probability	True	Enables probability estimates required for stacking.
Gradient Boosting Classifier	n_estimators	100	Number of boosting stages to be run.
Gradient Boosting	random_state	42	Seed for the

Classifier			random number generator to ensure reproducibility.

Define stacking classifier

After combining the basis classifiers, the Stacking classifier employed a logistic regression model as its final estimator. Stack_method='predict_proba' signifies that the meta-model employed the probability predictions from the basis models, and the cv=5 parameter specified 5-fold cross-validation. The training data converted by TF-IDF is used to train the stacked classifier. Table 3.16 displays the stacking classifier's parameters.

Table 3.16 Configuration and parameters for stacking classifier

Parameter	**Value**	**Description**
estimators	Three base classifier	Random Forest, Support Vector Machine, Gradient Boosting
final_estimator	LogisticRegression()	The final estimator that combines the base classifiers' predictions.
cv	5	Number of folds for cross-validation when training base classifiers.
stack_method	predict_proba	Method used to generate input features for the final estimator; uses probability estimates.

3.9 Data augmentation techniques for enhanced sentiment analysis

To enhance the diversity and robustness of the training set, we applied the Random Insertion method as our data augmentation technique. This approach involves inserting new, contextually relevant words into existing sentences to create additional samples. The objective was to generate a more varied dataset that could help improve the classifier's ability to generalize to unseen data. Before augmentation, the dataset was imbalanced, with the 'Positive' label having the most samples and 'Negative' the least. After augmentation, each label was balanced to 804 instances, creating a more evenly distributed training set. Table 3.17 compares the counts of positive, neutral, and negative

sentiments before and after data augmentation, highlighting the efforts to balance the dataset for improved model performance.

Table 3.17 Sentiment class distribution before and after augmentation

Label	**Original data**	**Augmented data**
Positive	804	804
Neutral	562	804
Negative	232	804

3.10. Conclusion

This study successfully analysed sentiments in Xitsonga-English code-mixed music reviews using different methods. We collected, cleaned, and prepared the data, then developed two types of sentiment classifiers: one using Long Short-Term Memory (LSTM) networks and another that combined several machine learning algorithms. We employed a data augmentation method known as Random Insertion to enhance the dataset's balance.

Chapter 4: Results

4.1 Introduction

This chapter presents the findings from the evaluation of the classifiers that were developed. This section describes the classifiers' performance metrics and results, including information on their accuracy and efficacy in relation to the evaluation criteria. We employed the following measures to assess the classifiers' performance: Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix. The performance of the developed classifiers on the training, testing, and validation datasets was evaluated using these measures.

4.2 The LSTM classifier performance

With an accuracy of 58%, the classifier was able to accurately predict 58% of the cases in the validation dataset. The test dataset received a score of 60%, meaning that 60% of the cases in the dataset were correctly predicted by the classifier.

4.2.1 Detailed classification report and analysis

Upon examining the classification report in Table 4.1, we can assess the performance of the classifier across three sentiment classes: negative, neutral, and positive.

Negative class:

- Precision**:** 0.40. This indicates that, out of all instances predicted as negative, only 40% were actually negative.
- Recall: 0.21. Only 21% of the actual negative cases were accurately predicted.
- F1 score: 0.28. For the negative class, the precision and recall harmonic mean is low, indicating a poor balance between them.
- Support: 57. This indicates how often the negative class actually occurs in the dataset.

The poor recall and precision imply that the model struggles to accurately detect and categorize negative instances. This could stem from an imbalance in the training data, or inherent challenges in distinguishing negative cases from the other classes.

Neutral class:

- Precision**:** 0.50. This indicates that 50% of instances predicted as neutral were correctly classified.
- Recall**:** 0.76. This shows that 76% of all actual neutral instances were correctly predicted.
- F1-Score**:** 0.60. This score reflects a moderate balance between precision and recall for the neutral class.
- Support**:** 143. The quantity of real instances of the neutral class.

The model appears to be effective at detecting neutral occurrences, as evidenced by the moderate precision and relatively high recall for the neutral class. However, this comes at the expense of some false positives.

Positive class:

- Precision**:** 0.79. This means that 79% of cases that were predicted to be positive turned out to be accurate.
- Recall**:** 0.60. This indicates that the prediction was accurate in 60% of all actual positive cases.
- F1-Score: 0.68. This implies that the positive class has a good balance between recall and precision.
- Support**:** 200. The quantity of actual instances of the positive class.

The positive class has the highest precision, meaning that when the model predicts positive, it is usually correct. However, the recall is lower, suggesting that the model misses some positive instances.

Overall, with an accuracy of 60%, the model's performance is moderate. The discrepancies between precision and recall across the different classes indicate that the model performs better with positive and neutral comments, but struggles with the negative class. The imbalance in class distribution and the model's difficulty in distinguishing negative instances are key factors contributing to this performance. The model's performance varies among sentiment classes, with the negative class presenting the

biggest obstacle, as further demonstrated by the discrepancies between weighted and macro averages.

Table 4.1 Classification report for LSTM classifier

	precision	recall	f1-score	support
negative	0.40	0.21	0.28	57
neutral	0.50	0.76	0.60	143
positive	0.79	0.60	0.68	200
accuracy			0.60	400
Macro avg	0.56	0.52	0.52	400
weighted avg	0.63	0.60	0.60	400

4.2.2 Comprehensive analysis of the confusion matrix

The confusion matrix provides insight into how well the model performs in terms of accurately and inaccurately assigning each label to instances. The following is our interpretation of the confusion matrix results that Table 4.2 provided:

Actual negative labels:

- Predicted negative (True Negatives): 12 instances were accurately categorized as negatives
- Predicted neutral (False Negatives): 38 instances that were actually negative were incorrectly categorized as neutral.
- Predicted positive (False Negatives): 9 instances that were actually negative were incorrectly categorized as positive.

Actual neutral labels:

- Predicted negative (false positives): 9 instances were incorrectly categorized as negative when they were actually neutral.

- Predicted neutral (true neutrals): 108 instances were correctly categorized as neutral.
- Predicted positive (false negatives): 26 instances were incorrectly categorized as positive when they were actually neutral.

Actual positive labels:

- Predicted negative (false positives): 9 instances that were actually positive were incorrectly classified as negative.
- Predicted neutral (false positives): 70 instances that were actually positive were incorrectly classified as neutral.
- Predicted positive (true positives): 121 instances were correctly classified as positive.

It's evident from the confusion matrix that the classifier struggled the most with distinguishing between negative and neutral sentiments, as well as between neutral and positive sentiments.

Table 4.2 Confusion matrix for LSTM classifier

	Negative predicted labels	Neutral predicted labels	Positive predicted labels
True negative labels	12	38	7
True neutral labels	9	108	26
True positive labels	9	70	121

4.2.3 Comments that have been misclassified by the classifier

Positive instances

- A significant 35% of true positive instances were predicted as neutral. This indicates an increasing difficulty in distinguishing between positive and neutral cases.

- 4.5% of positive instances are predicted as negative. While still relatively low, this suggests that the classifier may occasionally miss strong positive signals.

Neutral instances

- Approximately 18.18% of neutral instances are misclassified as positive. This persistent issue highlights the classifier's tendency to lean towards positive predictions when uncertain about neutral cases.
- 6.29% of neutral instances are predicted as negative. This suggests a modest improvement in the model's ability to correctly identify neutral instances.

Negative instances

- 12.28% of negative instances are misclassified as positive. This suggests that the classifier became better at not overestimating positive signals in negative cases.
- However, 66.67% of negative instances are predicted as neutral. This indicates a growing challenge in accurately identifying negative cases, likely due to insufficient negative examples in the training data.

4.2.4 Training the classifier

The training accuracy consistently increases and appears to stabilize towards the end, indicating the classifier's performance on the data it was trained with. The validation accuracy initially increases but then fluctuates around a certain point, indicating the model's performance on unseen data. The fact that it does not match the training accuracy suggests a gap in the model's generalization capability, which shows that the classifier is overfitting. This is presented by Figure 4.1.

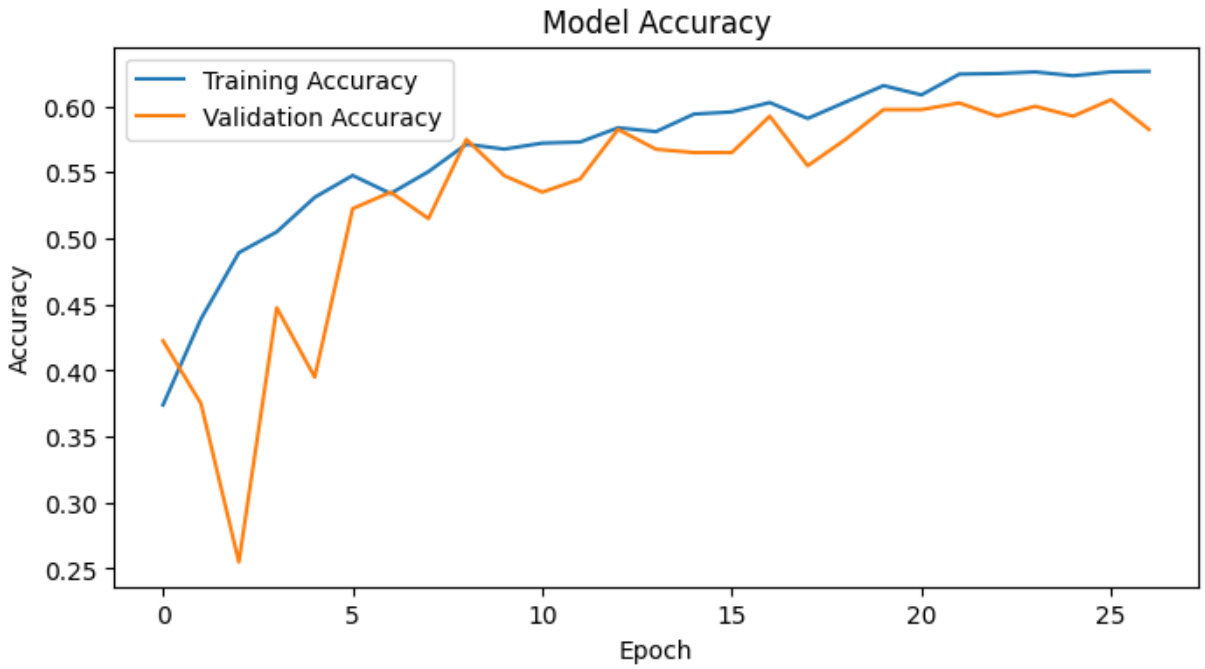

Figure 4.1 Comparison of training accuracy vs validation accuracy over epochs

The model loss curves indicate effective learning, with both training and validation loss decreasing significantly during the initial epochs and stabilizing thereafter. The small gap between the two suggests that the model generalizes reasonably well, with minimal overfitting. While the training loss is slightly lower than the validation loss, their parallel behaviour indicates that the model performs consistently on both seen and unseen data. This stability suggests that the classifier is well-trained and balances learning and generalization effectively. This is shown by Figure 4.2.

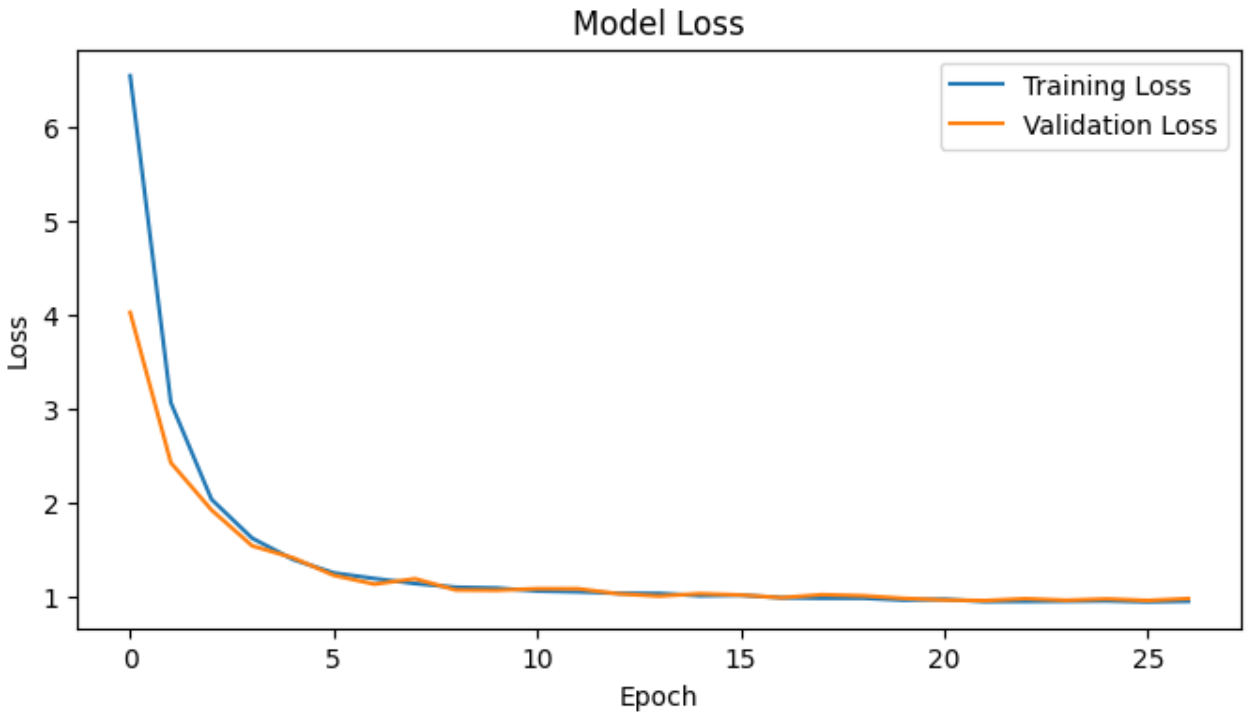

Figure 4.2 Comparison of training loss vs validation loss over epochs

4.3 Stacking classifier performance

The stacking model achieved an accuracy of 74%, highlighting its effectiveness in the classification of the comments.

4.3.1 Detailed classification report and analysis

After generating the classification report, we learnt the following:

- The Positive class had the highest precision (0.81) and recall (0.76), indicating the classifier performed best in identifying positive instances.
- The Neutral class also performed well, with a recall of 0.76, but slightly lower precision at 0.61.
- The Negative class had the lowest performance, with a precision of 0.66 and recall of 0.37, indicating that the classifier struggled with correctly identifying negative instances.
- The overall accuracy of the classifier is 0.70, which suggests that the classifier correctly predicted the class of instances 70% of the time.

- The macro average F1-score of 0.64 suggests that when each class is treated equally, the classifier's performance is lower compared to the weighted average F1-score of 0.70, which takes into account the class distribution. The classification report is shown by Table 4.3.

Table 4.3 Classification report for stacking classifier

	Precision	Recall	F1-score	Support
Negative	0.66	0.37	0.47	57
Neutral	0.61	0.76	0.67	143
Positive	0.81	0.76	0.78	200
Accuracy			0.70	400
Macro avg	0.69	0.63	0.64	400
Weighted avg	0.71	0.70	0.70	400

4.3.2 Comprehensive analysis of the confusion matrix

The confusion matrix reveals several important insights into the model's performance. The model excels at accurately identifying Positive and Neutral sentiments, indicating a strong understanding of these classes. However, there is notable confusion between Negative and Neutral classifications, suggesting that the model struggles to differentiate between these two categories. Additionally, the identification of Negative comments is less reliable, indicating that the model may require further refinement to improve its accuracy in this area. This is shown by Table 4.4.

Table 4.4 Confusion matrix for stacking classifier

	Predicted Negative labels	Predicted neutral labels	Predicted positive labels
True negative labels	21	25	11
True neutral labels	11	107	25
True positive labels	2	46	152

4.3.2 Comments that have been misclassified by the classifier

Positive instances

- **Neutral predictions**: A significant 23% of true positive instances were predicted as neutral. This misclassification indicates that the classifier struggles to distinguish between positive and neutral cases in almost a quarter of the instances.
- **Negative predictions**: Only 1% of positive instances were predicted as negative, suggesting that the classifier is quite effective at distinguishing positive cases from negative ones. This low misclassification rate indicates that the classifier's sensitivity to positive features is generally accurate, with minimal confusion in this area.

Neutral instances

- **Positive predictions**: Approximately 17.48% of neutral instances were incorrectly classified as positive. This might be due to some neutral instances having features that are more similar to positive ones or due to imbalanced training data that favours positive predictions.

- **Negative predictions**: About 7.69% of neutral instances were predicted as negative. This lower rate compared to the positive misclassification indicates a moderate challenge in differentiating between neutral and negative cases.

Negative instances

- **Positive predictions**: Nearly 19.30% of negative instances were predicted as positive, indicating a substantial misclassification issue. This misclassification suggests that the classifier might be overestimating positive signals, possibly due to biased training data or classifier's parameters that overly favour positive predictions.
- **Neutral predictions**: A notable 43.86% of negative instances were predicted as neutral. This high misclassification rate highlights a significant challenge in the classifier's ability to accurately identify negative cases. This could be due to feature overlap or insufficient representation of negative cases during training.

4.4 Some of reasons why the LSTM and stacked classifiers misclassified the comments

A significant number of the misclassified comments are code-mixed and Xitsonga comments. To gain a deeper understanding of the challenges the models encountered, we have outlined several potential reasons for the misclassifications. Through examining these incorrectly classified instances, we aim to uncover common patterns or features that might have contributed to these errors.

1. Challenges in language interpretation and mixing

   Some comments can be hard to interpret and might have more than one possible meaning. For example, a phrase like "damn I remember this song when I was still a kid" was classified as positive because of nostalgia while the statement is just neutral. Comments that express both positive and negative emotions, like nostalgia mixed with sadness, can be difficult for the classifier to classify accurately. For example, "mara ku reply ti comment ku like nyana won't hurt" the word "like" could be seen as positive and the word "hurt" can be seen as negative.

2. Complexities in understanding language slangs

   If the comments include local slang, the classifier might struggle to understand the sentiment accurately. For example, "wa dlaya hee mhani miswi hleketa nkarhi muni

marha" which translates to “you kill mother, when do you get time to think this” contains a slang word that the classifier doesn’t fully capture. The word “dlaya” which means to *kill* can be used as a slang in Xitsonga which can mean “*one rocks*”. The classifier classified this comment as negative.

3. Imbalance training data

   The data that was collected had few negative comments which led to the training data to have even fewer negative samples, this might have caused the classifier to underperform on the negative comments.

4. Complex sentence structure

   Comments with complex sentence structures can confuse the classifier, leading to misclassification. For instance, the classifier might misinterpret long, complex sentences with multiple clauses. Moreover, handling of negations can be tricky; For example, a comment like "don't like it" is misclassified as positive because the classifier failed to recognize the negation properly.

4.5 Conclusion

In conclusion, the stacking classifier outperformed the LSTM classifier, achieving an accuracy of 74%, compared to the LSTM's maximum accuracy of 60%. This superior performance can be giving to the stacking classifier's ability to combine predictions from multiple models, enhancing its generalization capabilities and improving sentiment classification. The stacking classifier performed well across all sentiment classes, but it was especially good at detecting positive sentiments, whereas the LSTM had trouble with negative instances.

Chapter 5: Conclusion

5.1 Summary of findings

This study evaluated the performance of various classifiers, with a particular focus on the LSTM classifier and the stacking classifier. Key findings include:

- **Generalization issues:** Both classifiers showed challenges in generalizing to unseen data, indicating a tendency towards overfitting.
- **Class imbalance:** The classifiers were more effective at identifying positive instances but struggled with negative instances, which suggests possible class imbalance in the dataset.
- **Feature representation:** Moderate performance in identifying neutral comments points to potential issues with feature representation, indicating a need for improved feature engineering.

5.2 Conclusion

The study successfully achieved its objectives by collecting 1,998 Xitsonga-English code-mixed music reviews. The polarity of the comments was determined to give a sentiment distribution of 50.25% positive, 35.29% neutral, and 14.46% negative. A sentiment classifier was developed using long short-term memory (LSTM) techniques, and the classifier's performance was thoroughly evaluated. An additional stacking classifier was developed. The evaluation of the LSTM and stacking classifiers reveals significant insights into their performance and generalization capabilities. Both classifiers demonstrated high training accuracy but struggled with unseen data, highlighting the need for strategies to mitigate overfitting and improve generalization. By implementing the recommended strategies and focusing on future research directions, the performance of these classifiers can be enhanced, leading to more robust and accurate sentiment analysis models.

Each objective contributed directly to the aim of developing a code-mixed sentiment analysis model for Xitsonga-English music reviews. Data collection provided the raw

dataset, while sentiment polarity determination ensured accurate labelling for training. Model development created an LSTM-based classifier, leveraging its ability to handle sequential code-mixed data. Finally, model evaluation assessed performance using metrics like accuracy and F1-score, ensuring the model's effectiveness and generalizability. Together, these steps systematically addressed the challenges of code-mixed sentiment analysis, achieving the goal of building a robust and region-specific sentiment analysis model.

5.3 Future work

To improve the performance and generalization of these classifiers, several strategies can be employed:

1. Cross-validation: Using cross-validation can guarantee that the model performs consistently across several data subsets.
2. Increasing training Data: Gathering more training data, especially for the underperforming classes, can help the model learn patterns that generalize better to unseen data.
3. Feature engineering: Enhancing feature engineering to better distinguish between the classes, particularly negative and neutral, can improve classification performance.
4. In future, the introduction of Bidirectional Gated Recurrent Units (BiGRU) is planned to enhance the model's ability to capture contextual information for Xitsonga-English code-mixed comments.

References

Ahmad, G. I., Singla, J., Ali, A., Reshi, A. A., and Salameh, A. A. (2022) 'Machine Learning Techniques for Sentiment Analysis of Code-Mixed and Switched Indian Social Media Text Corpus: A Comprehensive Review', *International Journal of Advanced Computer Science and Applications*, 13(2), pp. 455–467.

Ahuja, R., Chug, A., Kohli, S., Gupta, S., and Ahuja, P. (2019) 'The impact of features extraction on the sentiment analysis', in *Procedia Computer Science*. Elsevier B.V., pp. 341–348.

Ain, Q. T., Ali, M., Riaz, A., Noureen, A., Kamran, M., Hayat, B., and Rehman, A. (2017) Sentiment analysis using deep learning techniques: A review. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *8*(6), pp. 424-433.

Akhtar, M. S., Kumar, A., Ghosal, D., & Ekbal, A. (2020). "A Multilayer Perceptron-based Approach for Sentiment Analysis in Code-Mixed Text." Proceedings of the 28th International Conference on Computational Linguistics, 1234-1245.

Alexander, M (2023). 'The 11 languages of South Africa'. Available at: https://southafrica-info.com/arts-culture/11-languages-south-africa/ (Accessed: 27 October 2023).

Ali, M., Khan, M. A., and Asghar, M. N. (2019). *Logistic Regression: A Powerful Tool for Binary Classification in Data Mining*. *International Journal of Advanced Computer Science and Applications*, 10(4), pp. 163-170.

Ameer, I., Sidorov, G., Gómez-Adorno, H., and Nawab, R. M. A. (2022) 'Multi-Label Emotion Classification on Code-Mixed Text: Data and Methods', *IEEE Access*, 10, pp. 8779–8789.

Anbukkarasi S (2020) *SA-SVG@Dravidian-CodeMix-FIRE2020: Deep Learning Based Sentiment Analysis in Code-mixed Tamil-English Text*. *Proceedings of the Forum for*

Information Retrieval Evaluation (FIRE), 2826, pp. 123-130.

Asian, J., Dholah Rosita, M. and Mantoro, T. (2022) 'Sentiment Analysis for the Brazilian Anesthesiologist Using Multi-Layer Perceptron Classifier and Random Forest Methods', *Jurnal Online Informatika*, 7(1), p. 132.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). *SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining*. Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10), pp. 2200-2204

Baid, P., Gupta, A. and Chaplot, N. (2017) Sentiment analysis of movie reviews using machine learning techniques. *International Journal of Computer Applications*, 179(7), pp. 45-49.

Balouchzahi, F., Lakshmaiah Shashirekha, H. and Sidorov, G. (2021) CoSaD-Code-Mixed Sentiments Analysis for Dravidian Languages. *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages (DravidianLangTech-EACL2021)*, pp. 1-8.

Bharathi, B. and Samyuktha, G.U. (2021) Bharathi, B. and Samyuktha, G.U. (2021) Machine learning based approach for sentiment analysis on multilingual code mixing text. *Proceedings of the Forum for Information Retrieval Evaluation (FIRE)*, 3159, pp. 1-8.

Bonta, V., Kumaresh, N. and Janardhan, N. (2019) 'A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis', *Asian Journal of Computer Science and Technology*, 8(S2), pp. 1–6.

Bose, R., Aithal, P.S. and Roy, S. (2021) 'Survey of Twitter Viewpoint on Application of Drugs by VADER Sentiment Analysis among Distinct Countries', *International Journal of Management, Technology, and Social Sciences*, (March 2021), pp. 110–127.

Bühlmann, P., and Yu, B. (2002). *Analyzing bagging*. *The Annals of Statistics*, 30(4), pp.

927-961.

Cambria, E., Havasi, C. and Hussain, A. (2012) SenticNet 2: A semantic and affective resource for opinion mining and sentiment analysis. *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 202-207.

Cambria, E., Schuller, B., Xia, Y., and Havasi, C. (2017). *New Avenues in Opinion Mining and Sentiment Analysis*. IEEE Intelligent Systems, 28(2), pp.15-21.

Cambria, E., Poria, S., and Gelbukh, A. (2013). *SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Common Sense Knowledge*. Proceedings of the 27th International Conference on Computational Linguistics (COLING 2014), pp. 299–307.

Chakravarthi, B. R., Jose, N., Suryawanshi, S., Sherly, E., and McCrae, J. P. (2020). A sentiment analysis dataset for code-mixed Malayalam-English. *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pp. 177-184.

Chakravarthi, B. R., Muralidaran, V., Priyadharshini, R., and McCrae, J. P. (2020). Corpus creation for sentiment analysis in code-mixed Tamil-English text. Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), pp. 202-210.

Chakravarthi, B. R., Priyadharshini, R., Muralidaran, V., Suryawanshi, S., Sherly, E., and McCrae, J. P. (2020) 'Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text', *ACM International Conference Proceeding Series*, pp. 21–24.

Chakravarthi, B.R. *et al.* (2021) 'Findings of the Sentiment Analysis of Dravidian Languages in Code-Mixed Text', *CEUR Workshop Proceedings*, 3159, pp. 872–886.

Chiny, M., Chihab, M., Bencharef, O., and Chihab, Y. (2021) 'LSTM, VADER and TF-IDF based Hybrid Sentiment Analysis Model', *International Journal of Advanced Computer Science and Applications*, 12(7), pp. 265–275.

Choudhary, N., Singh, R., Bindlish, I., and Shrivastava, M. (2018). Sentiment analysis of code-mixed languages leveraging resource rich languages. *Proceedings of the 19th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, pp. 104-114.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. In *Proceedings of the 23rd International Conference on Machine Learning* (Vol. 28, pp. 84-92). PMLR.

Damarta, R., Hidayat, A. and Abdullah, A.S. (2021). The application of k-nearest neighbors classifier for sentiment analysis of PT PLN (Persero) twitter account service quality. *Journal of Physics: Conference Series*, 1722, 012002.

Dara, S., Wankhade, M., Chandra Sekhara Rao, A., and Kaushik, B. (2017) 'A Sentiment Analysis of Food Review using Logistic Regression Cost Efficient Protocol Design For Internet of Vehicle View project Speech Recognition View project Mayur Wankhade 2 PUBLICATIONS 12 CITATIONS SEE PROFILE A Sentiment Analysis of Food Review using Logistic Regression', 2(7), pp. 251–260.

Darwich, M., Mohd Noah, S. A., Omar, N., and Osman, N. A. (2019) 'Corpus-Based Techniques for Sentiment Lexicon Generation: A Review', *Journal of Digital Information Management*, 17(5), p. 296.

Davis, J., and Goadrich, M. (2006). *The Relationship Between Precision-Recall and ROC Curves*. In Proceedings of the 23rd International Conference on Machine Learning (ICML 2006), pp. 233-240.

Dou, Z.Y. (2017) 'Capturing user and product information for document level sentiment analysis with deep memory network', *EMNLP 2017 - Conference on Empirical Methods in*

Natural Language Processing, Proceedings, pp. 521–526.

Dutta, S., Agrawal, H. and Roy, P.K. (2021) 'Sentiment Analysis on Multilingual Code-Mixed Kannada Language', *CEUR Workshop Proceedings*, 3159, pp. 908–918.

Emily¨, E. and Emily¨ohman (2021) 'The Validity of Lexicon-based Emotion Analysis in Interdisciplinary Research', pp. 7–12.

Esuli, A., Sebastiani, F. and Moruzzi, V.G. (2006) 'S ENTI W ORD N ET : A Publicly Available Lexical Resource for Opinion Mining', pp. 417–422.

Fu, X., Chen, C., Rahman Laskar, M. T., Gardiner, S., Hiranandani, P., and Bhushan, S. (2019). Entity-level sentiment analysis in contact center telephone conversations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 484-491.

Garain, A., Mahata, S.K. and Das, D. (2020) JUNLP@SemEval-2020 Task 9: Sentiment analysis of Hindi-English code mixed data using grid search cross validation. *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pp. 1276-1280.

Gers, F.A., Schraudolph, N.N. and Schmidhuber, J. (2003) 'Learning precise timing with LSTM recurrent networks', *Journal of Machine Learning Research*, 3(1), pp. 115–143.

Guerini, M., Gatti, L. and Turchi, M. (2013) Sentiment analysis: How to derive prior polarities from SentiWordNet. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1259-1269.

Guia, M., Silva, R.R. and Bernardino, J. (2019) 'Comparison of Naive Bayes, support vector machine, decision trees and random forest on sentiment analysis', *IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 1(Ic3k), pp. 525–531.

Gupte, A., Joshi, S., Gadgul, P., and Kadam, A. (2014). Comparative study of

classification algorithms used in sentiment analysis. *International Journal of Computer Science and Information Technologies*, 5(5), pp. 6261-6264.

Hande, A., Priyadharshini, R. and Chakravarthi, B.R. (2020) KanCMD: Kannada CodeMixed Dataset for Sentiment Analysis and Offensive Language Detection. *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media (PEOPLES)*, pp. 54-63.

Hochreiter, S., and Schmidhuber, J. (1997). *Long Short-Term Memory*. Neural Computation, 9(8), pp. 1735-1780.

Hutto, C. J., and Gilbert, E. E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, pp. 216-225.

Javdan, S., Shangipour Ataei, T., and Minaei-Bidgoli, B. (2020). IUST at SemEval-2020 Task 9: Sentiment analysis for code-mixed social media text using deep neural networks and linear baselines. *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pp. 1270-1275.

Jhanwar, M.G. and Das, A. (2018). An ensemble model for sentiment analysis of Hindi-English code-mixed data. *Proceedings of the 1st Workshop on Humanizing AI (HAI) at IJCAI*, pp. 1-8.

Joshi, N. and Itkat, S. (2014) 'A Survey on Feature Level Sentiment Analysis', *International Journal of Computer Science and Information Technologies*, 5(4), pp. 5422–5425.

Kanwar, N., Agarwal, M. and Kumar Mundotiya, R. (2020) PITS@Dravidian-CodeMix-FIRE2020: Traditional approach to noisy code-mixed sentiment analysis. *Proceedings of the Forum for Information Retrieval Evaluation (FIRE)*, 2826, pp. 541-547.

Khanvilkar, G. and Vora, D. (2018) 'Sentiment analysis for product recommendation using random forest', *International Journal of Engineering and Technology (UAE)*, 7(3), pp. 87–

89.

Kharde, V.A. and Sonawane, S.S. (2016) Sentiment analysis of Twitter data: A survey of techniques. *International Journal of Computer Applications*, 139(11), pp. 5-15.

Kolchyna, O., Souza, T. T. P., Treleaven, P., and Aste, T. (2015). Twitter sentiment analysis: Lexicon method, machine learning method and their combination. *arXiv preprint arXiv:1507.00955*. https://doi.org/10.48550/arXiv.1507.00955 (Accessed: 07 January 2024).

Kolkur, S., Dantal, G., and Mahe, R. (2015) 'Study of Different Levels for Sentiment Analysis', *International Journal of Current Engineering and Technology*, 5(2), pp. 768–770.

Kumar, A., Gupta, S., and Singh, J. (2016). *Performance Evaluation Metrics for Classification Algorithms in Data Mining: A Review*. International Journal of Computer Applications, 140(9), pp. 5-10.

Kumar, A. and Sebastian, T.M. (2012) 'Sentiment Analysis: A Perspective on its Past, Present and Future', *International Journal of Intelligent Systems and Applications*, 4(10), pp. 1–14.

Lal, Y. K., Kumar, V., Dhar, M., Shrivastava, M., and Koehn, P. (2019). De-mixing sentiment from code-mixed text. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pp. 371-377.

Learned-miller, E.G. (2011) Supervised learning and Bayesian classification. Department of Computer Science, University of Massachusetts, Amherst, pp. 1–8.

Lipton, Z.C., Elkan, C. and Narayanaswamy, B. (2014) 'Thresholding Classifiers to Maximize F1 Score'. Available at: http://arxiv.org/abs/1402.1892 (Accessed: 22 March 2024).

Loria, S. (2018). *TextBlob: Simplified Text Processing*. Available at: https://textblob.readthedocs.io/en/dev/ (Accessed: 04 October 2024).
Mabaso, C. (2014). *Xitsonga Language and its Dialects: A Study of Tswa and Ronga*. University of Limpopo. Available at: http://ulspace.ul.ac.za/ (Accessed: 06 October 2024).

Mabokela, K.R. and Schlippe, T. (2022) 'A Sentiment Corpus for South African Under-Resourced Languages in a Multilingual Context', *1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages, SIGUL 2022 - held in conjunction with the International Conference on Language Resources and Evaluation, LREC 2022 - Proceedings*, pp. 70–77.

Mandal, S., Mahata, S.K. and Das, D. (2018) Preparing Bengali-English code-mixed corpus for sentiment analysis of Indian languages. *Proceedings of the 13th Workshop on Asian Language Resources (ALR)*, pp. 1-8.

Mäntylä, M. V., Graziotin, D. and Kuutila, M. (2018) 'The evolution of sentiment analysis—A review of research topics, venues, and top cited papers', *Computer Science Review*, 27, pp. 16–32.

Mas Diyasa, I. G. S., Mandenni, N. M. I., Fachrurrozi, M. I., Pradika, S. I., Manab, K. R. N., and Sasmita, N. R. (2021) 'Twitter Sentiment Analysis as an Evaluation and Service Base On Python Textblob', *IOP Conference Series: Materials Science and Engineering*, 1125(1), p. 012034.

Medhat, W., Hassan, A. and Korashy, H. (2014) 'Sentiment analysis algorithms and applications: A survey', *Ain Shams Engineering Journal*, 5(4), pp. 1093–1113.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2010). *Distributed Representations of Words and Phrases and Their Compositionality*. In *Advances in Neural Information Processing Systems* (Vol. 26).

Mohammad, S.M. (2016) 'A practical guide to sentiment annotation: Challenges and solutions', *Proceedings of the 7th Workshop on Computational Approaches to*

Subjectivity, Sentiment and Social Media Analysis, WASSA 2016 at the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 174–179.

Mtetwa, S. (2005). Cultural Practices and Language Preservation in Tsonga Society. *South African Journal of African Languages*, 25(2), pp. 56-68.

Muhammad, P.F., Kusumaningrum, R. and Wibowo, A. (2021) 'Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM) for Indonesian Hotel Reviews', in *Procedia Computer Science*. Elsevier B.V., pp. 728–735.

Muhammad, S.H. *et al.* (2023) AfriSenti: A Twitter sentiment analysis benchmark for African languages. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 13968-13981.

Nayak, A. and Natarajan, S. (2016) Comparative study of Naïve Bayes, Support Vector Machine and Random Forest classifiers in sentiment analysis of Twitter feeds. *International Journal of Advanced Studies in Computer Science and Engineering*, 5(1), pp. 1-8.

Nistor, S. C., Moca, M., Moldovan, D., Oprean, D. B., and Nistor, R. L. (2021). Building a Twitter sentiment analysis system with recurrent neural networks. *Sensors*, 21(7), pp. 2266.

Nkondo, M. (2013). Efforts to Preserve African Languages: The Case of Xitsonga. *Journal of African Cultural Studies*, 25(3), pp. 247-259.

Pano, T. and Kashef, R. (2020) 'A complete vader-based sentiment analysis of bitcoin (BTC) tweets during the ERA of COVID-19', *Big Data and Cognitive Computing*, 4(4), pp. 1–17.

Patel, A. and Tiwari, A.K. (2019) *Sentiment Analysis by using Recurrent Neural Network*. Available at: https://ssrn.com/abstract=3349572 (Accessed: 06 January 2024).

Patra, B.G., Das, D. and Das, A. (2018) ‘Sentiment Analysis of Code-Mixed Indian 74 | P a g e Languages: An Overview of SAIL_Code-Mixed Shared Task @ICON-2017’. Available at: http://arxiv.org/abs/1803.06745 (Accessed: 23 June 2024).

Patwa, P. *et al.* (2020). *SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets*. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 774–790.

Poria, S., Cambria, E. and Gelbukh, A. (2015) *Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2539–2544.

Powers, D. M. (2011). *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation*. *Journal of Machine Learning Technologies*, 2(1), pp. 37-63.

Prabhu, A., Joshi, A., Shrivastava, M., and Varma, V. (2016). *Towards sub-word level compositions for sentiment analysis of Hindi-English code mixed text*. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 2482–2491.

R, M.N.J. and Balaji, K. (2016) *Performance analysis of neural networks and support vector machines using confusion matrix.* International Journal of Advanced Research in Science, Engineering and Technology, 3(5), pp, 2106–2109.

Rani, S. and Kumar, P. (2019) ‘Deep Learning Based Sentiment Analysis Using Convolution Neural Network’, *Arabian Journal for Science and Engineering*, 44(4), pp. 3305–3314.

Reddy Sane, S., Tripathi, S., Reddy Sane, K., and Mamidi, R. (2019). *Deep learning techniques for humor detection in Hindi-English code-mixed tweets*. In Proceedings of the

Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 57–61.

Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) *Model-agnostic interpretability of machine learning*. In Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016), pp. 113–118.

Sabri, N., Edalat, A. and Bahrak, B. (2021) 'Sentiment Analysis of Persian-English Code-mixed Texts'. Available at https://arXiv:2102.12700 (Accessed: 23 January 2024).

Sabty, C., Elmahdy, M. and Abdennadher, S. (2019) 'Named Entity Recognition on ArabicEnglish Code-Mixed Data', Proceedings - 13th IEEE International Conference on Semantic Computing, ICSC 2019, pp. 93–97.

Sachin, S., Tripathi, A., Mahajan, N., Aggarwal, S., and Nagrath, P. (2020). *Sentiment analysis using gated recurrent neural networks*. SN Computer Science, 1(2), Article 74.

Sadia, A., Khan, F., and Bashir, F. (2018). An overview of lexicon-based approach for sentiment analysis. In *2018 3rd International Electrical Engineering Conference (IEEC 2018).* IEP Centre, Karachi, Pakistan, pp. 1-5.

Saito, T., and Rehmsmeier, M. (2015). *The Precision-Recall Plot is More Informative than the ROC Plot*. *Nature Methods*, 12(3), pp. 209-210.

Salma, A. and Silfianti, W. (2021) 'Sentiment Analysis of User Review on COVID-19 Information Applications Using Naïve Bayes Classifier, Support Vector Machine, and K-Nearest Neighbors', *International Research Journal of Advanced Engineering and Science*, 6(4), pp. 158–162.

Setati, M. (2002). Code-Switching in South African Classrooms: Language Practices in Multilingual Settings. *Perspectives in Education*, 20(1), pp. 1-13.

Shanmugavadivel, K., Janani, J. S., Navbila, K., and Subramanian, M. (2022) 'An analysis

of machine learning models for sentiment analysis of Tamil code-mixed data', *Computer Speech and Language*, 76(January), p. 101407.

Shirsat, V.S., Jagdale, R.S. and Deshmukh, S.N. (2018) 'Sentence level sentiment identification and calculation from news articles using machine learning techniques', *Advances in Intelligent Systems and Computing*, 810(5), pp. 371–376.

Singh, K., Sen, I. and Kumaraguru, P. (2018) *A Twitter corpus for Hindi-English code mixed POS tagging*. In Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, pp. 12–17.

Singh, P. and Lefever, E. (2020) *Sentiment Analysis for Hinglish Code-mixed Tweets by means of Cross-lingual Word Embeddings*, *European Language Resources Association (ELRA)*. In Proceedings of the 4th Workshop on Computational Approaches to Code Switching, pp. 45–51.

Smagulova, K. and James, A.P. (2019) 'A survey on LSTM memristive neural network architectures and applications', *European Physical Journal: Special Topics*, 228(10), pp. 2313–2324.

Sokolova, M., and Lapalme, G. (2009). *A Systematic Analysis of Performance Measures for Natural Language Generation*. In Proceedings of the 12th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 245-252.

Solorio, T. *et al.* (2014) 'Overview for the First Shared Task on Language Identification in Code-Switched Data', *1st Workshop on Computational Approaches to Code Switching, Switching 2014 at the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014 - Proceedings*, pp. 62–72.

Srinivasan, R. and Subalalitha, C.N. (2021) *Sentimental analysis from imbalanced code-mixed data using machine learning approaches*. Distributed and Parallel Databases, 41, pp. 37–52.

Steinberger, J., Brychcín, T. and Konkol, M. (2014) ‘Aspect-Level Sentiment Analysis in Czech’, *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 24–30.

Sultan, A., Salim, M., Gaber, A., and El Hosary, I. (2020). *WESSA at SemEval-2020 Task 9: Code-Mixed Sentiment Analysis using Transformers*. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 1342–1347.

Sweeney, C. (2017) ‘Multi-entity sentiment analysis using entity-level feature extraction and word embeddings approach’, pp. 733–740.

Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., and Gottschlich, J. (2018) ‘Precision and recall for time series’, *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS), pp. 1920–1930.

Tembhurne, J. V. and Diwan, T. (2021) ‘Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks’, *Multimedia Tools and Applications*, 80(5), pp. 6871–6910.

Tian, Y., Chen, X., & Zhang, Y. (2018). "Sentiment Polarity as a Robust Metric for Sentiment Classification." Journal of Natural Language Processing, 25(3), 45-60.

Tho, C., Heryadi, Y., Lukas, L., and Wibowo, A. (2021). *Code-mixed sentiment analysis of Indonesian language and Javanese language using Lexicon based approach*. In Journal of Physics: Conference Series (Vol. 1869, No. 1, p. 012084).

Tyagi, A. and Sharma, N. (2018) ‘Sentiment Analysis using logistic regression and effective word score heuristic’, *International Journal of Engineering and Technology(UAE)*, 7(2), pp. 20–23.

Usman, M., Shafique, Z., Ayub, S., and Malik, K. (2016) ‘Urdu Text Classification using Majority Voting’, *International Journal of Advanced Computer Science and Applications*,

7(8), pp. 265–273.

Wakade, S., Kulkarni, S., Deshmukh, S., and Joshi, A. (2012) *Text Mining for Sentiment Analysis of Twitter Data*. Available at: http://jaiku.com (Accessed: 13 January 2024).

Wang, X., Jiang, W. and Luo, Z. (2016) *Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts*. Available at: https://github.com/ultimate010/crnn (Accessed: 07 January 2024).

Wang, Y., Huang, M., Zhu, X., and Zhao, L. (2016) 'Attention-based LSTM for aspect-level sentiment classification', *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 606–615.

Wankhade, M., Rao, A.C.S. and Kulkarni, C. (2022) 'A survey on sentiment analysis methods, applications, and challenges', *Artificial Intelligence Review*, 55(7), pp. 5731–5780.

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., and Grave, E. (2019). CCNet: Extracting high quality monolingual datasets from web crawl data. Available at https://arXiv:1911.00359 (Accessed: 20 January 2024).

Woon, J. and Ho, Y. (2007) 'Code-mixing: Linguistic form and socio-cultural meaning', *The International Journal of Language Society and Culture*, (21), p. 8.

Yang, B. and Cardie, C. (2014) 'Context-aware learning for sentence-level sentiment analysis with posterior regularization', *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1(2008), pp. 325–335.

Yessenalina, A., Yue, Y. and Cardie, C. (2010) 'Multi-level structured models for document-level sentiment classification', *EMNLP 2010 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, (October), pp. 1046–1056.

Yimam, S. M., Ayele, A. A., Biemann, C., and Ruppenhofer, J. (2020). Exploring Amharic sentiment analysis from social media texts: Building annotation tools and classification models. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pp. 1048–1060.

Zerbian, S. (2007) 'A First Approach to Information Structuring in Xitsonga/ Xichangana', SOAS Working Papers in Linguistics, 15, pp. 65–78.

Zhang, J. (2016). *k-Nearest Neighbor Algorithm for Classification*. In *Encyclopedia of Data Science* pp. 1-5. Springer.

Zhang, M. and Qian, T. (2020) 'Convolution over hierarchical syntactic and lexical graphs for aspect level sentiment analysis', *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 3540–3549.

Zhang, Y., Wang, X., and Xie, D. (2019). *A survey on support vector machines and their applications in bioinformatics*. *International Journal of Data Mining and Bioinformatics*, 23(4), pp. 389-404.

Ziervogel, D., Louw, J. A., and Taljaard, P. C. (1971). *A Handbook of the Xitsonga Language*. Pretoria: Van Schaik Publishers.