RESEARCH ARTICLE

Comparing Fine-Tuned RoBERTa with Traditional Machine Learning Models for Stance Detection in Political Tweets

Bilal Khan[1], Khairullah Khan[1,*], Fida Muhammad Khan[2,*], Haseena Noureen[3], Ahmad Ali[4] and Mohsin Shah[5]

[1] Department of Computer Science and Information Technology, University of Science and Technology Bannu, Bannu, Pakistan
[2] Department of Computer Science, Qurtuba University of Science and Information Technology, Peshawar Campus, Peshawar, Pakistan
[3] Department of Computer Science and Information Technology, University of Malakand, Chakdara 18800, Pakistan
[4] College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China
[5] Department of Telecommunication, Hazara University, Mansehra, Khyber Pakhtunkhwa, Pakistan

Abstract

Stance detection identifies a text's position or attitude toward a given subject. A major challenge in Roman Urdu is the lack of a publicly available dataset for political stance detection. To address this gap, we constructed a high-quality dataset of 8,374 political tweets and comments using the Twitter API, annotated with stance labels: agree, disagree, and unrelated. The dataset captures diverse political viewpoints and user interactions. For feature representation, we employed TF-IDF due to its effectiveness in handling high-dimensional, context-sensitive Roman Urdu text. Several machine learning classifiers were evaluated, with Random Forest achieving the highest accuracy of 95%. Additionally, we fine-tuned the transformer-based RoBERTa model, which outperformed traditional methods with 97% accuracy. Our results demonstrate the potential of combining machine learning and deep learning for stance detection in low-resource languages. This study not only introduces a novel dataset but also provides a robust evaluation of methods, highlighting the importance of modern AI techniques in processing informal and multilingual text data.

Keywords: stance detection, Roman Urdu, machine learning, SVM, random forest, logistic regression, naïve Bayes, decision tree, RoBERTa.

Submitted: 24 February 2025
Revised: 27 March 2025
Accepted: 27 April 2025
Published: 25 May 2025

Vol. 1, **No.** 2, 2025.
10.62762/TACS.2025.928069

***Corresponding author:**
✉ Khairullah Khan
khair@ustb.edu.pk
✉ Fida Muhammad Khan
fida5073@gmail.com

Citation
Khan, B., Khan, K., Khan, F. M., Noureen, H., Ali, A., & Shah, M. (2025). Comparing Fine-Tuned RoBERTa with Traditional Machine Learning Models for Stance Detection in Political Tweets. *ICCK Transactions on Advanced Computing and Systems*, 1(2), 78–96.

© 2025 by the Authors. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

1 Introduction

Social media platforms have emerged as a crucial component of communication on the internet, enabling quick streams of information to spread amongst users. This digital communication infrastructure has democratized information and provided citizen voices, while simultaneously creating a major challenge: a long overdue demand for computational tools to aid the automatic analysis and understanding of all content published and re-published on social media platforms.

Stance detection is the process of determining a person attitude or viewpoint toward a specific subject, claim, or entity. It involves analyzing written or spoken language. Stance detection is the process of analyzing written content in order to determine the writer stance or point of view. Identifying whether the writer agree, disagree, or is unrelated to the topic being discussed on social media. For the purpose to follow public opinion, analyze political debate, and challenge incorrect information, stance detection is essential. It helps with fact-checking, policy analysis, and tracking ideological trends by automating position classification. Researchers, decision-makers, and media analysts gain from this application since it enhances digital information integrity and offers insights into public opinion.

Popular online platforms such as Twitter, Facebook, and discussion forums serve as spaces for discussing incidents, topics, and expressing views. In this realm, stance refers to the opinion expressed by an individual on a specific topic, event, or public figure [1]. The internet has gradually become the primary means through which people obtain information and express their views. Consequently, understanding public opinion online has become a critical task for governments and businesses alike. Applications range from analyzing user attitudes on voting and consumer behavior to supporting rumor detection, making stance detection an active area of current research [2]. Extracting user intent from social media text—drawn from platforms such as Twitter and Facebook—enables downstream tasks including bot recognition, satire detection, and fake news identification. In this context, stance detection identifies whether a piece of text is sentiment-oriented or ideologically biased toward a given target [3]. Stance detection is a subtask of text analytics that involves categorizing a user's position expressed in textual data relative to a specific target or claim, rather than predicting demographic attributes such as gender or location. Essentially, it is the process of automatically determining how a writer positions themselves with respect to a given topic, accomplished by analyzing written text or a user's engagement on social debate sites and social network platforms. Stance can be understood as an outward public act in which a social actor, through communicative means, evaluates an object, positions themselves relative to others, and aligns with other speakers along sociocultural dimensions. There are many definitions of stance detection in sociolinguistics. Kockelman characterizes it as the performance of an evaluative attitude or judgment toward a proposition, whereby individuals align or disalign themselves with others. Factor-based studies have examined stance through individual dimensions such as linguistic features and lexical-aspectual classes. In the machine learning scenario, stance detection is all to a classification problem. Predicaments in this problem are traditionally represented using a pair of text and an action option (favor, against, and none) [4]. The stance classification process in Roman Urdu faces special difficulties due to both its code-mixed composition and phonetic variations and absence of standardized writing conventions. Traditional NLP methods perform poorly because different users spell the same word three ways ("mujhe," "mjhy," or "mjhe"). The lack of language standards requires unique preprocessing approaches and specialized embedding methods during feature extraction. A text normalization pipeline with improved accuracy is developed in this work to handle the challenges presented in stance classification. Roman Urdu is commonly used in the Indo-Pak subcontinent for messaging over the Internet since years. People from subcontinent may speak same language of Urdu, however they might be using different scripts for writing. The Roman characters of Urdu script for writing became very popular in this modern age. Large volume of Roman Urdu data is available on online portals and social media platforms like Facebook, Twitter that can be extracted to get useful information if the data gets structured. The best example of the people who speak roman Urdu using Roman alphabet. It is currently spoken in Pakistan and using it as a chat language [5].

A branch of artificial intelligence called machine learning makes use of algorithms and statistical models to enable computer systems learn and carry out tasks without explicit programming. Algorithms for machine learning allow computers to recognize patterns and learn from examples, without being

explicitly programmed making its performance improve time after time unaided by human action. These algorithms fit into a few different sub-categories supervised learning, unsupervised learning, and reinforced learning, depending on the type of input data and the anticipated results. Natural language processing, image and speech recognition, recommendation systems, and predictive analytics are just a few of the many domains in which machine learning realizes utilization. ML technologies are used in various utilities like social media, health care, finance sector etc., Auto drive cars, social media, natural language processing, cyber security and marketing etc. It has projected those industries into automation, predictive analytics, recommendations and decision making. As machine learning continues to evolve it will cover the way for all kinds of innovation and problem solving in other fields as well. A systematic review of machine learning techniques specifically developed for stance detection demonstrates that advances in natural language processing have substantially improved the automatic identification of attitudes and positions expressed in texts [6].

A major challenge faced early in the research was the absence of a comprehensive dataset of Roman Urdu political tweets. Unlike other languages or dialects, Roman Urdu lacks well-structured, and there have been no up-to-date datasets that can be readily utilized for training machine learning models. This limitation meant that we could not rely on pre-existing resources to develop models capable of delivering the desired results for stance detection. Moreover, there has been minimal to no research conducted specifically on stance detection in Roman Urdu tweets making this a relatively unexplored area. As a result, we had to dedicate considerable effort to constructing a suitable dataset.

In this research work, we scraped Twitter's Roman Urdu political tweets and comments via the Twitter API. Then we preprocessed the dataset by eliminating stop words, tokenizing, POS tagging, and then lemmatizing as well as TF-IDF for converting textual data into numeric format to make it machine understandable. After that we experimented extensively with several machine learning and deep learning models. By categorizing and evaluating the effectiveness of current techniques along several dimensions, such as input selection, feature extraction, classification algorithms and more, these studies cover the most recent advancements in stance detection. The goal of our research work is to identify and categories Roman Urdu political tweets and comments, indicating whether the comments are in agreement (agree), disagreement (disagree), or unrelated to the tweet posted on Twitter. AI-driven stance identification presents concerns about algorithmic bias, privacy, and possible use. To stop the spread of false information and political manipulation, it is crucial to guarantee data transparency, equitable representation, and responsible use. In order to ensure fair stance classification, this study emphasizes ethical aspects in NLP models. The objectives of this research are as follows.

- To gather political tweets and comments in Roman Urdu via the Twitter API.
- To compile a collection of political tweets and comments in Roman Urdu in order to identify stances.
- To build and implement a machine learning system to detect stance in Roman Urdu political tweets.
- To implement and fine-tune a RoBERTa-based deep learning model tailored for stance detection in Roman Urdu political tweets, leveraging its contextual understanding for precise classification of stance by comparing its result with machine learning model.

We have briefly introduced the concept of stance and demonstrating how its features are applied in computational tasks. Stance refers to the position or attitude a speaker or writer takes toward a specific topic, expressed through text. Stance has been described in sociolinguistics as a publicly performed act through which a social actor evaluates objects, positions themselves relative to others, and aligns with fellow speakers across sociocultural dimensions [1].

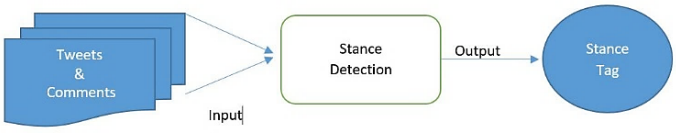


Figure 1. General procedure of stance detection.

As depicted in Figure 1, stance detection is modeled as a text classification problem. The process typically involves analyzing tweets and their accompanying comments, which are passed through a classification algorithm designed to categorize them based on the stance expressed, such as "agree," "disagree,"

or "unrelated." This schematic outlines the general approach to stance detection, emphasizing the role of algorithms in automatically assigning stance tags based on input features extracted from the text. We have further defined and distinguished different types of stance detection tasks through formal definitions showing how this applies across various real-world scenarios.

This paper's remaining sections are organized as follows. Within Section 2, we have provided a broad examination of the related work in terms of definition and task types. Section 3 presents the whole picture of the stance detection framework from data source, data acquisition, and preprocessing to annotation, designing a stance extraction algorithm, etc., and the main steps are introduced. In Section 4, we explore the core problem of the framework by categorizing the key methods, examining the underlying principles of each, comparing their performance, and evaluating the accuracy of all classifiers. Finally, Section 5 concludes the paper, offering insights into future trends.

2 Related Work

In this section, an exhaustive literature review of the research done in text mining is presented concerning some novel areas to build a system for stance detection in Roman Urdu tweets on Twitter. Stance is defined as the machine learning-and deep learning-based categorization of tweets into groups like agree, disagree, or unrelated. The particular objective of this study is to assess the existing detection techniques, looking at their advantages and disadvantages as well as potential areas for development. This study aims to lay a strong foundation of past research on stance detection in Roman Urdu online communities, which will ultimately lead to improved performance.

Küçük and Can [7] provided a Turkish language Twitter dataset annotated with named entity identification (NEI) and stances. This dataset is two sub-problems of NLP which are stance detection and named entity recognition. While initially only the stance annotations were released, both named entity and stance are now available online. This dataset enables potential links to be investigated with named entity recognition and stance detection in tweets.

Walker et al. [8] conducted stance categorization on a corpus of 4731 posts from the discussion website ConvinceMe.net for 14 themes, ranging from humorous to ideological. They demonstrate that the number of response postings in ideological disputes is higher, and that both people and trained classifiers find it much more difficult to identify stances in rebuttal replies. Additionally, they have shown that the quantity of subjective phrases fluctuates between arguments, a finding that is connected with how well computers perform when exposed to terms that carry sentiment. They show findings ranging from 60% to 75% for categorizing stances according to topic, in contrast to 47% to 66% for unigram baselines. According to their findings, accuracy is increased by features and techniques that consider the dialogic context of these posts.

Yan et al. [9] developed an entirely novel quick stance identification system for bipolar affinities on Twitter using deep learning. The 2016 US presidential election campaign generated millions of messages on Clinton and Trump every day on Twitter, which is therefore it is utilized as a test use case due to its remarkable and distinctive counterfactual characteristics. Furthermore, stance detection can be used to infer the public's political tendency. Comparing their research to a number of other stance identification techniques currently in use, experimental results demonstrate that their approach generates excellent accuracy outcomes.

Similar to sentiment analysis, stance detection is concerned with analyzing the sentiments of a user about some pieces of text. The goal of stance categorization is to automatically determine whether the source material favors, opposes or is neutral toward a target. Ayyub et al. [10] a study of the performance investigation on machine learning algorithms including deep learning based (DBN, CNN-LSTM and RNN), ensemble based (RF, AdaBoost) and classical one (NB. DT., SVM). There work was a feature-based analysis on sentiment, content, tweet-specific and part-of-speech features using the SemEval2016 and Sentiment Analysis in Twitter datasets of deep learning based RNN, CNN-LSTM and DBN. They also studied for stance categorization the quality of deep features such as GloVe and Word2Vec. The baseline features (Bag of Words, Ngram and TF-IDF) are also retrieved from the two dataset that facilitates a comparison with their deep based method. The results are also compared with existing, state-of-the-art performance evaluation metrics for ensemble learning-based stance classification upon the prior research. The evaluation results suggested that deep feature GloVe works best when used in conjunction with RNNs, and the optimal combinations of sentiment and part-of-speech content (for SVM), played an essential role in stance

classification.

The electronic media is a good thing. News circulates on online social networks because it is cheap, easy to find and delivered rapidly. Karande et al. [11] proposed a credibility analysis model for social media that leverages stance detection with BERT embeddings to assess information veracity. Their approach demonstrates that stance, used alongside article content and contextual word embeddings, achieves an accuracy of 95.32% in distinguishing credible from potentially misleading content.

Stance detection is a level up process from sentiment analysis, as it features the extent to which an author is leaning on some aspects of interest like person, event or government policy, favorable or against about the positions. Skanda et al. [12] explored stance detection in a code-mixed English–Kannada dataset composed of Facebook user comments. The task was framed as a binary classification problem with two labels: "against" and "favor." The dataset contained approximately 6,300 phrases distributed across both classes. To develop the stance detection system, the authors employed a variety of text representation techniques—including bag of tricks, word embeddings, Word2Vec, and GloVe—as well as deep learning architectures such as CNN and Bi-LSTM. Interestingly, the simpler "bag of tricks" approach outperformed the more complex deep learning models.

With regard to one or more target entities, the "stance detection" application on Twitter examines individuals' opinions in their tweets. A neural ensemble model that combines the strengths of two LSTM variations is presented by Siddiqua et al. [13] in order to produce better long-term associations. An attention mechanism is integrated into each module to enhance the contribution of important components to the final representation. Additionally, they use a multi-kernel convolution on top to extract the tweet representations at higher levels. The primary innovation of the unified model is its efficient contextual information learning capacity, which enhances the performance of stance detection and exceeds the most advanced deep learning-based techniques for both single and multi-target stance detection benchmark datasets. A thorough analysis of numerous single- and multi-target stance identification datasets shows that their suggested approach offers a notable enhancement over the most advanced deep learning-based techniques currently available.

Tian et al. [14] investigated stance classification as a transfer learning task in the context of Twitter, proposing a framework that learns stance representations from user comments without manual annotation. Their approach trains CNN and BERT-based models via transfer learning on external data sources, enabling automatic stance categorization based on user attitude distributions and tweet content. The study demonstrates that stance representations learned through transfer learning can generalize effectively across different tweet scenarios, offering valuable insights for stance classification in low-resource settings.

Kochkina et al. [15] proposed a sequential model based on long short-term memory (LSTM) for stance classification in Twitter conversations, achieving an accuracy of 0.784. Their branch-LSTM architecture captures the conversational thread structure of tweets, modeling the sequential dependencies between posts to improve stance classification performance. Although their label scheme differs from the agree/disagree/unrelated framework adopted in this study, their work demonstrates the effectiveness of sequential deep learning architectures in capturing stance-relevant contextual information across multi-turn social media interactions.

The classification of texts is a tool that divides text into predefined categories using supervised machine learning algorithms. The classification of texts is a tool that divides text into predefined categories using supervised machine learning algorithms. Usman et al. [16] applied five well-known classification methods on an Urdu language corpus and combined their outputs using a majority vote approach. The corpus contains 21,769 news articles spanning seven categories. Since the algorithms could not directly process raw text, several preprocessing steps were applied, including trigram extraction, stop word removal, and rule-based stemming. After preprocessing, 93,400 features were extracted for model training. The majority voting strategy achieved a recall and accuracy of 94%.

In any argument, participants may have different viewpoints on the topic at hand. A user can agree or disagree on the matter being debated. A user's viewpoint is defined as how much they agree with a topic. Küçük and Can [17] conducted a comprehensive survey of stance detection methods, covering feature-based approaches including SVM classifiers with diverse feature sets such as unigrams, bigrams, hashtags, and named entities. Their

analysis demonstrated that incorporating named entity recognition (NER) features can significantly enhance stance detection performance on social media texts.

Stance identification, which aims to determine whether a writer's attitude toward a specific topic is neutral, supportive, or opposed, plays a crucial role in analyzing significant social and political events. A notable contribution in this field is the P-STANCE dataset [18], a large-scale collection of 21,574 annotated political tweets gathered during the 2020 US presidential election. The dataset paper develops deep learning baselines on P-STANCE, with BERTweet achieving a state-of-the-art macro-average F1-score of 80.53%, further improved through semi-supervised learning. The P-STANCE dataset also facilitates cross-target stance detection research, and three additional publicly available Twitter datasets with stance annotations were released alongside it, enabling research across various political domains. Separately, Küçük and Can [19] developed an SVM-based approach to stance detection on tweets, employing a diverse feature space including lexical features (unigrams and bigrams), social media-specific elements (hashtags and external links), emotional cues (emoticons), and named entities. Their experimental results demonstrated that the joint use of unigrams, hashtags, and named entity features within an SVM framework yields the most effective performance, providing valuable insights for feature selection in tweet-based stance detection systems.

The "stance detection" application on Twitter analyzes the opinions users have about one or more target entities in their tweets. Lan et al. [20] propose a novel framework called COLA (Collaborative Role-Infused LLM-Based Agents) for stance detection, which leverages multiple large language model (LLM) agents, each assigned a distinct analytical role. Designed primarily for English-language social media and web content, COLA facilitates nuanced stance analysis through a three-stage pipeline: multidimensional text analysis, reasoning-based debate among agents, and final stance conclusion generation. The authors evaluate COLA on benchmark datasets including SEM16, P-Stance, and VAST. Experimental results show that COLA outperforms traditional stance detection methods, achieving accuracy rates of 76.5% and 74.1% on P-Stance and VAST, respectively. Moreover, COLA demonstrates strong explainability and user interpretability, making it particularly effective in zero-shot learning scenarios. Mets et al. [21] investigate stance detection in Estonian immigration discourse, a representative case of low-resource language processing. The study introduces a corpus of 7,345 manually annotated sentences categorized into "support", "against", and "neutral" stances. Two primary approaches are examined: fine-tuning BERT-based models and prompting large language models (LLMs) such as GPT-3.5. Experimental results show that both approaches—especially prompt-based zero-shot methods—achieve strong performance without task-specific training, demonstrating their utility in low-resource scenarios. The findings highlight how modern LLMs can effectively support both automatic analysis and manual annotation processes, paving the way for improved stance classification in underrepresented languages. Wang et al. [22] presented Dynamic Experienced Expert Modeling (DEEM) to improve large language models (LLMs) reasoning abilities through expert simulation. DEEM uses trained data to create expert potentials, then selects knowledge sources based on frequency counts and correct outcomes, before dynamically retrieving relevant experts during inference. Three standard datasets—P-Stance, SemEval-2016, and MTSD—were used to evaluate the proposed method. DEEM consistently outperformed existing methods including those utilizing self-consistency reasoning, achieving an average F1 score of 81.1 on P-Stance and 83.4 on MTSD. The implementation utilized InstructGPT and ChatGPT models, demonstrating DEEM's adaptability.

Stance detection tasks demonstrate robust behavior through this literature review of approaches that work with multiple languages across different domains and independent models. Low-resource languages like Roman Urdu are still understudied despite advances in stance detection because of a lack of datasets, inconsistent writing styles, and limited computing resources. By utilizing deep learning techniques and presenting a labeled dataset, this study solves these issues and advances the field of stance classification research in code-mixed and low-resource Language. The most recent research on Roman Urdu NLP have been included in related work, with an emphasis on developments in stance identification, sentiment analysis, and low-resource language processing. These studies show the shortcomings in Roman Urdu stance detection and give our findings important background. In our research, a new approach is proposed for stance detection in Roman Urdu tweets to make it more

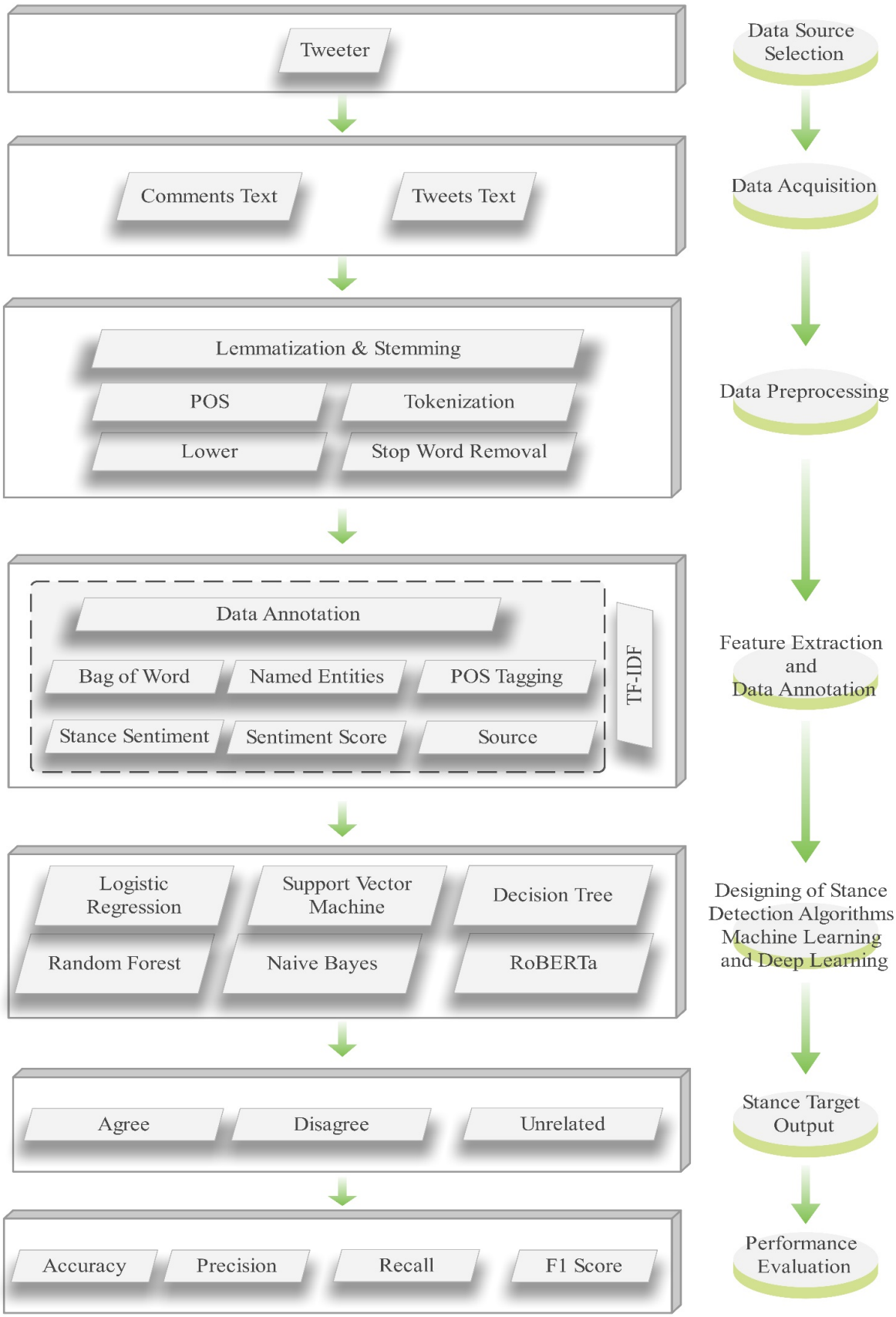


Figure 2. Framework for stance detection.

effective by employing machine learning algorithms. Our main objective is to create a dataset of political tweets and related comments collected through the Twitter API as initial. After this, we applied text preprocessing techniques including tokenization, stop word removal, lowercasing, and lemmatization. We have selected feature extraction techniques like TF-IDF and trained multiple machine learning classifiers such as logistic regression, random forest, support vector machine (SVM), decision tree, naïve bayes classifier, and RoBERTa, a deep learning model, to detect stance in Roman Urdu political tweets. These model results were evaluated for precision, accuracy, recall, and other performance metrics. Recent studies have shown that transformer-based architectures, including XLM-RoBERTa, T5, and GPT-based models, significantly enhance stance detection accuracy in multilingual and code-mixed settings. Their advanced contextual understanding makes them particularly well-suited for low-resource stance classification tasks.

3 Methodology

A general framework is suggested in this part as shown in Figure 2. We explain a brief overview of the core steps in this framework. The steps are data collection, data preprocessing, feature extraction and annotation, stance detection algorithm design, stance tag output, performance assessment, and data source selection.

3.1 Selecting and acquiring data sources

The entire textual data has been concatenated (tweets and related comments) are obtained via the Twitter API. There are also many other aspects to take care of in terms of the dataset quality and trusting it wholly. Hence, it was very carefully and thoroughly drafted to include a variety of Roman Urdu tweets. The details maintained during this process to ensure a complete corpus identical in lexical distribution also serve to reduce biases and maintain the integrity of the corpus.

3.2 Dataset

Tweets were gathered from a variety of political sources, such as the media, independent individuals, and official party accounts, in order to avoid dataset bias. By ensuring a balanced representation of political positions, a class distribution study decreased the possibility of skewed categorization results. We first created a developer account and applied for API access in order to create a Tweets dataset. Once given access, we generate the required API keys (API key, API secret, access token, and access token secret) and install necessary Python libraries, including Tweepy for interacting with the Twitter API and Pandas for manipulating data. We then authenticate the API keys using Tweepy. Client and specify pertinent political hashtags or keywords, such as the names of political parties or Pakistani political leaders. (e.g., "#ImranKhan", "#NawazSharif", "#PPP", "#PMLN", "#PTI"). The tweets are saved in a data frame using pandas, where we have further processed them for our research.

To maintain high-quality stance annotation, we established comprehensive labeling guidelines that clearly define the criteria for agree, disagree, and unrelated stance categories. The dataset was annotated by three independent annotators, with any discrepancies resolved through discussion. We computed the inter-annotator agreement to assess annotation consistency, which demonstrated substantial agreement. This rigorous approach minimizes ambiguity and enhances the dataset's reliability for stance classification.

To assess the robustness and generalizability of our model, we evaluate its performance on an independent dataset collected from Twitter. The deep learning model achieves an accuracy of 97%, indicating its ability to maintain consistent classification performance across diverse social media contexts. This evaluation highlights the model's effectiveness in handling variations in language, topic, and user-generated content, reinforcing its applicability to real-world stance detection tasks.

Moreover, we expand the dataset by adding features such as Stance Sentiment, Sentiment Score, Named Entities, Bag of Word Features, Source and Dependency Parsing Features, and Stance Target (Agree, Disagree, or Unrelated). This aims to capture the intricate web of linguistics present in Roman Urdu social media discourse. In this research, we have created a gold standard dataset and laid the foundation to perform extensive analysis on stance dynamics in Roman Urdu tweets with meticulous execution of different stages involved in creation. A summary of the compiled Roman Urdu political tweet dataset is shown in Table 1, which highlights the total number of entries, language, source, and stance labels. The detailed structure of the Roman Urdu stance dataset, including tweets, comments, and extracted linguistic features, is presented in Table 2.

Table 1. Summary of dataset.

Total Rows	Total Column	Language	Source	Stance Label
8374	11	Roman Urdu	Twitter	Agree, Disagree, Unrelated

Table 2. Roman Urdu stance dataset.

S.No	Tweet ID	Tweets	Comment	Bag-of-Wc	Named En	POS Tags	Dependen	Source	Stance (Ta)	Stance Ser	Sentiment Score
1	2901	@BBhutto	@User78	@BBhutto	@BBhutto	Noun(@B	Root: dem	Twitter	Disagree	Negative	-1
2	2760	@ImranK	@User123	@ImranK	ImranKha	Noun(Cor	Root: COV	Twitter	Disagree	Negative	-1
3	3522	"@Maryam	"#Maryam	PMLN	PMLN	Noun(PM	Root: kiya	Twitter	Agree	Positive	1
4	2925	@Maryam	@User567	@ImranK	@ImranK	Noun(@l	Root: logo	Twitter	Disagree	Negative	-1
5	2954	@BBhutto	@User78	@BBhutto	BBhuttoZa	Noun(Edu	Root: Edu	Twitter	Disagree	Negative	-1
8372	4529	@NawazS	@ZainabK	zindagi	sa zindagi	Noun(zin	Root: Wel	Twitter	Agree	Positive	1
8373	7520	@Profess	@User292	Zindagi	re Zindagi	Noun(Zin	Root: karn	Twitter	Unrelated	Neutral	0
8374	4549	@NawazS	@ZainabK	zindagi	sa zindagi	Noun(zin	Root: Wel	Twitter	Agree	Positive	1

3.3 Data preprocessing

The goal of preparing data is to clean or organize the data such that its readability, analyzability, and understandability make it much more learner-friendly. There are the following subtasks in data preprocessing we have performed.

- D*ata cleaning and tokenization:* it is a preprocessing step for NLP, or generally for data analysis. Data cleaning is the process of removing noise, handling missing values, and correcting errors in a dataset so that it becomes normalized, tokenization means breaking the provided words into smaller units, called tokens in many cases. These steps together convert raw data into clean, structured data fit for analysis or modelling.
- *Stop word removal:* Stop word removal in Roman Urdu eliminates meaningless words from the text, such as 'aur,' 'ya,' 'ki,' 'ke,' 'tha,' and 'uska.' This technique reduces noise and enhances data quality by ignoring unimportant terms.
- *Part of Speech Tagging:* This process assigns a part of speech, e.g., a noun or verb, to each word in the text. This is important for grammar and used by some parsing tasks or machine translation. There are rule-based, statistical, and machine-learning applications for POS tagging.
- *Lowercasing:* The task of lowercasing a dataset is simply converting all the text into lowercase. This is done to make the text data look alike and clean. Allowing the text to be all the same case uniformly, without capitalization, simplifies it for further splitting of input into meaningful parts, which we can use for analysis.
- Lemmatization and Stemming: Keywords are often reduced to their root or base form, which can be facilitated by lemmatization and stemming, these are two techniques used in NLP. When stemming, the prefixes and suffixes are removed from a word to determine what it consists of by contrast, lemmatization essentially depends on word morphology and thus invites consideration of context relying on grammar rules. Roman Urdu users employ inconsistent spellings due to phonetic-based writing, leading to variations such as "nai," "nahi," and "nhe" (meaning "no"). To address this, we standardize text by implementing spelling correction rules and phonetic mappings. Our approach ensures that variations of the same word are normalized, improving feature consistency and classifier performance. In addition, it may go without saying that one lemma varies greatly because it is named for it to be pointed out in the singular or plural case.
  Word: "khana" (to eat) Stemmed: "kha" Explanation: The stem of "khana" is "kha".
  Word: "chaliye" (let's go) Lemmatized: "chalna", Explanation: The lemma of "chaliye" is "chalna", which is the infinitive form of the verb "to go".

3.4 Designing of stance detection algorithm

Creating a model that can recognize the stance stated in the comments toward the tweet text, such as agree, disagree, or unrelated, is a necessary step in designing a stance identification method. Preprocessing, feature extraction, data collection, and the selection of appropriate machine learning and deep learning algorithms are usually steps in this process, which are explained in Section 4.

3.5 Feature extraction & adding additional features to dataset

After preparing a dataset, additional features are included in the dataset, data annotation, and inclusion of further features. Feature extraction is concerned with selecting the essential characteristics from the provided unprocessed data and consequently lessening the dimensionality of the dataset so the model can more easily identify meaningful patterns. The next step involves data annotation, which refers to the labeling of data with relevant tags or categories, which assists the model in identifying various factors. It is critical to ensure that data is accurately labeled because it will severely affect the output. Further adding of more features could assist in enhancing the model's formative capabilities by enabling it to identify patterns and connections. Therefore, all these steps contribute to turning unprocessed data into appropriate machine-learning inputs.

3.6 Stance tag output and performance evaluation

It contains the stance target labels (agree, disagree, or unrelated) that are predicted by applying an algorithm for every input text and evaluating on how accurate this process is. The output of the stance tag is the final result, which classifies an algorithm and suggests how it detects stances for each piece of text. We conduct the performance evaluation based on accuracy, precision, recall, and F1-score to illustrate how well an algorithm is performing. We have used it to establish the trustworthiness of an algorithm, as well as to help identify avenues for improvement our stance detection model can correctly and repeatedly classify stances across texts.

Performance evaluation: Model testing checks various classifier's performance to check the most effective classifier for a given dataset.

Confusion Matrix: The confusion matrix is a basic tool for assessing the adequacy of the classification model. It is a tabular representation of how the model predictions compare with the true/actual class labels from the training set and predicted labels of each class.

Accuracy: This is a measure of the general correct classification of the model prediction. It is defined as the ratio of the number of correct predictions to the total predictions on a dataset. While low accuracy suggests the model might be incorrectly identifying some of the occurrences, high accuracy indicates accurate prediction in all classes. The accuracy calculation formula is provided below.

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \tag{1}$$

Precision: The ratio of correctly predicted (positive instances) to all of the model's predicted (positive instances) is known as precision. It demonstrates the model's ability to avoid misclassification and false positives. The precision calculation formula is given below.

$$\text{Precision} = \frac{\text{correctly predicted instances (predicted class labels)}}{\text{total predicted instances (predicted class labels)}} \tag{2}$$

Recall: The true positive rate or recall or sensitivity is equal to the total number of actual classified (actual class labels) positive cases over the sum of difference of the total number of real class label positive cases over the total number of actual classified (actual class labels) positive cases. It is shown that the model can identify positive instances with high accuracy and ignores incorrect negative instance class detections. In the case of a model with high recall, it is highly likely to catch the majority of the positive instances, and low recall means it is most likely to miss a large percentage of positive instances. Recall can be determined using the following formula.

$$\text{Recall} = \frac{\text{correctly classified instances (correctly classified class labels)}}{\text{total actual classified (actual class labels)}} \tag{3}$$

F1 Score: The F1 score is the harmonic mean of precision and recall—the balance of the two. However, F1 score takes into account both false positives and false negatives in particular, hence it's the right metric to evaluate unbalanced dataset. The F1 score is achieved when the model has the maximum evaluation of false positives and false negatives by incorporating high precision and recall. The F1-score calculation formula is below.

$$\text{F1-Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

4 Result and discussion

4.1 Tool and language

The cloud-based platform we utilized, Google Colab (short for Google Colaboratory), enables users to develop and run Python code through the browser itself. Featured free virtual machines with libraries and frameworks most commonly used in machine learning and data analysis (TensorFlow, PyTorch Scikit-Learn).

In general, it is a nice and easy way to use some of the most popular tools for collaborative programming in data science and machine learning projects.

The research work has been done in Python language, well recognized as a powerful and efficient programming language due to its simplicity and ease of writing. It is employed extensively in different areas like web development, data analysis, and AI. Python is most likely a first option among developers because of its ease of syntax and vast library support for building many different types of applications.

4.2 Dataset splitting

Our dataset was separated into testing and training sets. By evaluating the results, the model is assessed on the testing set.

Training Data: 80% of the data used to train the machine learning and deep learning models is included in this slice of the dataset.

Testing Data: Finally, the remaining 20% of the data set is built from completely independent data and is used to evaluate how well the finished model performed after training. This method will reliably and impartially assess the model's capability to forecast and generalize untested new data. So, it ensures that the model is able to do the stated task of forecasting in realistic and useful situations.

4.3 Applying machine learning classifiers for model training

We build our model by training it with machine learning classifiers. We then proceed to apply machine learning classifiers that classify stances based on input features so as to solve different types of real-world problems across multiple domains. Which learn patterns and relationships in data that contain labels in datasets. These classifiers have their own strengths and weaknesses, as well as freedom in data type related to the task. Usually, it consists of several stages, such as model training, hyperparameter modification, and model evaluation using various metrics, such as accuracy, precision, recall, F1 score, etc.

4.3.1 Applying logistic regression

Logistic regression is a popular supervised machine learning statistic and statistical classification method. Based on the sigmoid or softmax function trying to find the relation between input features and target classes, it predicts the probability that the observation will belong to some certain class. Relatively, it is simple, efficient, and easy to understand, and commonly used because of it. Logistic regression is used in the context of using the tweets and comments in Roman Urdu political tweets and comments in order to classify the tweet and comment into one of three stance categories: "Agree," "Disagree," and "Unrelated." The technique works by employing TF-IDF technique to extract features from Roman Urdu text and have the likelihood of the comment being of each posture class. Below is the formula for the model's forecast, which is tied to the highest probability class.

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-z}} \tag{5}$$

where:

- $Z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$
- $P(y = 1 \mid X)$ gives the probability that the output y belongs to class 1 ("agree") given the input features X.
- Z is the linear combination of the input features X weighted by their corresponding coefficients β.
- β_0 is the intercept (bias), which acts as a baseline probability.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the learned weights for each feature. These values are learned during training to minimize the classification error.
- $X_1, X_2, \ldots, X_n$ are the input features (e.g., TF-IDF vector values).

The prediction was calculated for the class with the highest probability.

Classification Result of Logistic Regression: Performance of the model on Agree, Disagree, and Unrelated across precision (positive predictions), recall (actual positives), and F1 Score. This includes the support (number of actual instances per class), overall accuracy, and macro & weighted averages for aggregate metrics. As shown in Table 3.

4.3.2 Applying decision tree classifier

Decision tree is a popular supervised machine learning approach since the interpretability and user friendliness are obvious for classification and regression tasks. It builds a tree like structure representing decisions and various outcomes of each decision. The tree is constructed with three key components: internal nodes, branches, and leaf nodes. An internal node is a feature (or attribute) from the dataset over which to decide, whereas an internal node

Table 3. Classification result of logistic regression.

Class	Precision	Recall	F1-score	Support
Agree	0.91	0.91	0.91	583
Disagree	0.93	0.95	0.94	602
Unrelated	0.96	0.93	0.94	490
Accuracy			0.93	1675
Macro Avg.	0.93	0.93	0.93	1675
Weighted Avg.	0.93	0.93	0.93	1675

represents such feature. Internal nodes have branches that are possible outcome or condition in terms of that feature. The leaf nodes finally correspond to the output value, which most often is the prediction or final value in the case of classification problems, or the final value for regression.

In stance detection, Roman Urdu political tweets and comments are analyzed using Decision Tree to identify stance of comments regarding a target tweet into three categories agree, disagree and unrelated. This can be achieved by splitting the data iteratively based on the features derived from the text: specifically, words, phrases and their TF-IDF indexes (scores). It proceeds by taking in turn, and for each step, choosing the feature and splitting condition which best discriminates that data into purer subsets, where purer means mainly having instances of one stance class. Suppose the keyword in question is present in a comment, then it is an indication of agreement, and its absence is either an indication that the author disagreed on the topic or thought otherwise. While this procedure continues until the tree stops at some stopping condition like minimum number of samples per node or maximum depth.

Entropy:

$$\text{Entropy}(S) = -\sum p_i \log_2(p_i) \tag{6}$$

Information Gain:

$$IG(S, A) = \text{Entropy}(S) - \sum \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v) \tag{7}$$

Split on the feature with the highest information gain (IG). Stop at the maximum depth or when the minimum number of samples is reached. Each leaf node predicts the majority stance among the samples it contains (i.e., *agree*, *disagree*, or *unrelated*).

Classification Result of Decision Tree Classifier: The performance metrics that have been calculated by a decision tree after it predicts the class labels for a test dataset are its classification results. Precision, recall, F1-score, and accuracy of these metrics is an indicator of how well the model classifies the instances as per 3 categories. Every metric measures this aspect of the model's performance and, in a rigid sense, there is no competition between them. For the dataset with the three classes (Agree, Disagree, and Unrelated) the Decision Tree model classification performs very well in all the metrics. With an accuracy of 92% (i.e. 92 out of all the cases in the dataset were correctly classified by the model), the model indicates. The classification results of the Decision Tree model in terms of precision, recall, and F1-score for each stance class are shown in Table 4.

Table 4. Classification result of decision tree classifier.

Class	Precision	Recall	F1-score	Support
Agree	0.91	0.90	0.91	583
Disagree	0.94	0.93	0.93	602
Unrelated	0.92	0.94	0.93	490
Accuracy			0.92	1675
Macro Avg.	0.92	0.92	0.92	1675
Weighted Avg.	0.92	0.92	0.92	1675

4.3.3 Applying support vector machine (SVM) classifier

Support Vector Machine (SVM) is a very powerful supervised machine learning approach that works best for either classification or regression applications. It is particularly suited for high dimensional data and finds the best hyperplane for the data points to be divided into separate classes. The key is for SVM to maximizes the margin, i.e. the separation of the hyperplane and the closest support vectors (data points) from each class. The generalization and classification accuracy are improved by maximizing this margin.

SVM operates to classify Roman Urdu tweets into the three categories, agree, disagree and unrelated, in our research for stance detection. The algorithm works on the input features created using TF-IDF etc., and converts them into a high dimensional space. It provides a decision boundary that most well separates the stance categories. This is mainly due to SVM's capability to handle complex data distribution even in

Table 5. Classification result of support vector machine.

Class	Precision	Recall	F1-score	Support
Agree	0.93	0.90	0.92	583
Disagree	0.93	0.96	0.95	602
Unrelated	0.94	0.94	0.94	490
Accuracy			0.94	1675
Macro Avg.	0.94	0.94	0.94	1675
Weighted Avg.	0.94	0.94	0.94	1675

nonlinear scenarios; thus, it is a reliable choice over text classification. SVM makes use of kernel functions to adapt to the details of Roman Urdu text, dealing with linguistic diversity and minor variations in stance. Its scalability and precision at classifying nuanced textual data makes it a useful stance detection tool in political tweets because classified agree, disagree or unrelated sentiments are important to detect.

Objective Function:

$$\min \frac{1}{2}\|w\|^2 \quad \text{subject to } y_i(w \cdot x_i + b) \geq 1 \tag{8}$$

Kernel Function:

$$K(x_i, x_j) = \theta(x_i) \cdot \theta(x_j) \tag{9}$$

Decision Function:

$$F(x) = \text{sign}(w \cdot x + b) \tag{10}$$

Classification Result of Support Vector Machine: A Support Vector Machine yielded these numbers to demonstrate its testing set performance through accuracy, precision, recall, and F1 score measures. Table 5 demonstrates the scores acquired by the SVM classifier when used to predict stances in Roman Urdu tweets.

4.3.4 Applying random forest classifier

Our research made use of Random Forest algorithm as a flexible ensemble learning technique for classification problems. The training process of Random Forest includes building multiple decision trees which collectively generate combined results to boost accuracy levels. The final classification decision combines the predictions of individual trees through majority voting from all constructed trees. The combination of multiple decision trees using this approach leads to better generalization performance and enhances the overall robustness for the model.

We used Random Forest to analyze Roman Urdu text for stance detection by training multiple decision trees along with features based on TF-IDF values and word frequencies to determine whether tweets agreed or disagreed with the topic. All decisions made by singular trees combine their predicted classifications through majority voting to determine the final stance for individual tweets. This method creates highly accurate stance predictions.

Entropy:

$$\text{Entropy}(S) = -\sum p_i \log_2(p_i) \tag{11}$$

Information Gain:

$$IG(S, A) = \text{Entropy}(S) - \sum \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v) \tag{12}$$

Final Prediction:

$$\text{Prediction} = \text{Majority Vote}(\text{Tree}_1, \text{Tree}_2, \ldots, \text{Tree}_n) \tag{13}$$

where Tree_i is the prediction from the i-th decision tree.

Classification Result of Random Forest:

The classification result of the Random Forest classifier provides a classification report that gives us an exhaustive detail of how well the model performed in terms of precision, recall, F1-score and support for each class. Results are presented in Table 6.

Table 6. Classification result of random forest.

Class	Precision	Recall	F1-score	Support
Agree	0.96	0.92	0.94	583
Disagree	0.94	0.96	0.95	602
Unrelated	0.94	0.96	0.95	490
Accuracy			0.95	1675
Macro Avg	0.95	0.95	0.95	1675
Weighted Avg	0.95	0.95	0.95	1675

4.3.5 Applying naïve bayes classifier

The probabilistic supervised learning algorithm Naïve Bayes Classifier functions as our model for classification work. The Bayes theorem calculates class probability using features while making the assumption that each feature remains independent from other features. Naïve Bayes offers an excellent text categorization solution because it remains effective even though the independence assumption does not precisely hold true therefore serving as a good choice for high-dimensional data processing. We employ Naïve Bayes to identify Roman Urdu text stances which leads to classification into the three categories of "Agree," "Disagree," or "Unrelated." An evaluation of probabilities shows which stance class

Table 7. Classification result of naïve bayes classifier.

Class	Precision	Recall	F1-score	Support
Agree	0.84	0.89	0.86	583
Disagree	0.90	0.87	0.89	602
Unrelated	0.94	0.91	0.93	490
Accuracy			0.89	1675
Macro Avg.	0.89	0.89	0.89	1675
Weighted Avg.	0.89	0.89	0.89	1675

is most likely based on the selected features which include word frequencies or TF-IDF values within the algorithm. The combination of prior probabilities and class-related likelihood distributions supports Naïve Bayes in executing quick and efficient tweet classification based on stance.

Bayes' Theorem:

$$P(C_k \mid X) = \frac{P(X \mid C_k) \cdot P(C_k)}{P(X)} \tag{14}$$

Classification:

$$\hat{C} = \arg\max_{C_k} P(C_k) \prod_{i=0}^{n} P(x_i \mid C_k) \tag{15}$$

By applying Naïve Bayes, the stance detection classification achieves quick and reliable classification of Roman Urdu tweets into meaningful categories, contributing to improved analysis of user-generated content.

Classification Result of Naïve Bayes Classifier:

The Naïve Bayes Classifier's classification result is a gauge of how successfully it labels dataset occurrences. The key metrics for predicting ability accuracy, precision, recall, and F1-score are shown in Table 7.

4.4 Applying deep learning model

The advanced artificial intelligence method deep learning depends on networks of artificial neural layers to manage and analyze large datasets. The automatic analysis of raw input data through deep neural networks facilitates their designation as deep neural networks which lack requirements for manual feature extraction procedures. Deep learning's method based on human brain architecture effectively completes complex pattern identification and abstraction together with autonomous decision-making tasks and natural language processing as well as picture and speech recognition and predictive analytic tasks. Deep learning solves real-world problems accurately through its nonlinear pattern recognition abilities across multifaceted datasets. The innovation in various domains including healthcare and finance and robotics as well as others depends on deep learning through the usage of large-scale datasets along with contemporary technology frameworks comprising GPUs and TPUs. Although various deep learning models were studied, RoBERTa was selected due to its optimized architecture and superior performance in stance classification. Models such as BERT and LSTMs were excluded as they exhibited lower accuracy on Roman Urdu text and required higher computational resources. Future research will focus on exploring alternative transformer-based architectures to further enhance classification accuracy and efficiency.

4.4.1 Dataset augmentation

Data augmentation is a technique used to artificially increase the size and diversity of a dataset in machine learning and deep learning. For text data, common augmentation strategies include synonym replacement, back-translation, random insertion or deletion of words, and paraphrasing. These transformations preserve the underlying semantic meaning and stance labels while introducing lexical variation, improving model generalizability when the initial dataset is limited in size or class diversity. A comprehensive analysis of political stance distribution was conducted to ensure equitable representation of diverse political perspectives. To mitigate bias, data augmentation and class balancing techniques were implemented, preventing the overrepresentation of any single political group. These measures enhance the fairness and reliability of stance classification models.

4.4.2 Applying RoBERTa Model

A powerful natural language processing model called RoBERTa (A Robustly Optimized BERT Pretraining Approach) is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture. It improves upon BERT by optimizing the training procedure, using larger datasets, and removing certain constraints from the original design. RoBERTa employs dynamic masking, where the masked tokens in the input are changed during each training epoch, enhancing its ability to generalize. Additionally, it removes the Next Sentence Prediction (NSP) assignment, concentrating only on the goal of masked language modeling to enhance contextual awareness. With larger batch sizes, extended training steps, and diverse pretraining corpora, RoBERTa achieves state-of-the-art performance across

numerous NLP tasks, such as text classification, sentiment analysis, and question answering. This robust model highlights the importance of fine-tuning pretraining strategies to maximize the potential of transformer-based architectures.

We fine-tuned the RoBERTa model on our Roman Urdu political tweets dataset to classify stance into **agree**, **disagree**, or **unrelated** categories. The model robust pretraining and ability to capture contextual nuances in text allowed it to effectively handle the informal and code-mixed nature of Roman Urdu. By incorporating dataset augmentation, we achieved an impressive accuracy of 97%, demonstrating RoBERTa's exceptional performance in stance detection tasks.

A comparative analysis with recent stance detection studies reaffirms that transformer-based models consistently outperform traditional classifiers in stance classification. Our findings indicate that RoBERTa achieves a notable improvement in precision and recall, surpassing previous approaches in Roman Urdu stance detection. Specifically, RoBERTa attains its enhanced ability to capture contextual nuances and complex linguistic patterns, making it a highly effective model for stance classification tasks. The working mechanism of the RoBERTa model is defined as follows.

- Tweet: @MaryamNSharif: "PMLN ne sports ko promote karne ka irada kiya hai."
- Comment: @BilawalBZardari: "@MaryamNSharif PMLN sirf apne mufadat ke liye sports ko promote kar rahi hai, mulk ke liye nahi."
- Goal: Classify the comment as Agree, Disagree, or Unrelated to the tweet using RoBERTa.

Steps to Apply the RoBERTa Model While we preprocess the text (e.g., tokenization, lemmatization, and lowercasing), these steps are ignored by RoBERTa itself because it uses its own tokenizer. Here's how the data looks:

Processed Tweet: "pmln ne sports ko promote karna ka irada karna hai"

Processed Comment: "pmln sirf apna mufadat ka liye sports ko promote karna hai mulk ka liye nahi"

(i) **Input Preparation**
The tweet and comment are combined into a single sequence:
`X = [CLS] "pmln ne sports ko promote karna ka irada karna hai" [SEP] "pmln sirf apna mufadat ka liye sports ko promote karna hai mulk ka liye nahi" [SEP]`

- `[CLS]`: Marks the start of the sequence.
- `[SEP]`: Separates the tweet from the comment.

(ii) **Tokenization and Embedding Generation**
The input sequence X is tokenized using RoBERTa's subword tokenizer:
`Tokens: ["pmln", "ne", "sports", "ko", "pro", "##mote", ...]`
RoBERTa converts these tokens into contextual embeddings for all tokens, capturing semantic relationships between the tweet and comment.

(iii) **Attention Mechanism**
RoBERTa's self-attention mechanism compares every token with every other token in the sequence to:

- Show how the tweet and the comment are related.
- Focus on key words like "irada", "mufadat", and "nahi" to detect disagreement.

(iv) **[CLS] Token Embedding**
After processing the sequence, RoBERTa generates a special embedding for the [CLS] token:

$$E_{\text{CLS}} = \text{Contextual Representation of Entire Sequence}$$

This embedding encapsulates the relationship between the tweet and comment.

(v) **Classification Layer:**
The [CLS] embedding is passed through a fully connected layer to compute logits for each class (**Agree, Disagree, and Unrelated**):

$$Z = [z_{\text{agree}},\ z_{\text{disagree}},\ z_{\text{unrelated}}]$$

Example logits: $[-0.4,\ 2.5,\ 0.1]$.
The logits are converted into probabilities using the softmax function:

$$P(y \mid X) = \text{softmax}(Z)$$

Example probabilities: $[0.05,\ 0.92,\ 0.03]$.

Outcome

The model correctly classifies the comment as disagree because it contradicts the tweet's stance. By using

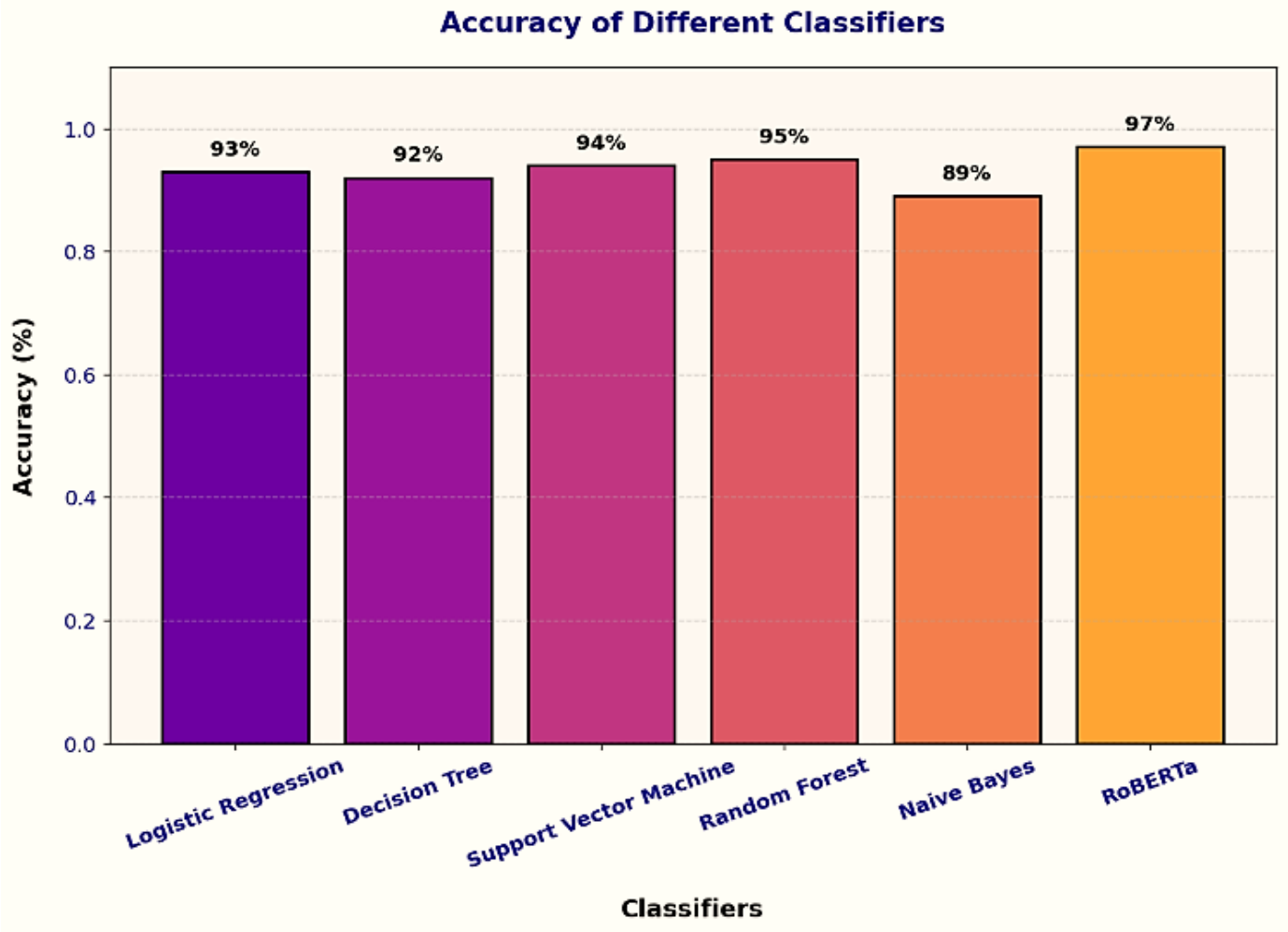


Figure 3. Accuracy comparison of different classifiers.

Table 8. Classification result of RoBERTa model.

Class	Precision	Recall	F1-Score	Support
Agree	0.98	0.93	0.95	2457
Disagree	0.95	0.99	0.97	2392
Unrelated	0.96	0.99	0.98	1850
Accuracy			0.97	6699
Macro Avg.	0.97	0.97	0.97	6699
Weighted Avg.	0.97	0.97	0.97	6699

RoBERTa embeddings and an attention mechanism, it effectively understands the context and nuances in the Roman Urdu text.

Classification Result of RoBERTa Model

The classification report for the augmented dataset using the RoBERTa model highlights its robust and balanced performance in stance classification tasks. The model gets a 97% overall accuracy rate, meaning it correctly predicts the stance in 97% of cases across all classes. The classification results of the RoBERTa model, evaluated using precision, recall, and F1-score across all stance categories, are summarized in Table 8. The model achieved an overall accuracy of 97%, demonstrating superior performance in detecting stance from Roman Urdu tweets.

4.5 Accuracy comparison of all classifiers

We evaluated the accuracy performance of various classifiers in predicting stance labels, using metrics such as accuracy, precision, recall, and F1-score to determine the most suitable classifier for stance detection. Among the tested models, the RoBERTa deep learning classifier achieved the highest accuracy at 97%, outperforming all other approaches. The machine-learning algorithms Random Forest Classifier followed with 95% accuracy, while Logistic Regression achieved 93%. The Support Vector Machine (SVM) recorded 94% accuracy, and the Decision Tree classifier achieved 92%. The Naïve Bayes classifier performed the worst, with an accuracy of 89%, which is close to random guessing. We examine training time, memory usage, and computational complexity across classifiers to evaluate model efficiency. While deep learning models like RoBERTa require more

processing power but provide better results, Random Forest offers cheaper computational cost but lesser accuracy. These results offer insight into the trade-offs between classification performance and efficiency. Misclassification primarily occurs in sarcastic, ambiguous, and indirect statements, where contextual cues are subtle and implicit. For example, some disagree-labeled tweets were misclassified as unrelated due to the absence of explicit negation, making it challenging for the model to capture implicit disagreement. This analysis highlights the need for advanced sarcasm detection techniques and contextual embeddings, which could significantly enhance stance classification accuracy in future research. The accuracy comparison of these classifiers is illustrated in Figure 3.

The bar chart provides a comparative analysis of the accuracy achieved by various classifiers for stance detection in Roman Urdu political tweets. Among the models evaluated, RoBERTa demonstrated the highest accuracy of 97%, showcasing its capability to handle complex linguistic structures and effectively capture contextual semantics in text. Random Forest achieved the second-highest accuracy at 95%, followed by Support Vector Machine (SVM) with 94%. Logistic Regression and Decision Tree models exhibited similar performance, achieving 93% and 92%, respectively. Naïve Bayes recorded the lowest accuracy at 89%, reflecting its limitations in dealing with the informal and code-mixed nature of Roman Urdu. The integration of dataset augmentation played a critical role in improving model performance by increasing data diversity and mitigating class imbalances, enabling the classifiers, particularly RoBERTa, to generalize better to unseen data. These results emphasize the effectiveness of RoBERTa combined with augmented datasets for stance detection tasks. We show the performance of each stance class (agree, disagree, unrelated) through F1-scores instead of using overall accuracy rates in our evaluation. The F1-score provides a balance between precision and recall in determining stance detection results. The performance metrics show RoBERTa reaches an F1-score of 0.95 for agree, 0.97 for disagree, and 0.98 for unrelated, demonstrating its excellence in stance classification across all categories.

5 Conclusion

This research delves into the area of stance detection over Roman Urdu tweets, its importance in the formation of public opinion on social media platforms by categorically using different machine learning and deep learning models. There were preprocessing methods like TF-IDF that really impacted classifier's performance. Out of the tested models, RoBERTa turned out as the best model with an accuracy of 97%, followed by Random Forest with 95%, SVM with 94%, Logistic Regression with 93%, Decision Tree with 92%, and Naïve Bayes with 89%. This shows the efficacy of applying transformer-based models on the stance detection tasks and also highlighting the role of preprocessing in achieving the best model performance. The study provides further insight into such stance detection in social media context and moreover emphasizes the importance of evaluation metrics including accuracy, precision, recall and F1 score to evaluate effectiveness of models. This research makes a contribution to computational linguistics and is a guide to future work in this area by providing insights on the strengths and limitations of different approaches.

To further improve results, we propose exploring novel feature engineering techniques, advanced preprocessing methods, and leveraging the latest deep learning frameworks, such as GPT-style models, XLNet, or cutting-edge multilingual transformers like XLM, for more accurate and robust stance detection in future research. Stance detection is essential for identifying misinformation and biased narratives on social media. By analyzing stances on controversial topics, this research enhances fact-checking systems, political discourse analysis, and misinformation mitigation strategies. Future studies will explore hybrid approaches that integrate stance detection with fake news classification to further improve detection accuracy and reliability. Future studies will investigate domain adaptation strategies to improve model generalization on a variety of multilingual datasets and social media platforms. Furthermore, in order to enhance stance identification performance beyond Roman Urdu datasets and guarantee greater applicability and resilience in a variety of textual settings, cross-domain training with linguistically related low-resource languages will be examined Additionally, investigating the impact of handling code-switching and informal language in a more comprehensive way could further enhance model performance. Future work could also focus on expanding the dataset to include a more diverse set of political topics, improving the generalizability of stance detection models, and developing tools that can operate efficiently in low-resource languages beyond Roman Urdu.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

Mohsin Shah served as an Associate Editor of the *ICCK Transactions on Advanced Computing and Systems* at the time of manuscript submission. To ensure the integrity of the peer-review process, Mohsin Shah was not involved in the editorial handling, peer review, or decision-making process for this manuscript, which was handled independently by another editor. The remaining authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Ghosh, S., Singhania, P., Singh, S., Rudra, K., & Ghosh, S. (2019). Stance detection in web and social media: a comparative study. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 10th International Conference of the CLEF Association, CLEF 2019, Lugano, Switzerland, September 9–12, 2019, Proceedings 10* (pp. 75-87). Springer International Publishing. [CrossRef]

[2] Cao, R., Lee, R. K.-W., & Hoang, T.-A. (2022). Stance detection for online public opinion awareness: An overview. *International Journal of Intelligent Systems*, 37(12), 11944-11965. [CrossRef]

[3] AlDayel, A., & Magdy, W. (2021). Stance detection on social media: State of the art and trends. *Information Processing & Management, 58*(4), 102597. [CrossRef]

[4] Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., & Cherry, C. (2016, June). Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 31-41). [CrossRef]

[5] Ansari, Z., Ali, S., & Khan, F. (2020). Use of roman script for writing urdu language. *International Journal of Linguistics and Culture, 1*(2), 165-178. [CrossRef]

[6] Alturayeif, N., Luqman, H., & Ahmed, M. (2023). A systematic review of machine learning techniques for stance detection and its applications. *Neural Computing and Applications, 35*(7), 5113-5144. [CrossRef]

[7] Küçük, D., & Can, F. (2019). A tweet dataset annotated for named entity recognition and stance detection. *arXiv preprint arXiv:1901.04787*. [CrossRef]

[8] Walker, M. A., Anand, P., Abbott, R., & Grant, R. (2012). That is your evidence?: Classifying stance in online political debate. *Decision Support Systems, 53*(4), 719-729. [CrossRef]

[9] Yan, Y., Chen, J., & Shyu, M. L. (2018). Efficient large-scale stance detection in tweets. *International Journal of Multimedia Data Engineering and Management, 9*(3), 1-16. [CrossRef]

[10] Ayyub, K., Javed, M., Shaukat, Z., & Ismail, M. (2021). Stance detection using diverse feature sets based on machine learning techniques. *Journal of Intelligent & Fuzzy Systems, 40*(5), 9721-9740. [CrossRef]

[11] Karande, H., Patil, S., & Joshi, R. (2021). Stance detection with BERT embeddings for credibility analysis of information on social media. *PeerJ Computer Science*, 7, e467. [CrossRef]

[12] Skanda, V. S., Kumar, M. A., & Soman, K. P. (2017, September). Detecting stance in kannada social media code-mixed text using sentence embedding. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 964-969). IEEE. [CrossRef]

[13] Siddiqua, U. A., Chy, A. N., & Aono, M. (2019, June). Tweet stance detection using an attention based neural ensemble model. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 1868-1873). [CrossRef]

[14] Tian, L., Zhang, X., Wang, Y., & Liu, H. (2020). Early detection of rumours on twitter via stance transfer learning. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I 42* (pp. 575-588). Springer International Publishing. [CrossRef]

[15] Kochkina, E., Liakata, M., & Augenstein, I. (2017). Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*. [CrossRef]

[16] Usman, M., Shafique, Z., Ayub, S., & Malik, K. (2016). Urdu text classification using majority voting. *International Journal of Advanced Computer Science and Applications, 7*(8). [CrossRef]

[17] Küçük, D., & Can, F. (2020). Stance detection: A survey. *ACM Computing Surveys (CSUR), 53*(1), 1-37. [CrossRef]

[18] Li, Y., Luo, Y., & Li, C. (2021). P-stance: A large dataset for stance detection in political domain. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2355-2365. [CrossRef]

[19] Küçük, D., & Can, F. (2018). Stance detection on tweets: An svm-based approach. *arXiv preprint arXiv:1803.08910*. [CrossRef]

[20] Lan, X., Gao, C., Jin, D., & Li, Y. (2024, May). Stance detection with collaborative role-infused llm-based agents. In *Proceedings of the international AAAI conference on web and social media* (Vol. 18, pp.

891-903). [CrossRef]

[21] Mets, M., Karjus, A., Ibrus, I., & Schich, M. (2024). Automated stance detection in complex topics and small languages: the challenging case of immigration in polarizing news media. *Plos one, 19*(4), e0302380. [CrossRef]

[22] Wang, X., Wang, Y., Cheng, S., Li, P., & Liu, Y. (2024). DEEM: Dynamic Experienced Expert Modeling for Stance Detection. *arXiv preprint arXiv:2402.15264*. [CrossRef]



Bilal Khan has received Master of Science (MS) in Computer Science from the Department of Computer Science, University of Science & Technology, Bannu, KP, Pakistan in 2024. His key areas of specialization include Machine Learning, Artificial Intelligence (AI), Data Science, and Natural Language Processing (NLP). (Email: bkbnu0928@gmail.com)



Khairullah Khan received the Ph.D. degree in information technology from University Teknologi PETRONAS, Malaysia, in 2012, where he worked on machine learning for the automatic detection of opinion targets from text. He is currently Professor with the Department of Computer Science, University of Science and Technology, Bannu, Pakistan. (Email: khair@ustb.edu.pk)



Fida Muhammad Khan is currently pursuing a Ph.D. degree in Computer Science at Qurtuba University of Science and Information Technology, Peshawar, Pakistan. He did his MS in Computer Science at the University of Science and Technology, Bannu, Pakistan. His research interests include Data Mining, Cybersecurity, IoT, Machine Learning, Deep Learning, and Natural Language Processing (NLP). (Email: fida5073@gmail.com)

Haseena Noureen completed her MS in computer science at the University of Malakand, Pakistan, where she currently holds the position of Lecturer in the Computer Science and IT Department. She is in the process of pursuing her PhD in Bioinformatics. Her research interests encompass a range of topics, including systems biology, computational modeling, Automata networks, graph theory, wireless sensor networks and software engineering. (Email: Haseenanoureen@uom.edu.pk)



Ahmad Ali received the master's and Ph.D. degrees from Shanghai Jiao Tong University, China, where he developed a solid foundation in computer science and engineering. Currently, he is a Postdoctoral Researcher with the College of Mechatronics and Control Engineering, Shenzhen University, China. In addition to his reviewing activities, he has been involved in several cutting-edge research projects. His work has led to numerous publications in top-tier journals and conferences, showcasing his expertise and contributions to the field. He is passionate about leveraging technology to solve real-world problems, particularly in urban environments where efficient data processing and intelligent decision-making are crucial. His research interests include deep learning, big data analytics, data mining, urban computing, cloud computing, and fog computing. He has reviewed over a thousand research articles for prestigious academic journals, such as Information Sciences, Information Fusion, Neural Networks, IEEE Internet of Things Journal, Neural Computing and Applications, Multimedia Tools and Application, Wireless Networks, and IEEE Access. (Email: ahmadali@szu.edu.cn)



Mohsin Shah received the BSc degree in Telecommunication Engineering from the University of Engineering and Technology Peshawar Pakistan in 2007, the MSc degree in Telecommunication Engineering from the University of Engineering and Technology Taxila Pakistan in 2012 and the PhD degree in Information and Communication Engineering from the University of Science and Technology of China in 2019. He has been a faculty member with the Department of Telecommunication, Hazara University Mansehra since 2009 where he is currently serving as an assistant professor. His research interests include information hiding, multimedia security, secure signal processing, image segmentation and image forgery detection. (Email: syedmohsinshah@hu.edu.pk)