Contextual Hope Speech Detection

Muhammad Haseeb

Department of Computer Science, Texas Tech University, Lubbock, Texas, USA

Abstract

This study presents a novel approach to hope speech detection in social media texts by moving beyond binary classification. We introduce a five-category taxonomy that distinguishes between Realistic Hope, Unrealistic Hope, Generalized Hope, Not Hope, and Sarcasm, capturing the nuanced expressions of hope in online interactions. Using a dataset of tweets, we implement a range of natural language processing techniques, including advanced preprocessing, contextual word embeddings, and both traditional and deep learning classification algorithms. Our experiments demonstrate that transformer-based models, particularly those leveraging contextual embeddings, outperform traditional machine learning approaches in identifying various hope categories. The inclusion of sarcasm detection adds an important dimension to hope speech analysis, accounting for seemingly positive content with potentially opposite intent. This research contributes to the growing field of positive content detection on social media and offers valuable insights for applications aimed at fostering supportive digital environments while acknowledging the complexity of human expression in online communication.

Keywords

Hope Speech Detection, Natural Language Processing, Social Media Text Analysis, Machine Learning Classification, Contextual Embeddings, Transformer Models, Sentiment Analysis, Multi-Class Classification, Sarcasm Detection

1. Introduction

Hope is one of the most beneficial emotions that a human being is blessed with, and it plays a vital role in all aspects of life—especially in human behavior, communication, and resilience, particularly in this era of digital spaces [1]. In recent years, various platforms like Twitter, Facebook, and other social media networks have emerged, where people share their personal beliefs, feelings, and thoughts, while the audience responds in their ways and according to their perceptions. This diversity in interpretation makes understanding the meaning behind a text one of the most challenging tasks.

The PolyHope shared task at IberLEF 2025 [2, 3, 4] was built to address this challenge and to develop computational and trustworthy systems to detect hope-related expressions in English and classify them into five distinct categories: *Realistic Hope*, *Unrealistic Hope*, *Generalized Hope*, *Sarcasm*, and *Not Hope*. These categories help differentiate the underlying meaning of the text—whether it reflects a genuine expression of hope, conveys sarcasm, or carries some other sentiment—by detecting subtle variations in language.

In this paper, we propose a comprehensive classification framework that spans from traditional machine learning models to advanced deep learning and transformer-based architectures. Various combinations of text preprocessing techniques have been employed to evaluate their effectiveness with different models. Multiple text representation techniques—including TF-IDF, static word embeddings (Word2Vec, GloVe), and contextual embeddings using transformer models such as BERT, RoBERTa, and DeBERTa—have been implemented. Our approach covers both binary classification and multi-class hope classification tasks as outlined in the shared task.

As a whole, our work presents four major contributions. First, we propose a multi-stage classification framework that integrates both classical and state-of-the-art NLP techniques for detecting hope-related expressions. Second, we explore a variety of preprocessing strategies and evaluate their impact across different model families, highlighting which techniques are beneficial in specific modeling paradigms. Third, we investigate the performance and limitations of each model family and conduct a comparative

IberLEF 2025 September 2025, Zaragoza, Spain

mhaseeb@ttu.edu (M. Haseeb)

https://www.linkedin.com/in/muhammad-haseeb04 (M. Haseeb)

analysis of multiple transformer architectures for the multi-class classification task, including a multi-task BERT model. Finally, our system achieved competitive results in the PolyHope shared task, ranking first in the binary classification track and second in the multi-class classification track.

2. Literature Review

Hope speech detection is an emerging area of study in natural language processing, aiming to identify positive and encouraging content in often sarcastic and emotionally charged social media environments. This task differs from earlier efforts that primarily focused on detecting and removing harmful material such as hate speech or offensive language.

Chakravarthi (2020) [5] defined hope speech detection as a computational task within the framework of Equality, Diversity, and Inclusion (EDI). The first multilingual hope speech detection dataset, HopeEDI, was created with 28,451 English, 20,198 Tamil, and 10,705 Malayalam YouTube comments, manually classified as either containing hope speech or not. The study reported moderate inter-annotator agreement scores, with Krippendorff's alpha at 0.63, and evaluated baseline performance using several machine learning classifiers, with decision trees achieving the highest macro F1 score of 0.46.

Building on this, Chakravarthi et al. (2022) [6] organized a shared task on hope speech detection at LT-EDI 2022. The task expanded to five languages—Tamil, Madras, Kannada, Spanish, and French. Participants employed diverse modeling approaches, with many top-performing systems utilizing transformer-based models. These systems achieved macro F1 scores ranging from 0.50 to 0.81 across different languages, highlighting the effectiveness of transformer architectures.

Garcia-Baena et al. (2023) [7] created a Spanish hope speech detection dataset focused on LGBTQ+ content. They defined hope speech as language that diffuses hostile environments and provides help, suggestions, and inspiration during difficult times. Their dataset, SpanishHopeEDI, consisted of 1,650 tweets about the LGBTQ community, annotated for presence or absence of hope speech. The study reported strong inter-annotator agreement (Krippendorff's Alpha = 0.88) and demonstrated strong performance using BERT-based models.

Jimenez-Zafra et al. (2023) [8] organized a hope speech detection shared task at IberLEF 2023, covering both English and Spanish. The competition attracted 50 registered teams, with 12 submitting results and 8 providing working notes. Subtasks included detecting hope speech in Spanish tweets and English YouTube comments. The best-performing systems achieved macro F1 scores of 0.916 for Spanish and 0.501 for English. Notably, ChatGPT-based approaches performed especially well on the Spanish subtask.

A novel approach was proposed by Balouchzahi et al. (2023) [9], extending the task from binary to two-level hope speech detection. The first level classified texts as either hope or not hope, and the second level further categorized hope expressions into Generalized Hope, Realistic Hope, or Unrealistic Hope. Their transformer-based models, especially BERT, achieved the highest macro F1 score of 0.72.

Jimenez-Zafra et al. (2024) [10] organized the second edition of the hope shared task at IberLEF 2024, focusing on two perspectives: hope for EDI and hope as expectation. The competition included 19 participating teams. The top team achieved a macro F1 score of 71.61 for the EDI subtask, while the best-performing teams in the expectations subtask achieved F1 scores exceeding 80.00 for binary classification and 78.50 for multiclass classification.

In a related direction, Sidorov et al. (2023) [11] compared different transformer models for detecting hope and regret. They evaluated several architectures using the PolyHope dataset for hope detection and the ReDDIT dataset for regret detection. Their results showed that uncased BERT-based models performed best for both tasks (macro F1 of 0.72 and 0.83, respectively). The study also found that longer textual context improved transformer performance.

2.1. Our Approach

Expanding on previous studies, our research aims to create a system that can detect hopeful speeches and classify them into binary and multiclass categories. Previous research grouped hope speech into a

limited number of subcategories, but we suggest expanding this classification to include five categories:

- Realistic hope.
- Unrealistic hope.
- Generalized hope.
- Not hope.
- Sarcasm.

We approach the research differently compared to previous studies in several important aspects. Initially, we emphasize the context, as hope can be expressed in various ways depending on the language and culture in which it is used. Next, we include the identification of sarcasm as a key component, as optimistic statements may actually convey a negative sentiment through sarcasm. Lastly, we ensure that the training data is cleaned up. We make use of a variety of classifiers as well as word embedding methods in our methodology to identify subtle semantic connections in expressions related to hope. Our goal is to improve the detection of hopeful messages to create a more positive and encouraging online atmosphere, while keeping the human aspect of digital interaction.

3. Methodology

Our text classification approach starts by dividing data into train-test sets using stratified sampling to preserve class proportions, followed by thorough data cleaning. Initially, we weighted word importance through TF-IDF vectorization, feeding these weighted features into several classifiers - Naive Bayes, Logistic Regression, Random Forest, XGBoost, SVM, and simple neural networks. After identifying top performers, we fine-tuned their parameters to maximize accuracy.

The encouraging results prompted us to explore more sophisticated techniques. We implemented various word embedding methods, including Word2Vec and GloVe, which we incorporated into both traditional classifiers and deeper neural architectures to better capture semantic relationships within texts.

To understand the impact of preprocessing techniques, we created four dataset variations. Our baseline (Version 1) included complete processing with lemmatization, contraction expansion, and emoji-to-text conversion.To further enhance performance, we incorporated bigram and trigram features to capture phrase-level patterns in classifiers. We then systematically excluded one technique at a time: Version 2 skipped lemmatization, Version 3 retained emojis in their original form, and Version 4 kept contractions unexpanded. This methodical approach helped us isolate each preprocessing step's contribution across different model architectures, revealing which techniques had the most significant impact on classification performance for our specific tasks.

3.1. Data Stratification

To ensure balanced representation across our training, validation, and testing splits for both classification tasks, we implemented a composite stratification approach. Since standard stratification methods only support a single label column, we combined the binary and multi-class labels using a hyphen delimiter. This strategy preserved the distribution of all label combinations across data splits, maintaining the integrity of both classification tasks simultaneously. Our composite stratification method effectively addressed the class imbalance inherent in hope speech data, ensuring proportional representation of less frequent categories, such as Sarcasm. This approach resulted in robust dataset splits that accurately reflect the original data distribution for both binary and multi-class classification tasks.

3.2. Data Cleaning

To improve the quality of data and analytical efficiency we have done some data cleaning prior to feeding it for classification. Stopwords and URLs were removed for dimensionality reduction. Punctuation

was removed to reduce noise and unnecessary non-essential tokenization. Emojis were converted to their textual equivalents, preserving their semantic contribution while standardizing vocabulary and improving tokenization. In order to focus on the semantic analysis we implemented contraction expansion. Lastly, lemmatization reduced words to their base forms further reducing the vocabulary and highlighting word frequency patterns. Together these techniques gave us a cleaner, structured dataset suited for natural language processing tasks.

3.3. Basic Vectorization

After preprocessing our dataset, we implemented TF-IDF vectorization as our primary feature extraction method. Our approach involved creating four distinct processing pipelines to evaluate the impact of different text normalization techniques.

For our baseline implementation (Version 1), we applied comprehensive text processing including lemmatization, contraction expansion, and emoji-to-text conversion. We enhanced this approach by incorporating bigram and trigram features to capture contextual patterns which were significant for hope speech detection.

To assess the contribution of individual preprocessing components, we created variants by excluding specific techniques. Version 2 omitted lemmatization while retaining other processes. Version 3 preserved emojis in their original form instead of converting them to text descriptions. Version 4 maintained contractions in their unexpanded state.

We evaluated these vectorization variants across multiple classification algorithms, including Naive Bayes, Logistic Regression, Random Forest, XGBoost, and Support Vector Machines. Additionally, we tested these four processing pipelines without n-gram features on a Feed Forward Neural Network architecture. The comparative results and their implications are discussed in the Results section.

3.4. Word Embedding

After TF-IDF vectorization, we proceeded to implement more sophisticated natural language processing techniques using word embeddings. Word embeddings provide in-depth vector representations that capture semantic relationships between words, offering deeper insights into the contextuality of hope.

We have used two distinct word embedding approaches: Word2Vec and GloVe (Global Vectors for Word Representation). These techniques transform words into continuous vector spaces where semantically similar words are positioned closer together, which helps us in capturing linguistic patterns and relationships than traditional approaches like vectorization and bag-of-words.

For evaluation of Word2Vec embeddings, we tested performance on traditional classifiers such as Random Forest and Logistic Regression. These models helped us to understand the value added by word embeddings over TF-IDF vectorization.

To leverage the full potential of word embeddings, we implemented a series of neural network architectures of increasing complexity. We began with a Feed Forward Neural Network, followed by a Convolutional Neural Network (CNN) to identify local patterns and features within the embeddings. For sequential pattern recognition, we implemented a Recurrent Neural Network using Long Short-Term Memory (LSTM) cells, which are particularly effective at capturing long-range dependencies in text.

The implementation of these various architectures allowed us to systematically evaluate how different neural network structures interact with word embeddings to identify hope speech patterns. Each model configuration was evaluated on both binary and multi-class classification tasks. The results of the performance for the embedding-based approaches is presented in the later section, with attention to how effective these models are to capture various categories of hope expression in our taxonomy.

3.5. Contextual Embedding

Our analysis aims to identify various types of hope expressed in Twitter posts by examining the context in which the tweets are written. To reach this, we utilized contextual embedding methods that predict words by considering the words around them instead of viewing them in isolation. Contextual

embeddings generate word representations depending on the specific sentences in which they are used, as opposed to conventional word embeddings which assign a fixed vector to every word irrespective of context. We have used several transformer-based models for contextual embedding.

- **DistilBERT**: A lighter, faster version of BERT that retains much of its performance while requiring fewer computational resources
- **BERT (Bidirectional Encoder Representations from Transformers)**: Considers context from both directions within text, allowing better understanding
- **RoBERTa (Robustly Optimized BERT Pretraining Approach)**: An optimized version of BERT with improved training methodology
- **DeBERTa (Decoding-enhanced BERT with disentangled attention)**: Enhances BERT architecture with disentangled attention mechanisms
- **Multi-task BERT**: Uses multiple attention heads to capture different aspects of context simultaneously and gives two outputs for binary and multi-label, respectively.

While our word embedding experiment yielded better results with deep learning models, our contextual embedding approach is exclusively focused on these transformer-based architectures. The comparative performance of these models and their effectiveness in hope classification are discussed in subsequent sections.

4. Results

In this section, we will discuss the results with combination of various techniques and classifiers. We organize our results into three main sections based on the modeling approaches used: the TF-IDF based classifiers, word embedding based models, and contextual embedding using transformer architectures.

4.1. TF-IDF Approach

We employed a range of machine learning classifiers to investigate the details of hope representation in social media discourse. These algorithms include classical statistical methods to deep learning approaches. Each classifier was evaluated using different feature representations derived from TF-IDF vectors. For the top-performing models, hyperparameter tuning was performed using Hyperopt and Grid search. To systematically assess the impact of text preprocessing, we developed four TF-IDF feature variants:

- **V1**: Full preprocessing, including lemmatization, contraction expansion, emoji-to-text conversion, and the removal of stopwords and punctuation.
- **V2**: Same as V1 but without lemmatization.
- **V3**: Same as V1 but emojis were retained in their original form.
- **V4**: Same as V1 but contractions were not expanded

This experimental setup enabled us to isolate the effect of each individual preprocessing step across various classifiers.

The research methodology used six classification techniques listed below:

1. Naive Bayes
2. Logistic Regression
3. Support Vector Machine
4. Random Forest
5. XGBoost
6. Feed-Forward Neural Network

4.1.1. Binary Classification Results

Among the evaluated models, Logistic Regression consistently demonstrated the most outstanding performance, achieving the highest accuracy of 80.34% across its V1 and V3 configurations. The analysis encompassed multiple feature representations, including baseline versions (V1–V4) and n-gram models (bigrams and trigrams), revealing nuanced insights into the classifiers' predictive capabilities. Logistic Regression consistently outperformed other algorithms such as Support Vector Machine (SVM), Naïve Bayes, Random Forest, XGBoost, and the Feed-Forward Neural Network. The study highlighted the effectiveness of linear models in capturing the subtle variations of hope detection, with bigram feature representations generally enhancing model performance. While most classifiers showed accuracies between 74% and 80%, the Feed-Forward Neural Network exhibited the lowest performance, suggesting that simpler models might be more suitable for this specific text classification task.

Table 1
Performance of Machine Learning Models on Binary Hope Speech Classification Using TF-IDF Representations. V1–V4 indicate preprocessing variants, with Bigrams and Trigrams representing extended n-gram features.

Model	Version	Accuracy	W. Prec	W. Rec	W. F1	M. Prec	M. Rec	M. F1
NaiveBayes	V1	0.7473	0.7523	0.7473	0.7437	0.7542	0.7400	0.7409
NaiveBayes	V2	0.7562	0.7593	0.7562	0.7536	0.7608	0.7500	0.7512
NaiveBayes	V3	0.7473	0.7523	0.7473	0.7437	0.7542	0.7400	0.7409
NaiveBayes	V4	0.7473	0.7523	0.7473	0.7437	0.7542	0.7400	0.7409
NaiveBayes	Bigrams	0.7595	0.7782	0.7595	0.7519	0.7827	0.7485	0.7482
NaiveBayes	Trigrams	0.7515	0.7757	0.7515	0.7419	0.7808	0.7393	0.7377
LogReg	V1	**0.8034**	**0.8045**	**0.8034**	**0.8024**	**0.8054**	**0.7994**	**0.8009**
LogReg	V2	0.7959	0.7967	0.7959	0.7950	0.7974	0.7921	0.7934
LogReg	V3	**0.8034**	**0.8045**	**0.8034**	**0.8024**	**0.8054**	**0.7994**	**0.8009**
LogReg	V4	0.8034	0.8045	0.8034	0.8024	0.8054	0.7994	0.8009
LogReg	Bigrams	0.7964	0.7964	0.7964	0.7959	0.7965	0.7937	0.7945
LogReg	Trigrams	0.7982	0.7982	0.7982	0.7978	0.7982	0.7957	0.7965
SVM	V1	0.7931	0.7931	0.7931	0.7927	0.7930	0.7905	0.7913
SVM	V2	0.7922	0.7922	0.7922	0.7917	0.7922	0.7895	0.7903
SVM	V3	0.7931	0.7931	0.7931	0.7927	0.7930	0.7905	0.7913
SVM	V4	0.7931	0.7931	0.7931	0.7927	0.7930	0.7905	0.7913
SVM	Bigrams	0.8010	0.8011	0.8010	0.8010	0.8001	0.8001	0.8001
SVM	Trigrams	0.7987	0.7991	0.7987	0.7988	0.7977	0.7985	0.7980
RndFrst	V1	0.7964	0.7967	0.7964	0.7965	0.7954	0.7960	0.7956
RndFrst	V2	0.7889	0.7891	0.7889	0.7890	0.7879	0.7883	0.7880
RndFrst	V3	0.7870	0.7874	0.7870	0.7871	0.7860	0.7867	0.7863
RndFrst	V4	0.7805	0.7811	0.7805	0.7806	0.7795	0.7803	0.7798
RndFrst	Bigrams	0.7674	0.7724	0.7674	0.7676	0.7693	0.7701	0.7674
RndFrst	Trigrams	0.7515	0.7553	0.7515	0.7518	0.7525	0.7536	0.7514
XGBoost	V1	0.7837	0.7836	0.7837	0.7833	0.7834	0.7813	0.7820
XGBoost	V2	0.7894	0.7894	0.7894	0.7888	0.7895	0.7865	0.7874
XGBoost	V3	0.7837	0.7836	0.7837	0.7833	0.7834	0.7813	0.7820
XGBoost	V4	0.7837	0.7836	0.7837	0.7833	0.7834	0.7813	0.7820
XGBoost	Bigrams	0.7688	0.7691	0.7688	0.7689	0.7677	0.7682	0.7679
XGBoost	Trigrams	0.7590	0.7594	0.7590	0.7591	0.7579	0.7585	0.7581
FFN	V1	0.7268	0.7266	0.7268	0.7257	0.7264	0.7230	0.7237
FFN	V3	0.7244	0.7245	0.7244	0.7231	0.7245	0.7202	0.7210
FFN	V4	0.7240	0.7254	0.7240	0.7243	0.7234	0.7244	0.7234
FFN	V2	0.7221	0.7217	0.7221	0.7214	0.7212	0.7189	0.7195

4.1.2. Multi-Class Classification Results

Regarding the multi-class configuration, the best-performing algorithm was XGBoost, which achieved 68.47% accuracy in the V1 and V3 configurations, outperforming other models. The analysis revealed a significant increase in complexity compared to the binary classification task, with overall accuracy dropping relative to previous results. Following XGBoost, Logistic Regression and Support Vector Machine (SVM) achieved 63.52% and 66.42% accuracy, respectively.

Notably, performance metrics varied considerably across different feature representations. Most classifiers experienced a decline in precision, recall, and F1-scores when switching to bigram and trigram features. Naïve Bayes delivered the weakest performance, with accuracies around 47.50% and macro F1-scores below 18.85%, indicating the difficulty of multi-class hope detection. Once again, the Feed-Forward Neural Network underperformed compared to other models, suggesting that the multi-class classification task may require more robust feature engineering or advanced neural network architectures.

Table 2
Performance of Machine Learning Models on Multi-Class Hope Speech Classification Using TF-IDF Representations

Model	Version	Acc	W Prec	W Rec	W F1	M Prec	M Rec	M F1
NaiveBayes	V1	0.4750	0.5184	0.4750	0.3452	0.4467	0.2385	0.1885
NaiveBayes	V2	0.4811	0.5137	0.4811	0.3537	0.4421	0.2433	0.1947
NaiveBayes	V3	0.4750	0.5184	0.4750	0.3452	0.4467	0.2385	0.1885
NaiveBayes	V4	0.4750	0.5184	0.4750	0.3452	0.4467	0.2385	0.1885
NaiveBayes	Bigrams	0.4568	0.3895	0.4568	0.3131	0.2510	0.2226	0.1630
NaiveBayes	Trigrams	0.4512	0.3881	0.4512	0.3023	0.2503	0.2179	0.1547
LogReg	V1	0.6352	0.6485	0.6352	0.6069	0.6619	0.4833	0.5224
LogReg	V2	0.6245	0.6381	0.6245	0.5947	0.6485	0.4711	0.5075
LogReg	V3	0.6352	0.6485	0.6352	0.6069	0.6619	0.4833	0.5224
LogReg	V4	0.6352	0.6485	0.6352	0.6069	0.6619	0.4833	0.5224
LogReg	Bigrams	0.6119	0.6381	0.6119	0.5732	0.6592	0.4452	0.4769
LogReg	Trigrams	0.5993	0.6298	0.5993	0.5520	0.6512	0.4226	0.4459
SVM	V1	0.6642	0.6628	0.6642	0.6515	0.6561	0.5566	0.5894
SVM	V2	0.6534	0.6502	0.6534	0.6396	0.6400	0.5449	0.5754
SVM	V3	0.6642	0.6628	0.6642	0.6515	0.6561	0.5566	0.5894
SVM	V4	0.6642	0.6628	0.6642	0.6515	0.6561	0.5566	0.5894
SVM	Bigrams	0.6520	0.6570	0.6520	0.6388	0.6503	0.5391	0.5728
SVM	Trigrams	0.6422	0.6508	0.6422	0.6273	0.6426	0.5230	0.5558
RndFrst	V1	0.6198	0.6298	0.6198	0.5917	0.6329	0.4744	0.5074
RndFrst	V2	0.6305	0.6401	0.6305	0.6029	0.6442	0.4856	0.5186
RndFrst	V3	0.6324	0.6460	0.6324	0.6046	0.6586	0.4846	0.5213
RndFrst	V4	0.6310	0.6425	0.6310	0.6014	0.6495	0.4817	0.5154
RndFrst	Bigrams	0.6030	0.6306	0.6030	0.5610	0.6551	0.4415	0.4636
RndFrst	Trigrams	0.5885	0.5987	0.5885	0.5385	0.6054	0.4143	0.4278
XGBoost	V1	**0.6847**	0.6673	**0.6847**	0.6717	0.6360	0.6048	**0.6134**
XGBoost	V2	0.6819	0.6640	0.6819	0.6697	0.6248	0.6018	0.6078
XGBoost	V3	**0.6847**	0.6673	**0.6847**	0.6717	0.6360	0.6048	**0.6134**
XGBoost	V4	**0.6847**	0.6673	**0.6847**	0.6717	0.6360	0.6048	**0.6134**
XGBoost	Bigrams	0.6688	0.6575	0.6688	0.6583	0.6119	0.5978	0.5969
XGBoost	Trigrams	0.6600	0.6505	0.6600	0.6498	0.6093	0.5924	0.5917
FFN	V1	0.5899	0.5809	0.5899	0.5832	0.5360	0.5035	0.5168
FFN	V2	0.5792	0.5750	0.5792	0.5702	0.5384	0.4791	0.4987
FFN	V3	0.5899	0.5794	0.5899	0.5809	0.5308	0.4982	0.5087
FFN	V4	0.5918	0.5813	0.5918	0.5818	0.5382	0.4946	0.5093

4.2. Word Embedding Approach

This research employed two popular word embedding techniques, Word2Vec and GloVe, to obtain semantic representations of words from our corpus. These techniques convert text into dense vector representations that capture semantic relationships and linguistic nuances.

The Word2Vec algorithm was applied using the skip-gram model. In this configuration, 300-dimensional vectors with 10-word context windows were used to predict surrounding context words to capture complex word relationships. Words appearing less than three times were excluded, and the model was trained for 20 epochs. Such an approach allows for a more nuanced semantic representation that encapsulates contextual and syntactic differences in the text corpus.

We added pre-trained GloVe embeddings for additional embedding diversity. The embedding loading process involved parsing a pre-trained GloVe embedding file and creating a dictionary mapping words to their vector representations. With this approach, global word co-occurrence statistics were derived for a large corpus.

4.2.1. Deep Learning Model Integration

To explore semantic representations we implemented three different deep learning architectures: Feed-forward neural networks (FNN), convolutional neural networks (CNN), and long short-term memory networks (LSTM) are examples. Each model used word embeddings as input features to investigate text classification across different neural network paradigms.

4.2.2. Feed-Forward Neural Network

Our work implemented two Feed-Forward Neural Network architectures for text classification based on their input layer and embedding strategies. The first employed pre-trained word embeddings as fixed-dimensional input vectors in a network structure having two hidden layers of 128 and 64 neurons each with ReLU activation. The randomly deactivated neurons were suppressed by introducing a 0.3 dropout layer to suppress overfitting during training.

The second approach embedded a trainable embedding layer directly into neural network architecture using more integrated feature representation method. Here, input tokens are mapped to dense vector representations and an embedding matrix is initialized with pre-trained embeddings but can be tuned during training. Then the network flattens to make the embedded sequences one vector, then dropout and dense layers that mirror the first approach. The real difference lies in input processing: The trainable embedding layer can dynamically adapt word representations during training to better capture dataset specific semantic nuances.

4.2.3. Convolutional Neural Network

Our CNN architectures use a trainable embedding layer as the first input stage mapping input sequences to dense vector representations with pre-trained word embeddings. The binary classification model employs a single convolutional layer with 128 filters and a 5-size kernel to capture local textual features via ReLU activation. The multi-class CNN has two consecutive Conv1D layers with 128 filters and a 5-size kernel and max-pooling layers to reduce feature dimensionality and obtain hierarchical representations.

In both models global max-pooling is used to aggregate most significant features across convolutional layers to produce feature maps with fixed-length representation. Dropout layers are integrated at multiple stages at rates 0.3 - 0.5 to prevent overfitting and promote model generalization. Dense layers activated by ReLU further abstract and transform the extracted features to provide a robust feature representation mechanism for capturing semantic patterns in text sequences.

4.2.4. Long Short-Term Memory Network

LSTM models follow a consistent architectural approach for binary and multi-class classification tasks with a trainable embedding layer as first input processing stage. The embedding layer maps input

sequences to dense vector representations initialized with pre-trained word embeddings and dynamically adapted during training. A key architectural feature is Bidirectional LSTM processing input sequences in forward and backward directions and capturing context from past and future sequence elements.

In both models the core has Bidirectional LSTM layer with 64 units returning one output vector aggregating the most significant sequential features. This allows the model to capture complex temporal dependencies and contextual details of the input text sequences. Dropout layers are added at 0.5 rate to prevent overfitting and followed by a 64 - neuron dense layer with ReLU activation. The architectural design focuses on capturing sequential patterns and contextual relationships leveraging the LSTM to maintain long-term dependencies on text data and robust feature representation mechanisms.

4.2.5. Significance of Approach

Our multi-faceted approach to word embedding generation combines the strengths of different embedding techniques, providing a robust and comprehensive semantic representation strategy. By employing both Word2Vec and GloVe, we mitigate potential limitations inherent in individual embedding methods and create a more nuanced linguistic feature representation.

Table 3
Performance of Deep Learning Models on Binary Hope Speech Classification Using Word Embeddings

Model	Accuracy	Macro F1	Weighted F1
Word2Vec - FFN	0.76	0.76	0.76
Word2Vec - FNN + Embedding	0.73	0.72	0.73
Word2Vec - CNN	0.69	0.69	0.69
Word2Vec - LSTM	0.76	0.76	0.76
GloVe - FFN	0.74	0.74	0.74
GloVe - FNN + Embedding	0.74	0.74	0.74
GloVe - CNN	0.78	0.78	0.78
GloVe - LSTM	0.78	0.77	0.78

Table 4
Performance of Deep Learning Models on Multi-Class Hope Speech Classification Using Word Embeddings

Model	Accuracy	Macro F1	Weighted F1
Word2Vec - FFN	0.59	0.44	0.55
Word2Vec - FNN + Embedding	0.63	0.52	0.60
Word2Vec - CNN	0.48	0.27	0.43
Word2Vec - LSTM	0.54	0.33	0.49
GloVe - FFN	0.56	0.39	0.52
GloVe - FNN + Embedding	0.54	0.37	0.49
GloVe - CNN	0.47	0.26	0.42
GloVe - LSTM	0.53	0.30	0.47

Note. The results reveal that deep learning models based on static embeddings such as Word2Vec and GloVe perform poorly in multi-label situations. LSTM and CNN architectures can capture word context but not effectively. Interestingly, traditional ML models with TF-IDF features outperformed these deep learning baselines, indicating the strength of simpler models without contextual embeddings.

4.3. Contextual Embedding

4.3.1. Pre Processing

In our experimentation with contextual models, we explored multiple pre-processing strategies. During this process, we observed that conventional NLP techniques, such as lemmatization, stopword removal, and punctuation stripping, degraded the performance of our transformer-based models. It is preferable to use raw text with these types of models, as they are capable of extracting deeper contextual meaning from elements like punctuation, stopwords, and un-lemmatized forms, as these elements often carry semantic significance. Our final preprocessing pipeline includes expanding contractions (e.g., "isn't" $\rightarrow$ "is not") and converting emojis into descriptive text, as emojis are typically treated as out-of-vocabulary or unknown tokens. This strategy was applied consistently across all transformer models to ensure a fair comparison of performance under similar input conditions.

All transformer models used a maximum input sequence length of 128, tokenized using model-specific tokenizers, and were evaluated on identical training and test splits across binary and multi-class configurations.

4.3.2. DistilBERT for Text Classification

We utilized a couple of models from the BERT family. The first one we used was **DistilBERT**, a distilled version of BERT, which retains almost 96% of BERT's performance while being significantly smaller in size. It is pretrained using knowledge distillation, where a smaller model learns to replicate the behavior of a larger teacher model (BERT) through certain optimizations.

Our experiment used a fine-tuned version of DistilBERT. We trained two models for binary and multi-class classification tasks. We updated the classification heads of the base models to map the `[CLS]` token embedding to output labels. Training was performed using cross-entropy loss and the AdamW optimizer. The batch size was set to 16 for training and 32 for evaluation. We experimented with different learning rates, including {5e-5, 3e-5, 2e-5}. The training loop was run for 20 epochs with early stopping integrated, using a patience of 5 and accuracy as the evaluation metric. The processed inputs were tokenized using `DistilBertTokenizer`, padded to a maximum length of 128, and fed into the model along with the attention mask.

4.3.3. BERT for Text Classification

We further evaluated the standard BERT base model `bert-base-uncased` to see if a larger architecture leads to better performance compared to DistilBERT. In contrast to DistilBERT, which is a lighter and faster variant, BERT base has 12 layers and 110 million parameters, but requires more computation to represent context.

We trained two separate models with BERT: one for binary classification and another for multi-class classification. The training setup followed the DistilBERT experiment: by fine-tuning the classification heads with cross-entropy loss and AdamW optimization, and including early stopping with a patience of 5, we fine-tuned the classification heads. We kept the same batch sizes (16 for training, 32 for evaluation) and explored learning rates of {5e-5, 3e-5, 2e-5}. Tokenization was done using the `BertTokenizer` with a maximum sequence length of 128.

The final evaluation was performed using accuracy and the confusion matrix. Results show detailed performance comparisons with DistilBERT.

4.3.4. RoBERTa for Text Classification

We also evaluated the standard RoBERTa base model against other models. RoBERTa is an enhanced version of BERT trained without the Next Sentence Prediction (NSP) objective in larger mini-batches and over a longer period. RoBERTa has 12 layers and 125 million parameters, whereas DistilBERT requires more computation.

Two separate models were trained with RoBERTa: one for binary and one for multi-class classification. The training setup followed the DistilBERT experiment: We fine-tuned the classification heads by adjusting the classification heads with cross-entropy loss, optimizing with AdamW, and early stopping with a patience of 5. The same batch sizes (16 for training and 32 for evaluation) and different learning rates were experimented with {5e-5, 3e-5, 2e-5}. Tokenization was performed with `AutoTokenizer` with a maximum sequence length of 128.

The final evaluation was based on accuracy and confusion matrix analysis. Results show comparisons with other related models.

4.3.5. DeBERTa for Text Classification

Furthermore, we assessed DeBERTa's performance using the `microsoft/deberta-v3-base` model, which separates positional and content embeddings and employs an enhanced attention mechanism compared to BERT. It has 12 layers and approximately 183 million parameters—making it larger and more expressive than both BERT and RoBERTa.

Two separate models were trained using DeBERTa: one for binary classification and another for multi-class classification. The training setup was identical to earlier models: fine-tuning of the classification heads with cross-entropy loss, optimization with AdamW, and early stopping with a patience of 5 epochs. We used a batch size of 16 for training and 32 for evaluation, and experimented with learning rates of {5e-5, 3e-5, 2e-5}. Tokenization was carried out using `AutoTokenizer` with a maximum sequence length of 128 tokens.

4.3.6. BERT Multi-Task Text Learning

Our research investigated an innovative approach to multi-task learning (MTL) centered on leveraging a BERT-based model architecture for handling diverse classification challenges. Instead of utilizing a conventional BERT encoder with shared classification layers, we implemented a methodology involving two distinct model training paths. Our first classification objective focused on a binary task of sentiment distinction, specifically discerning between hopeful and despairing tweet sentiments. Complementing this, we developed a multi-class classification framework designed to categorize tweets across five nuanced semantic domains: sarcastic, general, despair, unrealistic, and realistic.

The efficacy of multitask learning emerges most prominently when different classification tasks leverage identical input characteristics. In our approach, we applied both classification methodologies to a single tweet's textual content, where the binary classification serves as a broader contextual signal to inform the more granular multi-class categorization. This integrated learning paradigm enables the model to develop more sophisticated and adaptable representational insights, particularly when confronted with constrained labeled datasets.

Our model proposes a single BERT encoder base with two classification pathways. A binary classification head with two possible output categories is used in the first pathway, whereas in the second, a multi-class classifier is used to distinguish between five discrete classes. A composite loss function that aggregates binary and multi-class cross-entropy losses is implemented which allows the shared encoder to optimize representations for both classification tasks. This strategic approach improves the model performance and allows stronger generalization capabilities.

4.3.7. Performance Comparison

To evaluate the effectiveness of transformer-based contextual embedding models, we compare their performance across both binary and multi-class hope speech classification tasks. Table 5 summarizes results for binary classification, while Table 6 presents the corresponding performance on multi-class classification.

RoBERTa achieved the highest performance in both tasks, particularly in weighted and macro-averaged metrics, indicating its robustness across classes with imbalanced distributions. DistilBERT, on the other hand, performed competitively despite its compact size, making it a favorable choice

for environments with limited resources. The multi-head BERT model demonstrated slightly better generalization than the standard BERT, especially in the multi-class task, highlighting the advantage of shared contextual learning. DeBERTa, although the largest in terms of parameter count, underperformed relative to RoBERTa, possibly due to the small training dataset causing overfitting.

Table 5
Performance of Transformer-Based Models on Binary Hope Speech Classification

Model	Acc	W. Prec	W. Rec	W. F1	M. Prec	M. Rec	M. F1
DistilBERT	0.85	0.86	0.85	0.85	0.85	0.85	0.85
BERT	0.84	0.84	0.84	0.84	0.84	0.84	0.84
Multi-Head BERT	0.85	0.85	0.85	0.85	0.85	0.85	0.85
RoBERTa	**0.87**	**0.88**	**0.87**	**0.87**	**0.88**	**0.87**	**0.87**
DeBERTa	0.84	0.85	0.84	0.84	0.84	0.84	0.84

Table 6
Performance of Transformer-Based Models on Multi-Class Hope Speech Classification

Model	Acc	W. Prec	W. Rec	W. F1	M. Prec	M. Rec	M. F1
DistilBERT	0.76	0.76	0.76	0.75	0.74	0.68	0.70
BERT	0.75	0.76	0.75	0.76	0.72	0.72	0.72
Multi-Head BERT	0.77	0.77	0.77	0.77	0.74	0.72	0.73
RoBERTa	**0.77**	**0.77**	**0.77**	**0.77**	**0.74**	**0.73**	**0.73**
DeBERTa	0.72	0.75	0.72	0.72	0.68	0.72	0.69

4.4. Inference

Our work employed two distinct inference methods across different transformer models: BERT, DistilBERT, RoBERTa, and DeBERTa. The standard approach used with BERT, DistilBERT, RoBERTa, and DeBERTa is a two-model strategy wherein separate models perform binary and multi-class classification tasks independently. Each specialized model processes batched, tokenized inputs in its own inference pipeline with both models having the same prediction function but working sequentially. Instead, our multitask BERT approach employs a single unified model architecture that simultaneously outputs predictions for both classification tasks in one forward pass. Both implementations share core techniques: Batched processing for memory efficiency, gradient computation disabled during inference, tokenization with padding and truncation, logit-to-prediction conversion by argmax operations, and mapping numeric predictions back to human-readable labels. The multitask approach avoids duplicate processing while the two-model approach may allow more specialized optimization of each classification task.

5. Results and Discussion

Our team achieved significant success at the competitive event where research teams evaluated their machine learning models, securing **first place** in binary classification and **second place** in multi-class classification.

Comparing the performance of binary and multi-class classification provides deeper insights. RoBERTa showed strong performance in the binary classification task with a learning rate of 3e-5, while BERT achieved slightly higher accuracy at 2e-5. Specifically, `Roberta` achieved an accuracy value of 0.8723, while BERT achieved an accuracy value of 0.8748. These top-performing models consistently demonstrated strong performance across F1 scores, recall, and precision—both in macro and weighted metrics—suggesting they excel across various evaluation criteria.

In the multi-class classification situation, there is a slight shift in the performance rankings, with BERT (lr = 2e-5) achieving the highest accuracy of 0.7878, followed closely by `Deberta` with a lower learning rate of 2e-5 at 0.7752 accuracy. Performance metrics for multi-class tasks were lower than for binary classification, due to the increased difficulty of accurately distinguishing between multiple classes.

Macro metrics, which give equal weight to each class regardless of its size, and weighted metrics, which account for class imbalance, both offer valuable insight into model performance across different classification scenarios.

The tables below show detailed evaluation results across various transformer models using optimal learning rates found during hyperparameter tuning.

Table 7
Binary Classification Performance on Test Data (Sorted by Macro F1)

Model	M_Pr	M_Re	M_F1	W_Pr	W_Re	W_F1	Acc
BERT (lr = 2e-5)	0.8742	0.8743	0.8742	0.8748	0.8748	0.8748	**0.8748**
RoBERTa (lr = 3e-5)	0.8727	0.8744	0.8722	0.8753	0.8723	0.8724	**0.8723**
DeBERTa (lr = 2e-5)	0.8677	0.8646	0.8657	0.8672	0.8668	0.8665	0.8668
RoBERTa	0.8598	0.8588	0.8593	0.8600	0.8601	0.8600	0.8601
Multi-Head BERT (lr = 2e-5)	0.8572	0.8586	0.8558	0.8603	0.8559	0.8560	0.8559
Multi-Head BERT (lr = 5e-5)	0.8542	0.8533	0.8487	0.8585	0.8487	0.8486	0.8487
BERT (lr = 5e-5)	0.8488	0.8492	0.8490	0.8497	0.8496	0.8496	0.8496
DistilBERT (lr = 5e-5)	0.8412	0.8417	0.8382	0.8448	0.8382	0.8383	0.8382
Logistic Regression	0.8412	0.8417	0.8382	0.8448	0.8382	0.8383	0.8382

Table 8
Multi-Class Classification Performance on Test Data (Sorted by Macro F1)

Model	M_Pr	M_Re	M_F1	W_Pr	W_Re	W_F1	Acc
BERT (lr = 2e-5)	0.7590	0.7503	0.7515	0.7937	0.7878	0.7879	**0.7878**
DeBERTa (lr = 2e-5)	0.7431	0.7271	0.7322	0.7735	0.7752	0.7727	**0.7752**
RoBERTa (lr = 3e-5)	0.7282	0.7351	0.7280	0.7808	0.7626	0.7687	**0.7626**
Multi-Head BERT (lr = 2e-5)	0.7274	0.7401	0.7316	0.7737	0.7622	0.7665	**0.7622**
RoBERTa (lr = 5e-5)	0.7324	0.7325	0.7293	0.7748	0.7609	0.7654	**0.7609**
BERT (lr = 5e-5)	0.7405	0.7327	0.7320	0.7801	0.7727	0.7741	**0.7727**
Multi-Head BERT (lr = 5e-5)	0.7044	0.7449	0.7178	0.7708	0.7458	0.7535	**0.7458**
DistilBERT (lr = 5e-5)	0.7193	0.7142	0.7139	0.7612	0.7521	0.7545	**0.7521**
XGBoost	0.7482	0.7227	0.7219	0.7450	0.7328	0.7257	**0.7328**

These findings establish a strong foundation for understanding model performance; however, to further improve generalization, it is crucial to analyze the sources of misclassification. We examine this in the next section.

6. Error Analysis

In our study on detecting hope using three different methods, we found that TF-IDF combined with machine learning performed better than neural networks with fixed word embeddings. However, contextual embeddings showed the most impressive results overall. The success of the TF-IDF method can be attributed to its ability to accurately capture common language patterns through word frequency data in our dataset, as well as its effective utilization of traditional machine learning algorithms to handle sparse feature vectors. Moreover, on account of the reduced number of parameters that needed adjusting, these models were able to effectively learn from our limited dataset without experiencing

overfitting. In contrast, neural networks using static embeddings faced challenges in generalization, as they had inadequate training information and struggled to deal with sparse inputs.

Static word embeddings underperformed since they assign fixed vectors no matter context, limiting their ability to distinguish between various expressions of hope. Neural networks making use of these embeddings require substantial training information and substantial hyperparameter tuning to function well, conditions our project could not completely satisfy. Their black box nature even presented challenges for iterative improvement when compared with the greater interpretable TF IDF features, plus they lacked the correct inductive bias for our certain hope classification jobs, which might have depended much more on keyword patterns compared to complicated contextual understanding.

Because of the advanced transformer architecture as well as pre-training approach, contextual embeddings performed a lot better than the other options. These models generate flexible and responsive depictions, adjusting a word's significance depending on the text around it, enabling them to grasp subtle language nuances crucial for differentiating between various categories of hope. Their attention mechanisms with multiple layers effectively capture the connections between words at different distances, and their thorough pre-training on various datasets enabled successful transfer of knowledge to our specific tasks even with minimal training data. Their method of breaking down words into smaller units effectively dealt with unfamiliar terms, and also their ability to visually show where the model is focusing provided helpful information about how classifications are made, which makes them the perfect option for our sentiment analysis tasks.

7. Conclusion

This particular research provides a novel method of hope speech detection in social networking, expanding beyond conventional binary classification to a far more nuanced five category taxonomy. By leveraging advanced natural language processing methods, which includes transformer-based models and contextual embeddings, the study effectively distinguished between practical, generalized hope, unrealistic, and sarcastic expressions. The study achieved considerable performance metrics, with binary classification reaching as many as 87.48% accuracy and also multi class classification approaching 78.78%.

The study's main contribution is based on its sophisticated computational method for knowing the complicated emotional landscape of internet communication. By recording the slight contextual variants of hope, research offers invaluable insights into how individuals express hope in electronic environments. The addition of sarcasm detection adds depth to the evaluation, acknowledging the multifaceted nature of emotional expression in social networking. Ultimately, this particular work bridges technology and human emotion, providing a promising method of detecting and understanding optimistic content in a progressively digital world.

Declaration on Generative AI

Generative AI tools were used exclusively for language refinement and formatting. All research content, analysis, and conclusions were developed and verified by the authors. No AI-generated material influenced the scientific substance of this paper.

References

[1] G. Sidorov, F. Balouchzahi, L. Ramos, et al., Multilingual Identification of Nuanced Dimensions of Hope Speech in Social Media Texts, PREPRINT (Version 1) available at Research Square, 2025. URL: https://doi.org/10.21203/rs.3.rs-5338649/v1. doi:10.21203/rs.3.rs-5338649/v1.

[2] S. Butt, F. Balouchzahi, M. Amjad, S. M. Jiménez-Zafra, H. G. Ceballos, G. Sidorov, Overview of polyhope at iberlef 2025: Optimism, expectation or sarcasm?, Procesamiento del Lenguaje Natural (2025).

[3] S. Butt, F. Balouchzahi, A. Amjad, M. Amjad, H. G. Ceballos, S. M. Jiménez-Zafra, Optimism, expectation, or sarcasm? multi-class hope speech detection in spanish and english, https://doi.org/10.13140/RG.2.2.19761.90724, 2025. ResearchGate Preprint.
[4] J. Á. González-Barba, L. Chiruzzo, S. M. Jiménez-Zafra, Overview of IberLEF 2025: Natural Language Processing Challenges for Spanish and other Iberian Languages, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2025), co-located with the 41st Conference of the Spanish Society for Natural Language Processing (SEPLN 2025), CEUR-WS. org, 2025.
[5] B. R. Chakravarthi, Hopeedi: A multilingual hope speech detection dataset for equality, diversity, and inclusion, in: Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media, 2020, pp. 41–53.
[6] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, S. Cn, J. P. McCrae, M. Á. García, S. M. Jiménez-Zafra, R. Valencia-García, P. Kumaresan, R. Ponnusamy, et al., Overview of the shared task on hope speech detection for equality, diversity, and inclusion, in: Proceedings of the second workshop on language technology for equality, diversity and inclusion, 2022, pp. 378–388.
[7] D. García-Baena, M. Á. García-Cumbreras, S. M. Jiménez-Zafra, J. A. García-Díaz, R. Valencia-García, Hope speech detection in spanish: The lgbt case, Language Resources and Evaluation 57 (2023) 1487–1514.
[8] S. M. Jiménez-Zafra, M. Á. Garcia-Cumbreras, D. García-Baena, J. A. Garcia-Díaz, B. R. Chakravarthi, R. Valencia-García, L. A. Ureña-López, Overview of hope at iberlef 2023: Multilingual hope speech detection, Procesamiento del lenguaje natural 71 (2023) 371–381.
[9] F. Balouchzahi, G. Sidorov, A. Gelbukh, Polyhope: Two-level hope speech detection from tweets, Expert Systems with Applications 225 (2023) 120078.
[10] D. García-Baena, F. Balouchzahi, S. Butt, M. Á. García-Cumbreras, A. L. Tonja, J. A. García-Díaz, S. Bozkurt, B. R. Chakravarthi, H. G. Ceballos, R. Valencia-García, et al., Overview of hope at iberlef 2024: Approaching hope speech detection in social media from two perspectives, for equality, diversity and inclusion and as expectations, Procesamiento del lenguaje natural 73 (2024) 407–419.
[11] G. Sidorov, F. Balouchzahi, S. Butt, A. Gelbukh, Regret and hope on transformers: An analysis of transformers on regret and hope speech detection datasets, Applied Sciences 13 (2023) 3983.

Appendix

Table 9
Data Sample in Binary and Multi-Class Classification

Text Sample	Binary Classification	Multi-Class Classification
#USER# #USER# #USER# #USER# You expect a man that literally refers to himself as a god and is a blatantly obvious narcissist to be.. humble?üíÄ	Not Hope	Not Hope
#USER# #USER# #USER# #USER# Tinubu is actually a bonus. Lols	Not Hope	Not Hope
it'd be nice if missguided actually had stock for onceÔøÔøÔøÔøÔøÔøÔøÔøÔøÔøÔøÔøÔøÔø	Hope	Sarcasm
#USER# Anyway love u bubbly i know i can count on you when its about fairy tailüòåü§ö	Hope	Generalized Hope
,Äúyou have a lot of people rooting for you whether you believe it or not shouldn,Ä't want to let them down,Äù lmao i will cry please b quiet	Not Hope	Not Hope
Hoping to get on the brown a lot more after I move to the part of the city with a lot of bars	Not Hope	Not Hope
#USER# no exactly :/ and like i,Äôve seen (non gvfx) people say (after today,Äôs photos) omg you all need to move on from the hobicore aestheticüôÑüôÑand???	Not Hope	Not Hope
#USER# You have lost because your so called culture is backward never progressed Protestant working class don,Ä't aspire thru education catholic working class burst the glass ceiling every year whilst the working class Protestants stand still	Not Hope	Not Hope
#USER# #USER# #USER# #USER# Hi!üòä I,Äôm hoping to clearthelist of these items for my 1st graders	Hope	Realistic Hope
#USER# #USER# My dear Rwandans still remember what happened to them in 1994, so don,Ä't expect to forget so easily	Not Hope	Not Hope
"Well the club season is over. We ended up tied for 44th place out of 130 in the Aspire division at the AAU National Championships."	Hope	Generalized Hope
Find me a mass shooting in Utah where concealed carry on campus is legal	Not Hope	Sarcasm
#USER# I have been using 95 from the airport to the Delaware state line for over 20 years and it's always filthy	Not Hope	Not Hope
#USER# #USER# #USER# high time motu koitabians get behind a young nationalist who's sole desire is to take back what belong to indigenous motuans	Hope	Unrealistic Hope
#USER# #USER# And not expect someone else, aka the government, to pay for a child you obviously couldn,Ä't afford in the first place.	Not Hope	Not Hope
#USER# Today, I put my fave sweater on, hoping that this is my lucky day	Hope	Realistic Hope

Table 10
Combined Binary and Multi-Class Classification Distribution

Binary Classification	Multi-Class Classification	Total Count
Hope	Generalized Hope	1751
Hope	Realistic Hope	736
Hope	Sarcasm	195
Hope	Unrealistic Hope	643
Not Hope	Not Hope	3061
Not Hope	Sarcasm	749

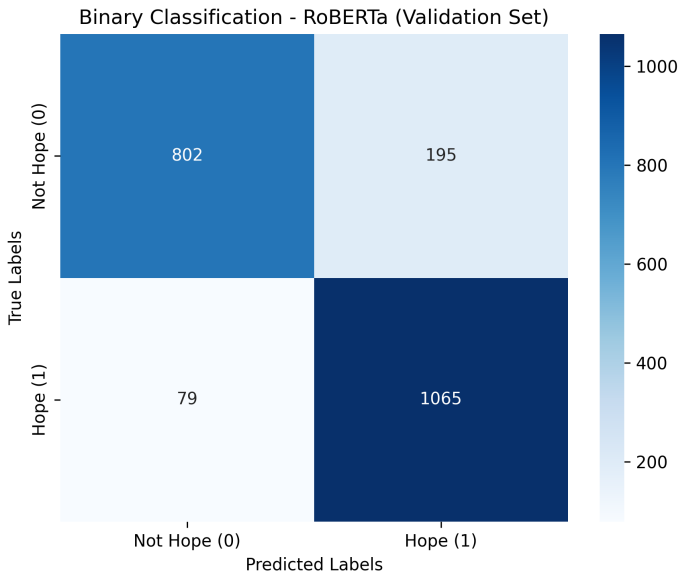

Figure 1: Confusion matrix for binary classification using RoBERTa on the validation set.

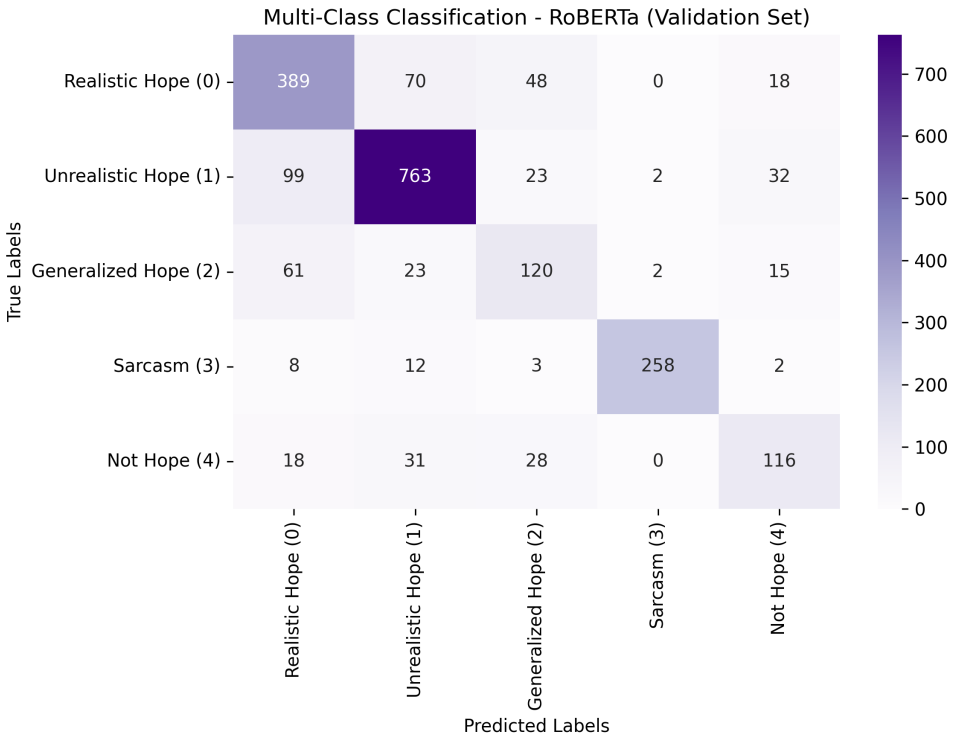

Figure 2: Confusion matrix for multi-class classification using RoBERTa on the validation set.