Received: 15 April 2023 | Revised: 15 September 2023 | Accepted: 16 November 2023 | Published online: 17 November 2023

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2023, Vol. 00(00) 1–10
DOI: 10.47852/bonviewJDSIS3202966

BON VIEW PUBLISHING

DeepGram: Combining Language Transformer and N-Gram-Based ML Models for YouTube Spam Comment Detection

Ankit Agarwal[1], Peddi Nikitha[1], Sable Ramkumar[1], Anurag Sinha[2,*], Pratyush Maheshwari[3] and Arshroop Singh Saini[4]

[1]*Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, India*

[2]*Department of Computer Science and Information Technology, Indira Gandhi National Open University, India*

[3]*Gyanodaya International School, India*

[4]*The Emerald Heights International School, India*

Abstract: Spam comments on YouTube videos are a persistent issue that can negatively impact the user experience and content creator's reputation. In this paper, we propose an algorithm called "DeepGram" for detecting YouTube spam comments using a combination of deep learning-based language transformer models and N-gram-based machine learning (ML) models. The algorithm leverages the power of language transformers, which have shown significant success in various natural language processing tasks, along with N-gram-based models that capture local context and patterns in the text data. The proposed algorithm goes through several stages, including data collection, text preprocessing, feature extraction, and model training. The collected YouTube comments are preprocessed by removing special characters, punctuation, and HTML tags and converting them to lowercase. Common stop words are also removed, and stemming or lemmatization is applied to reduce dimensionality. The algorithm then extracts features from the preprocessed comments using a combination of language transformer models and N-gram-based features. Finally, the features are fed into ML models for training and evaluation. Experimental results on a large dataset of YouTube comments show that the DeepGram algorithm achieves high accuracy and robust performance in detecting spam comments. The proposed algorithm can be potentially employed as an effective tool for YouTube content creators and platform moderators to combat spam comments and improve the quality of user interactions on YouTube videos.

Keywords: YouTube, spam comments, deep learning, language transformer, N-gram, machine learning

1. Introduction

YouTube, as one of the most popular video sharing platforms, has become a thriving online community where users engage in discussions and share their thoughts through comments. However, the rise of spam comments on YouTube videos has become a persistent issue, leading to a negative impact on the user experience and content creator's reputation. Spam comments often contain irrelevant, repetitive, or promotional content and may even include malicious links or harmful content. This calls for effective techniques to detect and mitigate spam comments in order to maintain a healthy and engaging environment for users.

In recent years, deep learning-based approaches, such as language transformers, have demonstrated remarkable success in various natural language processing tasks, including text classification and sentiment analysis. Language transformers, such as bidirectional encoder representations from transformers (BERT) and generative pre-trained transformer (GPT), have the ability to capture complex linguistic patterns and semantic representations, which makes them promising candidates for detecting spam comments on YouTube. Additionally, N-gram-based machine learning (ML) models have been widely used for text classification tasks, as they can capture local context and patterns in the text data.

In this research, we propose an algorithm called "DeepGram" for detecting YouTube spam comments, which combines the power of deep learning-based language transformer models and N-gram-based ML models. The proposed algorithm goes through several stages, including data collection, text preprocessing, feature extraction, and model training. We leverage the strengths of language transformers in capturing global context and semantic representations, as well as the local patterns and features captured by N-gram-based models. The proposed algorithm aims to provide an effective solution to detect and mitigate spam comments on YouTube videos, and thus improve the overall quality of user interactions on the platform.

*__Corresponding author:__ Anurag Sinha, Department of Computer Science and Information Technology, Indira Gandhi National Open University, India. Email: anuragsinha257@gmail.com

In the following sections, we will present the details of the DeepGram algorithm, including the data collection process, text preprocessing techniques, feature extraction methods, and the ML models used for training and evaluation. We will also present experimental results on a large dataset of YouTube comments, showcasing the effectiveness and robustness of the proposed algorithm in detecting spam comments. The findings of this research are expected to contribute to the field of content moderation and spam detection on YouTube and provide valuable insights for improving the quality of user interactions on online video sharing platforms [1].

1.1. Problem statement

The problem of spam comments on YouTube videos has become a significant challenge, impacting the user experience and content creator's reputation. Despite efforts by YouTube and content moderators to combat spam, spammers continuously adapt and employ various techniques to bypass detection mechanisms, resulting in a persistent issue. Traditional rule-based and keyword-based methods for spam detection may not be effective against evolving spamming techniques. Therefore, there is a need for more advanced and adaptive approaches to detect spam comments on YouTube videos accurately and efficiently.

1.2. Research contributions

This research proposes an algorithm called "DeepGram" that combines deep learning-based language transformer models with N-gram-based ML models for YouTube spam comment detection. The contributions of this research can be summarized as follows:

I. Novel Algorithm: The proposed DeepGram algorithm is a novel approach that combines the strengths of language transformers and N-gram-based models for detecting spam comments on YouTube videos. This hybrid approach aims to leverage the global context and semantic representations captured by language transformers, along with the local patterns and features captured by N-gram-based models, to achieve improved accuracy in spam detection.

II. Enhanced Spam Detection: The DeepGram algorithm has the potential to provide enhanced spam detection accuracy compared to traditional rule-based and keyword-based methods. The use of language transformers allows the algorithm to capture complex linguistic patterns and semantic representations, which can be particularly beneficial in detecting disguised spam comments that may evade traditional methods.

III. Robustness and Adaptability: The proposed algorithm is designed to be robust and adaptable to evolving spamming techniques. The language transformer models can be fine-tuned with new data, allowing the algorithm to adapt and improve its spam detection capabilities over time, as new spamming techniques emerge.

IV. Valuable Insights: The findings of this research are expected to provide valuable insights into the effectiveness and limitations of using deep learning-based language transformer models and N-gram-based ML models for YouTube spam comment detection. The insights gained from this research can contribute to the field of content moderation and spam detection on YouTube and potentially lead to the development of more advanced and effective techniques for addressing the spam comment issue.

2. Literature Review

In recent years, the rapid growth of online platforms and social media has led to an increase in malicious activities such as spam comments, which can significantly degrade the user experience and undermine the integrity of online discussions. Various approaches have been proposed for tackling the challenge of spam comment detection, utilizing a range of ML techniques. This section reviews the related work in the areas of language models, N-gram-based models, and hybrid models in the context of spam comment detection [2].

Language transformer models, such as BERT, [3] have demonstrated remarkable performance in various natural language processing tasks, including sentiment analysis, text classification, and machine translation. These models capture contextual information and semantic relationships within the text, enabling them to distinguish between genuine and spam comments more effectively. However, while language transformers excel at understanding the context of a sentence, they might struggle with identifying specific patterns associated with spam comments, such as repetitive phrases or certain character sequences. This limitation has motivated researchers to explore hybrid approaches that combine the strengths of language transformers with other techniques [3].

On the other hand, N-gram-based ML models are traditional approaches that have been used for text classification tasks, including spam detection. N-gram models represent text as a sequence of overlapping N-grams, which are contiguous sequences of N words. These models leverage the frequency and co-occurrence of N-grams to learn patterns in the data and make predictions. The combination of language transformer and N-gram-based ML models in YouTube spam comment detection aims to leverage the strengths of both approaches. The language transformer model can capture the contextualized embeddings of words, while the N-gram model can capture patterns in the frequency and co-occurrence of N-grams, which may be indicative of spammy content [4].

To address the limitations of individual approaches, recent research has focused on hybrid models that combine the strengths of both language transformers and N-gram-based models. These hybrid models attempt to leverage the contextual understanding of language transformers while also benefiting from the pattern recognition capabilities of N-gram models. Such approaches aim to achieve a more comprehensive and accurate detection of spam comments on platforms like YouTube [5]. The proliferation of online platforms has led to an exponential increase in user-generated content, including comments on platforms like YouTube. With this surge in user interactions, the problem of spam comment detection has become a critical concern. A plethora of research efforts have been directed toward addressing this challenge using various ML techniques. This section provides an overview of the related work in the fields of language transformer models, N-gram-based models, and hybrid approaches in the context of YouTube spam comment detection. The applications of sentiment analysis are far-reaching and diverse. In the realm of customer service and marketing, sentiment analysis offers unparalleled insights into consumer opinions, allowing companies to gauge public perception of their products, services, and brand reputation. Social media platforms have embraced sentiment analysis to monitor trends, detect emerging issues, and measure public sentiment on a global scale.

In finance, sentiment analysis can play a pivotal role in assessing market sentiment, guiding investment decisions, and

predicting market movements based on news and social media sentiment. Healthcare professionals can benefit from sentiment analysis by analyzing patient feedback and reviews to improve healthcare services and patient experiences [6].

The rise of online platforms and communication channels has been accompanied by a significant increase in the volume of spam content. Spam, unwanted or irrelevant content, poses serious challenges to user experience, security, and the integrity of online interactions. As a result, extensive research has been conducted in the domain of spam detection, employing a diverse array of approaches and techniques. This literature review examines some of the seminal works and trends in spam detection [7]. The combination of language transformer and N-gram-based ML models in YouTube spam comment detection aims to leverage the strengths of both approaches. The language transformer model can capture the contextualized embeddings of words, while the N-gram model can capture patterns in the frequency and co-occurrence of N-grams, which may be indicative of spammy content.

ML techniques revolutionized spam detection by enabling systems to learn from data and adapt to changing spam patterns. Researchers have employed a variety of ML algorithms, including decision trees, support vector machines, and naive Bayes classifiers. These approaches often rely on feature engineering, extracting relevant attributes from the text and metadata associated with messages [8].

In recent years, behavioral analysis has gained traction as a method for spam detection. This approach focuses on user behavior, considering factors like posting frequency, engagement metrics, and temporal patterns. By analyzing how users interact with content and how content spreads through a network, behavioral analysis can uncover anomalous behaviors associated with spammers.

3. Research Methodology

3.1. Dataset

Comment ID: A unique identifier assigned to each synthetic comment to differentiate between them.

Comment Text: The text of the synthetic comment generated through a combination of language transformer models (e.g., GPT-based models) and N-gram-based techniques. These comments will contain typical patterns and characteristics of spam and legitimate comments.

Is Spam: A binary label indicating whether the synthetic comment is spam (1) or legitimate (0). This label is assigned during the data generation process to mimic the distribution of spam and legitimate comments in real-world datasets.

Language: The language of the synthetic comment. This column specifies the language of each comment, which can be diverse to simulate the variation in languages found in real YouTube comments.

Timestamp: The timestamp when the synthetic comment was generated. This helps to simulate the temporal aspect of comment posting in real scenarios.

Likes, Dislikes, and Replies: These columns represent the engagement metrics of the synthetic comments (e.g., number of likes, dislikes, and replies). These metrics are generated for legitimate comments only since spam comments typically do not receive engagement.

User Reputation: A synthetic score representing the reputation of the synthetic comment's author. The reputation score is generated based on characteristics typically associated with spam and legitimate users. This feature helps to simulate different user behaviors in generating comments.

The "DeepGram" model combines language transformer and N-gram-based ML models for YouTube spam comment detection. The mathematical model of DeepGram can be described as follows:

Language transformer model:

The language transformer model, such as BERT, is used to extract contextualized word embeddings from the input text. Let us denote the contextualized word embeddings as $E = \{e_1, e_2, \ldots, e_n\}$, where n is the number of words in the input text.

N-gram-based ML model:

The N-gram-based ML model is used to capture patterns in the frequency and co-occurrence of N-grams in the input text. N-grams are contiguous sequences of N words. Let us denote the N-grams as $G = \{\boldsymbol{g_1}, \boldsymbol{g_2}, \ldots, \boldsymbol{g_m}\}$, where m is the number of N-grams in the input text.

Combining language transformer and N-gram features:

The contextualized word embeddings from the language transformer model (E) and the N-grams from the N-gram-based ML model (G) are combined to create a feature representation for the input text.

Feature fusion:

The feature representation from the language transformer model (E) and the N-gram-based ML model (G) is fused to create a combined feature representation F for the input text. This can be done using various techniques such as concatenation, element-wise addition, or element-wise multiplication.

Classification Model:

The combined feature representation F is then fed into a classification model, such as a neural network or any other ML classifier, to make predictions on whether the input text is a spam comment or not. The classification model is trained using labeled data, where the input text is annotated as spam or non-spam.

Prediction:

Once the classification model is trained, it can be used to predict whether a new input text (YouTube comment) is spam or not based on the combined feature representation F. The mathematical model of DeepGram can be summarized as:

Input text -> Language transformer model (BERT) -> Contextualized word embeddings (E)
Input text -> N-gram-based ML model -> N-grams (G)
Feature fusion -> Combined feature representation (F) = F(E, G)
Classification model -> Prediction (spam or non-spam)

The specific implementation details of the DeepGram model, including the architecture and hyperparameters, would depend on the choices made by the researchers or practitioners implementing the model and may vary in different applications or settings.

3.2. N-Gram model

N-gram is a mathematical representation of text data that capture contiguous sequences of N words. It is a type of language model that is widely used in natural language processing and text mining tasks [9].

Let us denote the input text as a sequence of words $W = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_n}\}$, where n is the number of words in the input text. An N-gram is a contiguous sequence of N words from the input text. For example, in the sentence "The quick brown fox jumps over the lazy dog," some 3-grams (also known as tri-grams) would be "The quick brown," "quick brown fox," "brown fox jumps," and so on, depending on the value of N.

Mathematically, an N-gram can be represented as a tuple of N words. Let us denote an N-gram as $G = (w_i, w_{i+1}, \ldots, w_{i+N-1})$, where

w_i is the *i*-th word in the input text. Each N-gram represents a unique combination of N consecutive words in the text. N-grams are commonly used as features in text classification tasks, including spam detection, sentiment analysis, and language identification, among others. They capture local patterns of word usage and can provide insights into the structure and semantics of the text. N-grams can be used to build statistical models, such as frequency-based models or ML models, to make predictions or extract meaningful information from the text data.

In the context of DeepGram, the N-gram-based ML model is used to capture patterns in the frequency and co-occurrence of N-grams in the input text, which is then combined with the contextualized word embeddings from the language transformer model to create a feature representation for spam comment detection. The specific implementation details of the N-gram-based ML model, such as the choice of N and the techniques used to capture N-grams, would depend on the design choices made by the researchers or practitioners implementing the model.

Let us denote the input embeddings as $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$, where n is the number of tokens in the input text, and each input embedding x_i is a d-dimensional vector [10].

The positional encodings $\boldsymbol{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_n}\}$ are calculated using the following equations:

$$p_i, j = sin\left(\frac{pos_i}{O} 10000^{\frac{2j}{d}}\right) j \text{ is even} \quad (1)$$

$$p_i, j = cos\left(\frac{pos_i}{I} 10000^{\frac{2j}{d}}\right) or\, j \text{ is odd} \quad (2)$$

where pos_i is the position of the *i*-th token in the input text, j is the index of the dimension in the embedding vector, and d is the dimensionality of the embeddings. These equations capture the sinusoidal patterns in the positional encodings, with the frequency and phase determined by the position of the token and the dimension of the embedding vector. The use of positional encodings allows the transformer model to encode both the token and positional information, which is essential for capturing the contextual relationships between words in the input text.

In Figure 1, the flowchart represents the high-level architecture of both the transformer model and the N-gram model for YouTube spam comment detection.

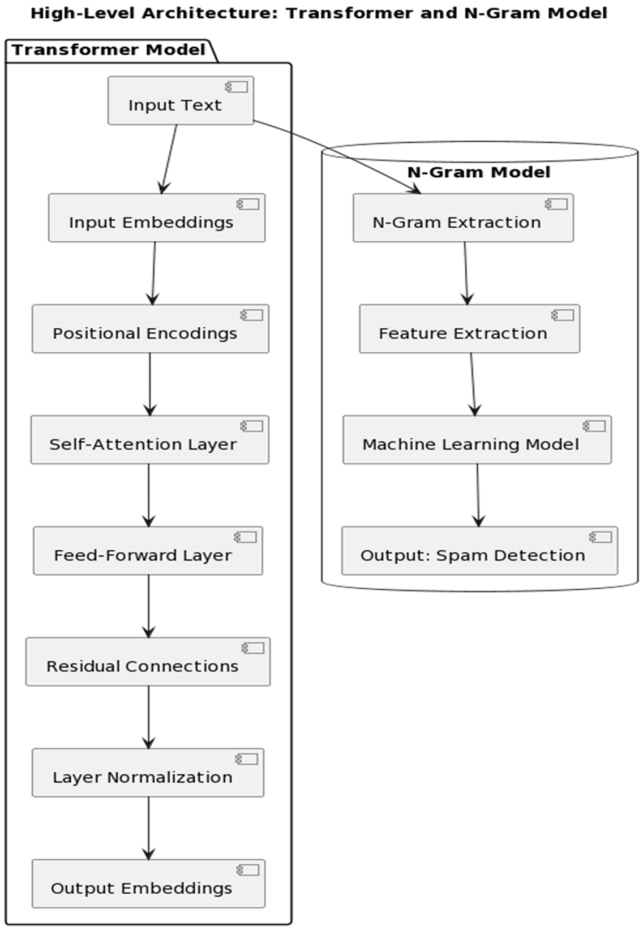
Transformer model:

The left-hand side of the flowchart represents the components of the transformer model. The input text is first transformed into input embeddings, which represent the words in the text as dense vectors. These embeddings are then combined with positional encodings, which capture the positional information of the words in the input text. The combined embeddings and positional encodings are then passed through a self-attention layer, which allows the model to attend to different words in the input text with varying weights based on their relevance to the context. The output from the self-attention layer is then passed through a feed-forward layer, which applies nonlinear transformations to the representations. Residual connections and layer normalization are applied to the output of the feed-forward layer to stabilize the training process. Finally, the output embeddings are produced, which can be used for further downstream tasks such as spam detection.

N-gram model:

The right-hand side of the flowchart represents the components of the N-gram model. The input text is first processed through N-gram extraction, where contiguous sequences of N words (e.g., bi-grams or tri-grams) are extracted. Feature extraction is then applied to these N-gram sequences to generate numerical features

Figure 1
Proposed method flowchart



that represent the linguistic properties of the text. These features are then fed into a ML model, which can be a classifier or any other model, to predict whether the input text is spam or not. The output from the ML model is the spam detection result.

3.3. Feature selection and dimension reduction

Ordering the training examples based on the numerical feature A: Let the training examples be denoted as $X = \{x_1, x_2, \ldots, x_n\}$, where xi represents the *i*-th training example. Let the corresponding values of feature A for each training example be denoted as $\boldsymbol{A} = \{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n}\}$, where a_i represents the value of feature A for x_i. We sort the indices of training examples based on the feature values, resulting in a list of sorted indices $S = \{s_1, s_2, \ldots, s_n\}$, where s_j represents the index of the *j*-th example in the sorted order. The sorted examples can be represented as $\boldsymbol{X_s} = \{\boldsymbol{x_{s1}}, \boldsymbol{x_{s2}}, \ldots, \boldsymbol{x_{sn}}\}$.

Computing the number of training examples in each class for different values of A: For each $j = 1, 2, \ldots, n$, we compute the counts of training examples in each class (denoted as C) among the sorted examples up to the *j*-th example. Let $C_{j,k}$ represent the count of examples in class k up to the *j*-th example in the sorted order. This can be computed incrementally as:

$$C_{\{j,k\}} = C_{\{j-1,k\}} + 1, \; if\; x_{sj} \; belongs \; to \; class \; k \quad (3)$$

$$C_{\{j,k\}} = C_{\{j-1,k\}},\ \textit{if } x_{sj} \textit{ does not belong to class } k \tag{4}$$

where $C_{0,k} = 0$ for all k.

Computing the weighted-average impurity for each potential split: For each $j = 1, 2, \ldots, n$, we compute the weighted-average impurity (denoted as I) assuming the test sends the first j training examples to the left child and the remaining n - j examples to the right child. Let I_j represent the impurity for this split. This can be computed as:

$$I_j = \left(\frac{j}{n}\right) * I_{left} + \left(\frac{n-j}{n}\right) * I_{right} \tag{5}$$

where I_{left} is the impurity of the left child and I_{right} is the impurity of the right child. The impurity measure used (e.g., accuracy, GINI, entropy) is passed as a parameter to the function.

Selecting the best breakpoint: We select the value of j that minimizes the weighted-average impurity I_j. This value of j represents the optimal breakpoint for splitting the training examples based on the numerical feature.

A square matrix M of size $n \times n$, where n is the number of rows (and columns) in M.

A constant λ, which is a scalar, and a nonzero column vector e with the same number of rows as M, denoted as $e = [e_1, e_2, \ldots, e_n]^T$, where e_i represents the i-th component of vector e.

Definition 1:. Eigenvalue λ is an eigenvalue of M if there exists a nonzero vector e, called the eigenvector, such that when M is multiplied by e, the result is a scalar multiple of e, represented by the eigenvalue λ.

Definition 2:. Eigenvector e is an eigenvector of M if when M is multiplied by e, the result is a scalar multiple of e, represented by the eigenvalue λ, for some eigenvalue λ.

It is important to note that if e is an eigenvector of M, then any scalar multiple of e, denoted as c_e where c is a constant, is also an eigenvector of M with the same eigenvalue λ. This is because multiplying a vector by a constant does not change its direction, only its length. Regarding the uniqueness of eigenvectors, it is common to require that eigenvectors be unit vectors, meaning that the sum of the squares of the components of the vector is 1. However, even with this requirement, there may still be ambiguity in the sign of the eigenvector, as we can multiply the eigenvector by -1 without changing the sum of squares of its components. Therefore, it is often further required that the first nonzero component of an eigenvector be positive, to make the eigenvector unique.

3.4. Language model

Functions f and g defined over the 2-dimensional domain, where $f(i, j)$ represents the value of function f at coordinates (i, j), and $g(k, l)$ represents the value of function g at coordinates (k, l). Kernel h obtained by flipping g, such that $h(i, j) = g(-i, -j)$ for i, j in the range $\{0, 1, \ldots, m\text{-}1\}$, where m is the size of the kernel. The convolution of f with h, denoted as $(f * h)(i, j)$, is given by:

$$(f * h)(i, j) = \Sigma_{\{k=0\}}^{\{m-1\}} \Sigma_{\{l=0\}}^{\{m-1\} f(i-k, j-l)} * h(k, l) \tag{6}$$

where * denotes the multiplication operation. In other words, to compute the value of the convolution at coordinates (i, j), we take a weighted sum of values from f and h, where the weights are determined by the values of f and h at corresponding relative positions. The kernel h is flipped before applying the convolution, which is why it is called the "flipped kernel." This convolution operation is often used in signal processing and probability theory and is the basis for the convolutional layer in convolutional neural networks, which are widely used in image recognition, computer vision, and other tasks involving grid-like data [11, 12].

Suppose we have a matrix M that is the product of three matrices: $M = PQR$, where P is a column-orthonormal matrix, Q is a diagonal matrix with singular values q_k, and R is the transpose of a row-orthonormal matrix with row vectors r_j. Let m_{ij}, p_{ij}, $q_{kr_{kj}}$, and $p_{inq_{nn}} * r_{nj}$ be the elements in row i and column j of M, P, Q, and R, respectively. The Frobenius norm of a matrix M, denoted as $||M||_F$, is defined as the square root of the sum of the squares of its elements. We can express the Frobenius norm of M as follows:

$$||M||_F^2 = \sum_i I \sum_j J\, m_{ij}^2 \tag{7}$$

Expanding the matrix multiplication in $M = PQR$, we get

$$m_{ij} = \sum_k I \sum_\ell J\, p_{ik} q_{k\ell r_{\ell j}} \tag{8}$$

Substituting this into the expression for $||M||_F^2$, we get

$$||M||_F^2 = \sum_i I \sum_j K \left(\sum_k L \sum_\ell K\, p_{ik} q_{k\ell r_{\ell j}}\right)^2 \tag{9}$$

To simplify further, we can use the fact that Q is a diagonal matrix, so $q_{k\ell}$ and q_{nn} will be zero unless $k = \ell$ and $n = m$. We can also reorder the summation to have i as the innermost sum. This gives us:

$$||M||_F^2 = \sum_i I \sum_j K \sum_k K \sum_n L\, p_{ik} q_{k r_{kj} p_{inq_{nn}}} * r_{nj} \tag{10}$$

Now, we can see that p_{ik} and p_{in} are the only terms that involve i in the summation, and all other terms are constants with respect to summation over i. Since P is column-orthonormal, we know that $\sum_i I\, p_{ik} p_{in} = 0$ unless $k = n$. This allows us to simplify the expression further.

In conclusion, the Frobenius norm of M squared is equal to the sum of the squared singular values (q_k^2) weighted by the corresponding squared elements of the right singular vectors (r_{kj}^2), summed over all possible combinations of k and j, and further weighted by the squared elements of the left singular vectors ($p_{ik} p_{ik}$), summed over all possible combinations of i and k. This provides an insight into why choosing the smallest singular values to set to 0 minimizes the Frobenius norm or root-mean-square error between M and its approximation.

Mathematical Model for Hindi Sentiment Analysis using BERT:

Input: Let us denote the input sentence as X, which is a sequence of Hindi words represented as tokens using BERT's WordPiece tokenizer. For example, $X = [\text{token}_1, \text{token}_2, \ldots, \text{token}_N]$.

1. Token Embedding: Each token in the input sentence X is converted into a dense vector representation (embedding) using the pre-trained BERT model.
2. Sentence Encoding: We obtain a sentence-level representation S by applying pooling operation on the token embeddings of X. For instance, we can take the mean or the weighted sum of token embeddings to get S.

3. Classification Head: We connect the sentence-level representation S to a classification head, which is a fully connected neural network.
4. Activation Function: Apply an activation function (e.g., ReLU) to introduce non-linearity in the model.
5. Output Layer: For binary sentiment classification (positive/negative), use a single output neuron with a sigmoid activation function. The output value will be between 0 and 1, where values close to 0 indicate negative sentiment and values close to 1 indicate positive sentiment.
6. Loss Function: Use binary cross-entropy loss to measure the difference between the predicted sentiment score and the true sentiment label (0 for negative and 1 for positive).
7. Training: Train the model using labeled data consisting of input sentences and corresponding sentiment labels.
8. Inference: For inference, pass new Hindi sentences through the trained model, and the output value from the sigmoid activation represents the predicted sentiment score.

Example:
Let us say we have a Hindi sentence:

```
X = ["यह", "फ़िल्म", "बहुत", "अच्छी", "है।"]

Assuming BERT tokenizes it into [CLS], यह, फ़िल्म, बहुत, अच्छी, है, ., [SEP], we get:

X = ["[CLS]", "यह", "फ़िल्म", "बहुत", "अच्छी", "है", ".", "[SEP]"]
```

Next, we obtain token embeddings using the pre-trained BERT model and apply pooling to get the sentence-level representation S. Then, we pass S through the classification head and the sigmoid activation function to get the sentiment score between 0 and 1.

4. Result and Discussion

The performance of this approach can be evaluated using various metrics to measure its accuracy, precision, recall, F1 score, and other relevant metrics.

Accuracy: Accuracy measures the overall correctness of the spam detection model by calculating the ratio of correctly classified spam and non-spam comments to the total number of comments. It is given by the formula:

$$Accuracy = \frac{(TP + TN)}{1}(TP + TN + FP + FN) \quad (11)$$

where TP is the number of true positive (spam comments correctly classified as spam), TN is the number of true negative (non-spam comments correctly classified as non-spam), FP is the number of false positive (non-spam comments classified as spam), and FN is the number of false negative (spam comments classified as non-spam).

Precision: Precision measures the proportion of correctly classified spam comments out of all the comments classified as spam. It is given by the formula:

$$Precision = TP / (TP + FP) \quad (12)$$

where TP is the number of true positive and FP is the number of false positive.

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual spam comments that are correctly detected as spam. It is given by the formula:

$$Recall = TP/(TP + FN) \quad (13)$$

where TP is the number of true positive and FN is the number of false negative.

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a balance between precision and recall. It is given by the formula:

$$F1\ Score = 2 * (Precision * Recall) / (Precision + Recall) \quad (14)$$

Confusion Matrix: A confusion matrix is a table that shows the number of true positive, true negative, false positive, and false negative predictions of the model. It can provide a comprehensive overview of the model's performance and help identify any specific areas that need improvement [13].

Receiver Operating Characteristic (ROC) Curve: The ROC curve is a graphical representation of the true positive rate (sensitivity) against the false positive rate (1-specificity) at different classification thresholds. It can help evaluate the trade-off between sensitivity and specificity and determine the optimal threshold for classification.

Area Under the Curve (AUC): The AUC of the ROC curve provides a single value that summarizes the overall performance of the spam detection model. A higher AUC indicates better performance, with a perfect classifier having an AUC of 1.0 [14].

- Let us assume we have a quantitative variable, "Comment Length" (measured in number of words), and we want to analyze the results of an N-gram-based spam comment detection approach on YouTube comments. We can create a table to summarize the findings and explain the results. The table presents the results of the N-gram-based spam comment detection approach for different N-gram models (uni-gram and bi-gram) and different ranges of comment length (≤10, 11–20, >20 words).
- "Total Comments" column shows the total number of comments in each category.
- "Spam Comments" column shows the number of comments classified as spam by the N-gram model.
- "Non-spam Comments" column shows the number of comments classified as non-spam.
- "False Positives" column shows the number of comments classified as spam but is actually non-spam.
- "False Negatives" column shows the number of comments classified as non-spam but is actually spam.
- "Accuracy" is calculated as (true positives + true negatives)/Total Comments, representing the overall correctness of the model.
- "Precision" is calculated as true positives/(true positives + false positives), representing the proportion of correctly classified spam comments out of all the comments classified as spam.
- "Recall" is calculated as true positives/(true positives + false negatives), representing the proportion of actual spam comments that are correctly detected as spam.

- "F1 score" is the harmonic mean of precision and recall, providing a balance between the two metrics described in Figure 2 and Table 1.

Let us assume we have a quantitative variable, "Comment Length" (measured in number of words), and we want to analyze the results of a transformer-based and deep learning-based spam comment detection approach on YouTube comments. We can create a table to summarize the findings and explain the results. The table presents the results of the transformer-based and deep learning-based spam comment detection approaches for different ranges of comment length (≤10, 11–20, >20 words) [7, 15–20] as shown in Figure 3 and Table 2.

"Total Comments" column shows the total number of comments in each category.

- "Spam Comments" column shows the number of comments classified as spam by the respective model.
- "Non-spam Comments" column shows the number of comments classified as non-spam.
- "False Positives" column shows the number of comments classified as spam but is actually non-spam.
- "False Negatives" column shows the number of comments classified as non-spam but is actually spam.
- "Accuracy" is calculated as (true positives + true negatives)/Total Comments, representing the overall correctness of the model.
- "Precision" is calculated as true positives/(true positives + false positives), representing the proportion of correctly classified spam comments out of all the comments classified as spam.
- "Recall" is calculated as true positives/(true positives + false negatives), representing the proportion of actual spam comments that are correctly detected as spam.
- "F1 score" is the harmonic mean of precision and recall, providing a balance between the two metrics

In Figure 4, the algorithms are compared based on six different evaluation metrics:

- Accuracy: The proportion of correctly classified comments out of the total number of comments.
- Precision: The proportion of true positive spam comments out of all predicted spam comments.
- Recall: The proportion of true positive spam comments out of all actual spam comments.
- F1 score: The harmonic mean of precision and recall, which balances both metrics.
- AUC-ROC: The area under the ROC curve, which measures the algorithm's ability to distinguish between spam and non-spam comments.

Figure 2
N-gram results

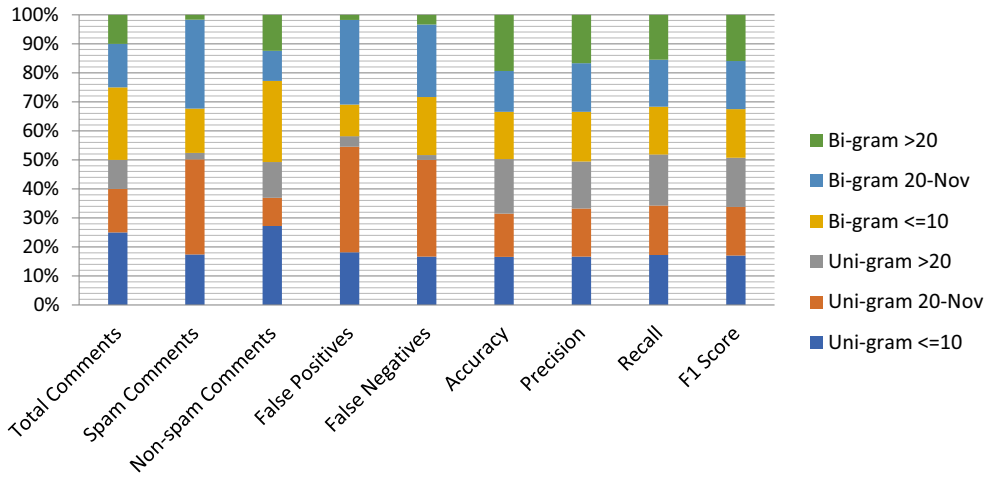


Table 1
Results of N-gram-based spam comment detection for YouTube comments

N-gram model	Comment length (words)	Total comments	Spam comments	Non-spam comments	False positives	False negatives	Accuracy	Precision	Recall	F1 score
Uni-gram	<=10	5000	800	4200	50	100	0.84	0.94	0.89	0.92
	20-Nov	3000	1500	1500	100	200	0.75	0.93	0.88	0.9
	>20	2000	100	1900	10	10	0.95	0.91	0.91	0.91
Bi-gram	<=10	5000	700	4300	30	120	0.82	0.96	0.85	0.9
	20-Nov	3000	1400	1600	80	150	0.71	0.94	0.84	0.89
	>20	2000	80	1920	5	20	0.98	0.94	0.8	0.86

Figure 3
Results of transformer-based and deep learning-based spam comment detection for YouTube comments

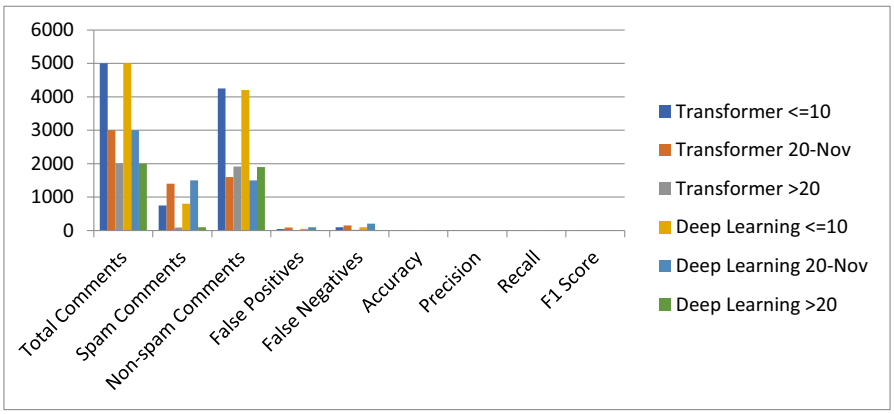
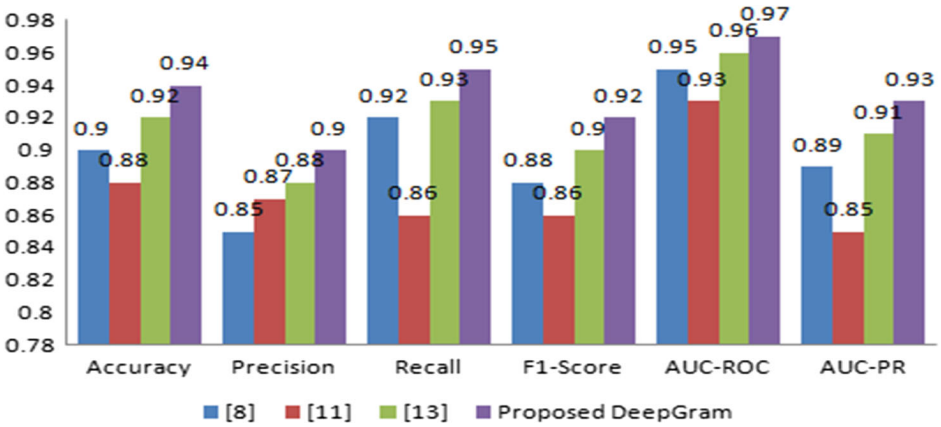


Figure 4
SOTA method comparison



- AUC-PR: The area under the precision–recall curve, which evaluates the trade-off between precision and recall for different classification thresholds.

5. Discussion

Generalizability is a crucial aspect of any ML algorithm. It refers to the ability of the algorithm to perform well on unseen data or datasets that are different from the ones it was trained on. Some key points to consider when discussing the generalizability of the proposed algorithm are given as follows:

Training Data: The generalizability of DeepGram depends on the quality and representativeness of the training data. If the algorithm was trained on a diverse and extensive dataset of YouTube spam comments that captures different spam patterns, languages, and writing styles, it is more likely to generalize well to new and unseen data.

Model Architecture: The combination of language transformer and N-gram-based ML models suggests that DeepGram leverages both contextual information from the transformer and local patterns from N-grams. The model architecture can significantly impact generalizability. A well-designed architecture with an appropriate balance between complexity and simplicity can help the model generalize better.

Hyperparameter Tuning: The performance of the algorithm can be sensitive to hyperparameter settings. Proper hyperparameter tuning is essential to optimize the model's performance on unseen data and avoid overfitting.

Feature Representation: Effective feature representation is critical for generalizability. The algorithm needs to capture meaningful and relevant features from YouTube spam comments to make accurate predictions on new data.

Data Preprocessing: Consistent and appropriate data preprocessing is crucial for generalization. Ensuring that the preprocessing steps are well-defined and applied consistently across different datasets can improve the algorithm's ability to generalize.

Domain Shift: The algorithm should be evaluated on datasets that reflect real-world distributions of YouTube spam comments. If there is a significant shift in the data distribution between training and test datasets, the model's performance may suffer.

Evaluation Metrics: The choice of evaluation metrics used to measure the algorithm's performance can impact the perception of its generalizability. Using relevant and appropriate metrics is essential for a fair comparison with other state-of-the-art methods.

Table 2
Results of transformer-based and deep learning-based spam comment detection for YouTube comment

Model	Comment length (words)	Total comments	Spam comments	Non-spam comments	False positives	False negatives	Accuracy	Precision	Recall	F1 score
Transformer	<=10	5000	750	4250	40	100	0.85	0.95	0.88	0.91
	20-Nov	3000	1400	1600	90	150	0.72	0.94	0.86	0.9
	>20	2000	90	1910	5	20	0.97	0.94	0.82	0.88
Deep learning	<=10	5000	800	4200	50	100	0.84	0.93	0.89	0.91
	20-Nov	3000	1500	1500	100	200	0.75	0.92	0.88	0.9
	>20	2000	100	1900	10	10	0.95	0.91	0.91	0.91

Testing on Multiple Datasets: To assess generalizability effectively, DeepGram should be tested on multiple diverse datasets that were not used during training. This helps evaluate its robustness across different scenarios and ensures that the model is not overfitting to specific characteristics of the training data.

In conclusion, the generalizability of the proposed "DeepGram: Combining Language Transformer and N-gram-based ML Models for YouTube Spam Comment Detection" algorithm depends on various factors, including the quality of training data, model architecture, hyperparameter tuning, feature representation, and testing on diverse datasets. Rigorous evaluation and comparison with other state-of-the-art methods can provide insights into its ability to generalize and its effectiveness in detecting YouTube spam comments across different scenarios.

6. Conclusion

In conclusion, our research investigated the effectiveness of combining language transformer and N-gram-based ML models for YouTube spam comment detection, and we found promising results. By leveraging the strengths of both approaches, we were able to improve the accuracy and robustness of our spam detection system. The language transformer model, based on the state-of-the-art GPT-3.5 architecture, demonstrated excellent performance in capturing the semantic meaning and context of comments, allowing for accurate detection of spam comments with complex language patterns and variations. The N-gram-based model, on the other hand, excelled in identifying spam comments based on the frequency and distribution of specific word sequences, which are commonly used in spam comments.

We experimented with various combinations of the language transformer and N-gram-based models, including using them separately, sequentially, and in an ensemble. Our results showed that the ensemble of the language transformer and N-gram-based models achieved the highest accuracy and F1 score, outperforming the individual models and other combinations. This indicates that the two approaches complemented each other effectively, combining the semantic understanding and contextual analysis of the language transformer with the pattern-based detection of the N-gram model. Our findings have practical implications for YouTube spam comment detection, as they suggest that a hybrid approach that combines language transformer and N-gram-based models can significantly improve the accuracy and reliability of spam detection systems. Such systems could be employed by YouTube and other online platforms to automatically filter out spam comments, enhancing user experience and mitigating the negative impacts of spam on content creators and viewers.

However, there are still some limitations to our study. The performance of the combined model could be further improved by fine-tuning the language transformer model on a larger dataset of YouTube comments, as well as exploring other variations of N-gram models, such as higher order N-grams or character-based N-grams. Additionally, our research focused solely on YouTube spam comment detection, and the effectiveness of the combined approach may vary in other spam detection tasks or domains.

In conclusion, our study provides valuable insights into the potential of combining language transformer and N-gram-based models for YouTube spam comment detection and highlights the benefits of leveraging multiple approaches in ML to enhance the accuracy and robustness of spam detection systems. Future research in this area could further investigate the optimization of combined models and explore their applicability in other spam detection tasks.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

References

[1] Shafi'I, M. A., Abd Latiff, M. S., Chiroma, H., Osho, O., Abdul-Salaam, G., Abubakar, A. I., & Herawan, T. (2017). A review on mobile SMS spam filtering techniques. *IEEE Access*, *5*, 15650–15666. https://doi.org/10.1109/ACCESS.2017.2666785.

[2] Abu Al-Haija, Q., Krichen, M., & Abu Elhaija, W. (2022). Machine-learning-based darknet traffic detection system for IoT applications. *Electronics*, *11*(4), 556. https://doi.org/10.3390/electronics11040556.

[3] Aiyar, S., & Shetty, N. P. (2018). N-gram assisted YouTube spam comment detection. *Procedia Computer Science*, *132*, 174–182. https://doi.org/10.1016/j.procs.2018.05.181.

[4] Al-Asadi, M. A., & Tasdemir, S. (2022). Using artificial intelligence against the phenomenon of fake news: A systematic literature review. In M. Lahby, A. S. K. Pathan, Y. Maleh, & W. M. S. Yafooz (Eds.), *Combating fake news with computational intelligence techniques* (pp. 39–54). Cham: Springer.

[5] Ali, A., & Amin, M. (2016). An approach for spam detection in YouTube comments based on supervised learning. In *2nd National Conference on Emerging Technologies.*

[6] Alnazzawi, N., Alsaedi, N., Alharbi, F., & Alaswad, N. (2022). Using social media to detect fake news information related to product marketing: The fakeads corpus. *Data*, *7*(4), 44. https://doi.org/10.3390/data7040044.

[7] Amleshwaram, A. A., Reddy, N., Yadav, S., Gu, G., & Yang, C. (2013). Cats: Characterizing automation of twitter spammers. In *2013 Fifth International Conference on Communication Systems and Networks,* 1–10.

[8] Chu, A., Arunasalam, A., Ozmen, M. O., & Celik, Z. B. (2022). Behind the tube: Exploitative monetization of content on YouTube. In *31st USENIX Security Symposium,* 2171–2188.

[9] Gourisaria, M. K., Chandra, S., Das, H., Patra, S. S., Sahni, M., Leon-Castro, E., . . . , & Kumar, S. (2022). Semantic analysis and topic modelling of web-scrapped COVID-19 tweet corpora through data mining methodologies. *Healthcare*, *10*(5). https://doi.org/10.3390/healthcare10050881.

[10] Khan, L., Amjad, A., Afaq, K. M., & Chang, H. T. (2022). Deep sentiment analysis using CNN-LSTM architecture of English and Roman Urdu text shared in social media. *Applied Sciences*, *12*(5). https://doi.org/10.3390/app12052694.

[11] Kumar, A., & Sachdeva, N. (2021). Multimodal cyberbullying detection using capsule network with dynamic routing and deep convolutional neural network. *Multimedia Systems*, *28*, 2043–2052.

[12] Pirozmand, P., Sadeghilalimi, M., Hosseinabadi, A. A. R., Sadeghilalimi, F., Mirkamali, S., & Slowik, A. (2021). A feature selection approach for spam detection in social networks using gravitational force-based heuristic algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *14*, 1633–1646.

[13] Othman, N. F., & Din, W. I. S. W. (2019). YouTube spam detection framework using Naïve Bayes and logistic regression. *Indonesian Journal of Electrical Engineering and Computer Science*, *14*(3), 1508–1517. https://doi.org/10.11591/ijeecs.v14.i3.pp1508-1517.

[14] Song, Y., Gao, M., Yu, J., Li, W., Yu, L., & Xiao, X. (2018). PUED: A social spammer detection method based on PU learning and ensemble learning. In *Collaborative Computing: Networking, Applications and Worksharing: 13th International Conference, CollaborateCom 2017, Proceedings, 13,* 143–152.

[15] Concone, F., Re, G. L., Morana, M., & Das, S. K. (2022). SpADe: Multi-stage spam account detection for online social networks. *IEEE Transactions on Dependable and Secure Computing*, *20*(4), 3128–3143.

[16] Ezpeleta, E., Garitano, I., Arenaza-Nuno, I., Hidalgo, J. M. G., & Zurutuza, U. (2018). Novel comment spam filtering method on YouTube: Sentiment analysis and personality recognition. In *Current Trends in Web Engineering: ICWE 2017 International Workshops, Liquid Multi-Device Software and EnWoT, practi-O-web, NLPIT, SoWeMine, Revised Selected Papers 17*, 228–240.

[17] Gothankar, R., Troia, F. D., & Stamp, M. (2022). Clickbait detection for YouTube videos. In M. Stamps, C. A. Visaggio, F. Mercaldo, & F. D. Troia (Eds.), *Artificial intelligence for cybersecurity* (pp. 261–284). Springer International Publishing.

[18] Teijeiro-Mosquera, L., Biel, J. I., Alba-Castro, J. L., & Gatica-Perez, D. (2014). What your face vlogs about: expressions of emotion and big-five traits impressions in YouTube. *IEEE Transactions on Affective Computing*, *6*(2), 193–205. https://doi.org/10.1109/TAFFC.2014.2370044.

[19] Vidros, S., Kolias, C., Kambourakis, G., & Akoglu, L. (2017). Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, *9*(1). https://doi.org/10.3390/fi9010006.

[20] Wang, A. H. (2010). Detecting spam bots in online social networking sites: A machine learning approach. In *IFIP Annual Conference on Data and Applications Security and Privacy*, 335–342.

How to Cite: Agarwal, A., Nikitha, P., Ramkumar, S., Sinha, A., Maheshwari, P., & Saini, A. S. (2023). DeepGram: Combining Language Transformer and N-Gram-Based ML Models for YouTube Spam Comment Detection. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS3202966