Yahoo News Comments Classifier: NLP-Based Classification of Yahoo Comments for Enhanced Content Moderation

SAIRAJ KUCHANA, Old Dominion University, USA
HEMANT KUMAR, Old Dominion University, USA
HARSHIT KONCHARLA, Old Dominion University, USA

ACM Reference Format:
Sairaj Kuchana, Hemant Kumar, and Harshit Koncharla. 2024. Yahoo News Comments Classifier: NLP-Based Classification of Yahoo Comments for Enhanced Content Moderation. 1, 1 (May 2024), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

This research presents a novel approach to comment classification utilizing Natural Language Processing (NLP) techniques, particularly focusing on extracting comments from Yahoo News. The objective is to develop a model capable of classifying comments into six distinct categories: Ideological, Humorous, Consolidating, Abusive, Spam and Neutral. Leveraging machine learning algorithms and NLP methodologies, our study contributes to the growing field of comment classification, providing insights into automated content moderation and facilitating a better understanding of online discourse dynamics.

2 INTRODUCTION

The emergence of social media and online discussion boards has completely changed how people interact, work together, and share ideas. These online forums act as virtual gathering places for people from all walks of life to talk about anything from politics and entertainment to technology and culture. In these virtual environments, comments are an essential type of user-generated content that provide people with the chance to voice their thoughts, exchange viewpoints, and converse with others.

However, the open and decentralized nature of online platforms also presents inherent challenges, chief among them being the task of effectively moderating the vast volumes of user-generated content.[3]The anonymity and accessibility afforded by these platforms can sometimes lead to the proliferation of harmful or inappropriate comments, ranging from abusive language and hate speech to spam and irrelevant content. Addressing these challenges requires scalable and efficient solutions that leverage advancements in computational linguistics and machine learning.

A promising approach to automating content moderation is comment categorization, which gives platforms the ability to classify comments according to their content and take the necessary precautions to protect the safety and integrity of online communities. Through the application of natural language processing (NLP) tools, scholars and programmers

Authors' addresses: Sairaj Kuchana, Old Dominion University , 1 Thørväld Circle, Norfolk, USA, skuch001@odu.edu; Hemant Kumar, Old Dominion University, Norfolk, USA, hkuma001@odu.edu; Harshit Koncharla, Old Dominion University, Norfolk, USA, hkonc001@odu.edu.

Manuscript submitted to ACM

can examine the grammatical patterns, semantic indicators, and contextual subtleties included in comments to ascertain their fundamental meaning and group them into appropriate categories.

In this paper, we present a comprehensive methodology for developing a comment classification model tailored specifically to Yahoo comments. Our approach encompasses data collection, preprocessing, feature extraction, model development, and evaluation, culminating in the deployment of a robust and scalable system capable of categorizing comments into six distinct categories. Through empirical analysis and experimentation, we aim to demonstrate the efficacy of our approach and contribute to the advancement of automated content moderation systems, thereby fostering healthier, more inclusive, and vibrant online communities.

3 RELATED WORK

In order to classify and evaluate information, text classification and natural language processing techniques have been thoroughly researched and used in a variety of disciplines. Using various methods, prior research has significantly improved content classification.

In a well-known work, [10] suggested a topic modeling approach for categorizing user-generated content in online forums using Latent Dirichlet Allocation (LDA). Their study showed how well probabilistic models work for finding and classifying themes in online discussion forums.

In a similar vein, [2] created a sentiment analysis technique to categorize messages on social media sites like Reddit into positive, negative, and neutral sentiment categories. Their research demonstrated how sentiment analysis methods can be used to identify and categorize the attitudes and sentiments of users.

Although the field of content classification has evolved as a result of these studies, our study takes a different approach by using a keyword-based methodology. Comments can be categorized into six different categories using the YahooNewsCommentsClassifier: Ideological, Humorous, Consolidating, Abusive, Spam, and Neutral. With content that is personalized to each user's interests, preferences, and information needs, this user-centric approach seeks to improve content discoverability and user happiness.

Previous studies have looked into a number of methods for detecting hate speech and classifying content on internet platforms. A study [11]looked into the identification of hate speech in social media posts using machine learning methods, particularly Support Vector Machines (SVMs) and Random Forests. Their research demonstrated how crucial feature engineering is to enhancing classification accuracy, as is the selection of pertinent language and contextual variables.

The use of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks in a deep learning-based technique for hate speech identification is another noteworthy contribution by [7]. Their study showed how these neural network designs might be used to better classify hate speech by identifying intricate patterns and contextual information in textual data.

[1] created a semi-supervised learning system in the content moderation field to identify abusive language in internet forums. Their method addressed the difficulty of obtaining enough labeled examples for training machine learning models in this situation by combining a small quantity of labeled data with sizable unlabeled datasets.

Even though these studies have made a substantial impact, there are still issues with how to deal with the dynamic and ever-changing nature of internet content as well as the complexity brought on by linguistic and cultural differences. By combining domain-specific information and utilizing the most recent developments in machine learning and natural language processing, our research seeks to improve upon these previous methods in order to create a reliable and flexible content classification system.

Our work aims to fill in the gaps and limits found in the literature by analyzing and integrating the results of earlier studies, thereby advancing the creation of more dependable and efficient content classification techniques for online platforms.

4 METHODOLOGY

Our methodology encompasses several key phases aimed at developing a comment classification model tailored specifically to Yahoo comments, as outlined below:

Data Collection: We initiate the process by collecting a substantial dataset of comments from Yahoo News. Leveraging web scraping techniques, we extract comments from a diverse range of articles across various topics. This dataset serves as the foundation for training and testing our classification model.

Preprocessing: The collected comments undergo preprocessing to ensure they are in a suitable format for analysis. This preprocessing includes tasks such as removing HTML tags, handling irrelavant characters, lowercasing and store the comments the Excel. The goal of preprocessing is to enhance the consistency and quality of the data, making it suitable for further analysis. We are using 10 sets of datasets for each category to train the model. This ensures that the model captures the specific characteristics and patterns inherent in each category.

Training the LLM models: The categorized comments are provided as prompts to the LLM during the training phase. The model learns to generate responses based on the patterns and information present in the training data. Currently Llama2, Gemini, LSTM, GPT-3.5 models are being trained using the preprocessed dataset and the best model between them is chosen.

LLAMA2 Implementation:

In this study, we employed two distinct datasets: a training dataset and a testing dataset. The training dataset comprised 20 comments for each of the following six categories: Humor, Consolidating, Abusive, Ideological, Spam, and Neutral. The testing dataset, on the other hand, consisted of 5 unique comments per category, which were distinct from those in the training dataset.

We leveraged the Replicate meta/llama-2-70b-chat model, which is a variant of the llama2 API, for model implementation and adopted the few-shot prompt method for training. The meta/llama-2-70b-chat model is a large language model trained with 70 billion parameters, and is designed for open-ended dialogue and task completion.

Specifically, all the comments from the training dataset were provided as input to the API, accompanied by a system prompt instructing the model to train based on the given examples.[12] This few-shot training process was repeated three times to enhance model fitting.

Once the model was trained on the training dataset comments, we evaluated its performance using the testing dataset. Each comment from the testing dataset was passed to the API with the following prompt: "Predict the category of the comment among the 6 categories. Just output the category." The model then generated predictions for the categories of these comments.

Evaluation:

To assess the model's performance, we compared the predicted categories with the actual categories of the comments in the testing dataset. Based on this comparison, we calculated various performance metrics, including accuracy, precision, recall, and F1-score.

Example Prompt:

For illustrative purposes, here is an example of a few-shot prompt used during the training phase:

Spam: "Why is Yahoo allowing this spammy garbage on its platform? Do better, Yahoo!" Abusive: "What a garbage article. Yahoo, you should be ashamed of yourselves for publishing this trash."

This prompt provides examples of comments from the "Spam" and "Abusive" categories, allowing the model to learn the characteristics of these categories.

Testing and Evaluation: Once trained, the LLM's Model performance is evaluated using the reserved set of 10 comments from each category. The generated responses are compared with human-labeled responses or ground truth data to assess accuracy and quality. Metrics such as accuracy, precision, recall, and F1-Score are utilized for evaluation.

Gemini Implementation:

The experimental process used to train and assess the Gemini model was quite similar to the one used to train the LLAMA2 model. We used the Gemini-Pro model for our investigation, utilizing the Gemini API implementation made possible by the Gradient third-party library. In order to start the training process, every remark in the specified training dataset was supplied into the Gemini API one at a time, enabling the model to calibrate and modify its parameters as needed. The trained Gemini model was then fed the comments from the testing dataset, producing projected output categories for every remark.[8]

After the prediction stage, the actual categories given to the testing dataset comments were compared with the predicted output categories. This made it easier to calculate a number of performance indicators, including as Accuracy, Precision, Recall, and F1-Score, which allowed for a thorough assessment of the Gemini model's effectiveness in classifying comments.

GPT-3.5 Implementation :

The implementation of GPT-3.5 for our research paralleled the approach adopted for the Gemini model. Leveraging the advanced natural language processing capabilities of GPT-3.5, we employed it as a key component in our research methodology. GPT-3.5, developed by OpenAI, represents the pinnacle of large-scale language models, boasting 175 billion parameters and unparalleled fluency and coherence in text generation tasks. Similar to the Gemini model, the GPT-3.5 API was utilized for seamless integration into our classification framework.

In the training phase, the entirety of the training dataset comments was passed through the GPT-3.5 API for fine-tuning[9] and adaptation to our specific task of comment classification. Subsequently, the testing dataset comments were inputted into the trained GPT-3.5 model, generating predicted output categories for each comment. Post-prediction, the predicted categories were compared against the actual categories assigned to the testing dataset comments. Utilizing standard evaluation metrics such as accuracy, precision, recall, and F1-score, we quantified the performance of the GPT-3.5 model in accurately categorizing comments across the predefined categories, thereby assessing its efficacy and generalization capabilities in the context of comment classification tasks.

LSTM Implementation :

For the classification of Yahoo comments into six distinct categories, we implemented a Long Short-Term Memory (LSTM) neural network model. LSTMs are a type of recurrent neural network (RNN) architecture specifically designed to capture long-term dependencies and patterns within sequential data, making them well-suited for tasks involving natural language processing.

Our LSTM model architecture consisted of multiple layers, including an embedding layer, one or more LSTM layers, and a final output layer with a softmax activation function to generate probability distributions over the six categories.

In the preprocessing stage, the Yahoo comments were tokenized and converted into numerical sequences, with each word represented as a unique index. These sequences were then padded or truncated to ensure uniform length across all comments, a necessary step for input into the LSTM model.

During training, the model learned to map the input sequences of words to the corresponding category labels through an iterative process of forward and backward propagation, guided by a chosen optimization algorithm such as Adam or RMSprop, and a predefined loss function, typically categorical cross-entropy for multi-class classification tasks. A single dataset of 20 comments is passed to this model, it is split into 80 percent training data and 20 percent testing data.

To prevent overfitting, we employed techniques such as dropout regularization and early stopping. Dropout randomly disables a fraction of neurons during training, while early stopping halts training when the model's performance on a separate validation dataset begins to degrade, thus preventing the model from memorizing noise in the training data.

After training, the performance of the LSTM model was evaluated using a separate test dataset. Metrics such as accuracy, precision, recall, and F1-score were computed to assess the model's effectiveness in accurately classifying Yahoo comments into the predefined categories.

5 DATASET

The dataset utilized in this research comprises Yahoo News headlines along with their corresponding comments, structured in an Excel spreadsheet format. The headlines, serving as the primary focus of analysis, are located in the first column, while the respective comments are situated in the second column. Obtained through web scraping techniques leveraging the Beautiful Soup library in Python, the dataset encapsulates a diverse array of topics, reflecting the breadth of coverage provided by Yahoo News.[5]

After obtaining the dataset, the dataset comments are classified into 5 different categories namely Ideological, Humorous, Consolidating, Abusive, Spam and Neutral.

The training dataset includes 180 comments in total, 60 comments for each of the three headlines. Within these sixty remarks, which fall into the six previously specified categories. The language models are trained with a small number of shot prompts using this dataset.

There are 137 comments in all across several categories in the testing dataset. The language model receives these comments and predicts the category output. After all of these comments have been received, the parameters for accuracy, precision, recall, F1 score, and other factors are assessed for each model, and the best model will be selected.

Ideological: The comments which provides a summary of a discussion, outlines its content, provides an in-depth evaluation, and reconciles various ideas or perspectives into a cohesive whole. For example, religious comments or political comments.

Humorous: Humorous comments, including jokes, puns, and witty remarks, add entertainment value to discussions and engage participants by utilizing humor as a means of engagement.

Consolidating: These comments outlines the main points of a discussion, including the contributions, summaries, and responses that aim to provide a comprehensive overview of the subject matter. For example, sad comments, comments expressing emotional sentiments.

Abusive: Comments that are insulting, personal, or use language that is disparaging to certain people or groups. Anything that encourages harassment, discrimination, or hate speech on the basis of sexual orientation, gender, race, religion, or other categories. Threats or acts of intimidation directed at other people.

Spam: Comments that are irrelevant or contain unsolicited marketing or adverts.Repeated or duplicate posts made in excess across the discussion thread. Links to unrelated or self-promotional content on other websites.

Neutral: Remarks that don't adopt a firm position or voice an opinion on the subject.

remarks or observations that convey objective knowledge without of prejudice or sentiment.

answers that don't state whether they agree or disagree but instead ask questions or demand explanation.

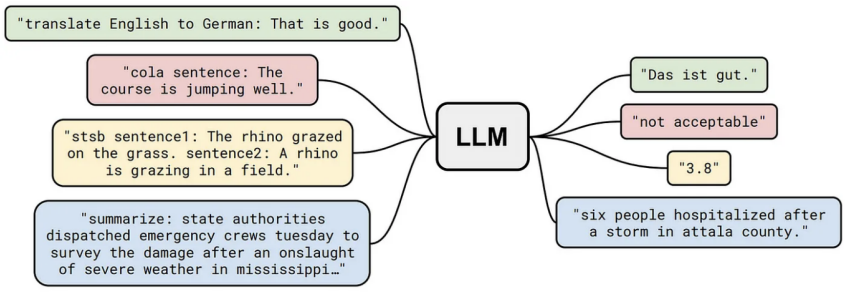

Fig. 1. Prompt Architechture

6 ARCHITECTURE

The data flow for the archtiechture starts with

Data collection: The initial step involves scraping data from Yahoo comment sections using scraping techniques. This process entails extracting comments from the website, capturing relevant information such as the comment text.

Data preprocessing: Once the data is collected, it undergoes preprocessing to prepare it for modeling. This stage typically involves cleaning the data by removing any irrelevant or redundant information, handling missing values if present, and performing text normalization or tokenization. Text normalization may involve tasks like converting text to lowercase, removing punctuation, and applying stemming or lemmatization techniques to reduce word variations.

Data splitting: The preprocessed data is then split into two or more subsets to facilitate model training and evaluation. Common splits include dividing the data into training and testing sets or using techniques like cross-validation to create multiple train-test splits. The training set is used to train the model, while the testing set is held back to evaluate the model's performance.

Model training: The training data is fed into the chosen model for training. The model could be any model out of Google Gemini, LLAMA2, GPT3 and LSTM models. During training, the model learns patterns and relationships in the data to make predictions or classifications.

Model evaluation: Once the model is trained, it is evaluated using the testing set or cross-validation folds. Evaluation metrics, including accuracy, precision, recall, F1 score, and others, are calculated to assess the model's performance. These metrics provide insights into how well the model generalizes to unseen data and whether it achieves the desired level of accuracy.

Model selection: Based on the evaluation metrics, the model that performs the best is chosen. It could be the model with the highest accuracy or a combination of metrics.

Model deployment: After selecting the best-performing model, it can be deployed for real-world use.

Comparison Baselines:

Five state-of-the-art models that are well-known for their effectiveness in natural language processing tasks Google Gemini, LLAMA2, GPT3 and LSTM models are carefully examined in the context of categorizing Yahoo comments. This exhaustive assessment comprises a thorough investigation that makes use of cutting-edge analytical techniques in addition to well-established performance criteria.

With its bidirectional context-aware architecture, which enables it to capture minute subtleties in language usage, GPT3 transformed the sector.

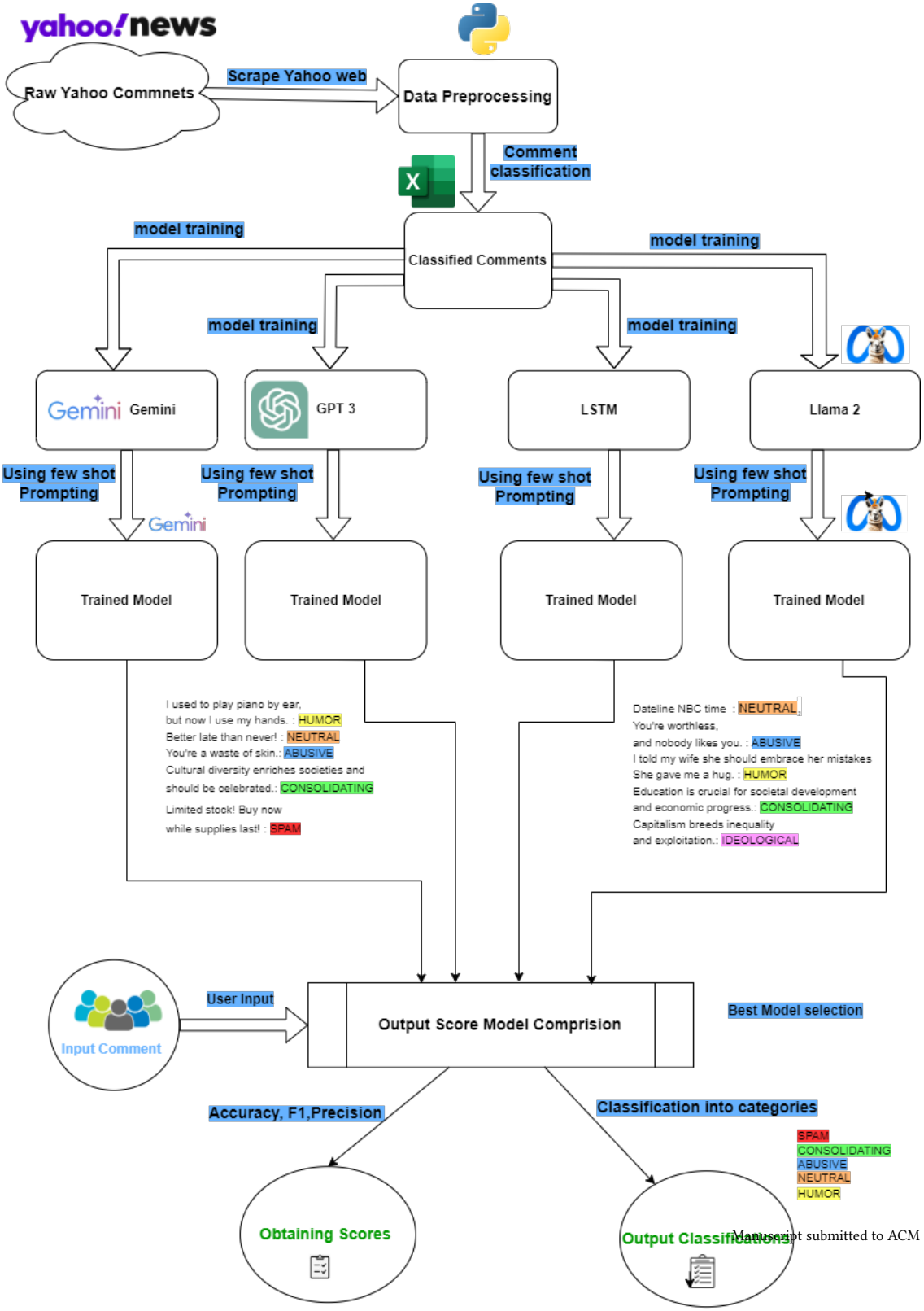

Fig. 2. Architecture Diagram

LSTMs are designed to process sequences of data, making them suitable for text where the order of words is important. They can remember and utilize context over long sequences of text, which is crucial for understanding the meaning within conversations. LSTMs are capable of learning long-term dependencies in text data, detecting patterns that span across several words or sentences. In our project, LSTM models were employed to model and categorize sequences of Yahoo comments, leveraging their ability to process sequential data effectively.

Apart from these well-known models, Google Gemini stands out as a potential competitor, using graph neural networks to extract semantic associations from textual input. By combining transformer designs and attention mechanisms, LLAMA2, an advancement of the original LLAMA model, excels in comprehending long-range dependencies and semantic coherence inside documents.

This comparison analysis offers insightful information to researchers and practitioners alike by exploring not only the technical details of each model but also evaluating their practical usefulness in real-world circumstances.

Performance Metrics Evaluation:

Precision, recall, F1 score, accuracy, and other established performance measures are important comparison points when evaluating the classification performance of Google Gemini, LLAMA2, GPT3 and LSTM on Yahoo comments. These metrics provide deep insights into how well the models are able to classify various types of comments.

* **Accuracy:** The accuracy measure is a key performance indicator that quantifies the percentage of correctly identified cases throughout the whole dataset. It provides a basic evaluation of the model's performance, showing the general accuracy of classification across different comment categories.

* **F1 Score:** The F1 score is a balanced metric that takes into account both false positives and false negatives. It is derived from the harmonic mean of precision and recall. This statistic ensures that imbalances in the distribution of classes within the dataset do not bias the model's performance and offers a nuanced assessment of the effectiveness of categorization.

* **Precision and Recall:** These two metrics provide complimentary insights into the categorization abilities of the model. Recall measures how well the model can distinguish genuine positives from all of the dataset's real positive examples. Precision, on the other hand, highlights the model's ability to reduce false positives by quantifying the percentage of real positives among all cases categorized as positive.

Through a thorough evaluation of these performance indicators across several models, researchers can advance the state-of-the-art in comment categorization by gaining a full grasp of their classification skills and identifying areas for improvement.

Choosing the best model:

To determine which language model is the most suitable for classifying Yahoo comments, this research makes use of a wide range of measurements and analysis techniques to carefully evaluate the two models. By using such a thorough approach, this research hopes to give stakeholders a detailed grasp of the advantages and disadvantages of each model, enabling them to choose the best model for a given text categorization problem.

A variety of traditional performance metrics are included in the evaluation process, such as recall, accuracy, precision, F1 score, and specificity. These metrics offer preliminary insights into the models' relative performance across several comment categories and act as fundamental benchmarks for evaluating the algorithms' effectiveness in classification.

To further explore the underlying principles and behavior of the models, the study employs advanced analytical approaches. This could entail performing exploratory data analysis to find patterns and trends in the Yahoo comment dataset, feature importance analysis to determine which features are most useful for discrimination in each model, and error analysis to learn more about the kinds of misclassifications the models experience.

Additionally, the assessment system takes into account how resilient the models are to other elements including unequal class distribution, noise in the dataset, and domain-specific difficulties associated with classifying user-generated material. To evaluate each model's generalizability and practical usefulness in real-world contexts, the study looks at how it performs in various scenarios and sensitivity assessments.

Furthermore, the study might investigate interpretability strategies to clarify the models' decision-making process and give interested parties practical understanding of the reasoning behind categorization results. This could entail using visual aids like saliency or attention maps to draw attention to the parts of the input text that have the biggest impact on the model's predictions.

Ultimately, this study aims to provide a thorough and insightful comparison of the language models under examination by combining results from a variety of measures and analysis techniques. This kind of methodology not only helps choose which model to use for classifying Yahoo comments, but it also advances the text classification field by offering insightful information about the relative benefits of various modeling strategies and assessment techniques.

7 SYSTEM

Hardware Resource Details of Llama 2

For training a model like Llama 2, the key hardware components involved are:

CPU (Central Processing Unit): Responsible for general-purpose computations. Although Llama 2 utilizes GPU acceleration, the CPU still plays a role in managing the overall training process and coordinating tasks between CPU and GPU.

GPU (Graphics Processing Unit): Highly recommended for training deep learning models like Llama 2. GPUs are specialized for handling matrix and vector operations, which are common in deep learning. They can significantly speed up the training process compared to using only CPUs.

RAM (Random Access Memory): Important for storing the training data, intermediate computations, and the model itself during training. Sufficient RAM is necessary to handle the large matrices and tensors involved in deep learning computations efficiently.

Storage (HDD or SSD): Used for storing the dataset, model checkpoints, and any intermediate outputs generated during training. An SSD is preferable over an HDD due to its faster read/write speeds, which can reduce the overall training time.

Hyper-parameter Details of Llama 2

Hyperparameters are the configuration settings used to structure the machine learning model. In Llama 2, the main hyperparameters include:

Number of layers: Typically, Llama 2 may consist of multiple layers of transformer blocks. The specific number of layers can vary depending on the model architecture and the complexity of the task.

Number of attention heads: Each layer in Llama 2 may contain multiple attention heads, which allow the model to focus on different parts of the input sequence simultaneously.

Sequence length: The maximum length of input sequences processed by the model. Longer sequences may require more memory and computational resources.

Learning rate: The rate at which the model's parameters are updated during training. A suitable learning rate is crucial for effective optimization and convergence of the model.

Dropout rate: Dropout is a regularization technique used to prevent overfitting by randomly disabling a fraction of units during training. The dropout rate determines the probability of units being dropped out. Batch size: The number

of samples processed in each iteration during training. Larger batch sizes may lead to faster convergence but require more memory.

Optimizer: The optimization algorithm used to update the model's parameters based on the computed gradients. Common optimizers include Adam, SGD, and RMSprop.

Loss function: The objective function used to quantify the difference between the model's predictions and the actual targets. Common loss functions for classification tasks include categorical cross-entropy and binary cross-entropy.

Metrics: Evaluation metrics used to assess the model's performance during training and validation. Common metrics include accuracy, precision, recall, and F1-score.

Hardware Resource Details of GPT-3.5

For training a model like GPT-3.5, the hardware resources required are typically significant due to the model's size and complexity. The key hardware components involved are:

CPU (Central Processing Unit): Responsible for managing the overall training process, preprocessing data, and coordinating tasks between CPU and GPU.

GPU (Graphics Processing Unit): Essential for training large transformer-based models like GPT-3.5. GPUs are optimized for parallel processing and can handle the massive number of parameters and computations involved in training deep learning models efficiently.

RAM (Random Access Memory): GPT-3.5 requires a substantial amount of RAM to store the model parameters, activations, and intermediate computations during training. High-capacity RAM is essential to avoid memory-related issues during training.

Storage (SSD or HDD): Used for storing the dataset, model checkpoints, and any intermediate outputs generated during training. Due to the large size of GPT-3.5, fast and ample storage, such as SSDs, is necessary to handle the data efficiently.

Hyper-parameter Details of GPT-3.5

Hyperparameters are the configuration settings used to structure the machine learning model. In GPT-3.5, the main hyperparameters include:

Number of layers: GPT-3.5 consists of a massive number of layers, typically in the range of hundreds to thousands, depending on the variant.

Number of attention heads: Each layer in GPT-3.5 contains multiple attention heads, allowing the model to capture different aspects of the input sequence simultaneously.

Sequence length: The maximum length of input sequences processed by the model. GPT-3.5 can handle long sequences, but longer sequences may require more memory and computational resources.

Learning rate: The rate at which the model's parameters are updated during training. Optimizing the learning rate is crucial for effective convergence and stability during training.

Batch size: The number of samples processed in each iteration during training. GPT-3.5 can benefit from large batch sizes to utilize the parallel processing capabilities of GPUs efficiently.

Optimizer: The optimization algorithm used to update the model's parameters based on the computed gradients. Common optimizers include Adam, SGD, and RMSprop.

Loss function: The objective function used to quantify the difference between the model's predictions and the actual targets. GPT-3.5 is typically trained using language modeling objectives such as cross-entropy loss. Evaluation metrics: Metrics used to assess the performance of the model during training and validation. Common metrics include perplexity, BLEU score, and accuracy on downstream tasks.

Hardware Resource Details of Gemini

For training a model like Gemini, the key hardware components involved are:

CPU (Central Processing Unit): Responsible for managing the overall training process, preprocessing data, and coordinating tasks between CPU and GPU.

GPU (Graphics Processing Unit): Essential for training deep learning models like Gemini. GPUs are optimized for parallel processing and can handle the computations involved in training neural networks efficiently.

RAM (Random Access Memory): Important for storing the training data, intermediate computations, and the model itself during training. Sufficient RAM is necessary to handle the large matrices and tensors involved in deep learning computations efficiently.

Storage (SSD or HDD): Used for storing the dataset, model checkpoints, and any intermediate outputs generated during training. Fast storage, such as SSDs, is preferred for faster data access and training speed.

Hyper-parameter Details of Gemini

Hyperparameters are the configuration settings used to structure the machine learning model. In Gemini, the main hyperparameters include:

Number of layers: Gemini may consist of multiple layers of transformer blocks, similar to other transformer-based models.

Number of attention heads: Each layer in Gemini contains multiple attention heads, allowing the model to capture different aspects of the input sequence simultaneously.

Sequence length: The maximum length of input sequences processed by the model. Longer sequences may require more memory and computational resources.

Learning rate: The rate at which the model's parameters are updated during training. Optimizing the learning rate is crucial for effective convergence and stability during training.

Batch size: The number of samples processed in each iteration during training. Larger batch sizes may lead to faster convergence but require more memory.

Optimizer: The optimization algorithm used to update the model's parameters based on the computed gradients. Common optimizers include Adam, SGD, and RMSprop.

Loss function: The objective function used to quantify the difference between the model's predictions and the actual targets. Common loss functions include categorical cross-entropy and binary cross-entropy for classification tasks.

Metrics: Evaluation metrics used to assess the model's performance during training and validation. Common metrics include accuracy, precision, recall, and F1-score.

Hardware Resource Details of LSTM

For training an LSTM model, particularly with TensorFlow and Keras, the key hardware components involved are:

CPU (Central Processing Unit): Responsible for general-purpose computations. In most cases, the CPU is involved in data pre-processing and other tasks that are not directly related to training the neural network.

GPU (Graphics Processing Unit): Highly recommended for training LSTM models. GPUs are specialized for handling matrix and vector operations, which are common in deep learning. They can significantly speed up the training process.

RAM (Random Access Memory): Important for storing the training data and the model during training. Insufficient RAM can lead to slower training or the inability to train large models.

Storage (HDD or SSD): Used for storing the dataset, model, and any intermediate outputs. An SSD is faster than an HDD and can speed up the loading of data.

Hyper-parameter Details of LSTM:

Hyperparameters are the configuration settings used to structure the machine learning model. In an LSTM model, the main hyperparameters include:

Embedding Layer:

input dim: The size of the vocabulary (number of unique words in your dataset). output dim: The dimension of the dense embedding, set to 128. This represents the size of the feature vectors for each word. input length: The length of input sequences, set to 100.

LSTM Layers:

First LSTM layer with 64 units (return sequences=True to return the full sequence to the next layer). Second LSTM layer with 32 units.

Dropout Layers: Set at 0.5, used to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training.

Dense Layer: The output layer with 6 units, one for each target column, using the 'sigmoid' activation function (commonly used for binary classification).

Model Compilation:

optimizer: 'adam', an efficient stochastic optimization method. loss: 'binary crossentropy', commonly used for binary classification tasks. metrics: 'accuracy', to evaluate the performance of the model.

> **Training:** batch size: Set to 128, determining the number of samples processed before the model is updated. epochs: Set to 2, the number of complete passes through the training dataset. validation split: Set to 0.1, indicating that 10

Early Stopping Callback: Monitors the validation loss and will stop training if the loss doesn't improve after three epochs (patience=3).

8 DISCUSSION

In exploring methodologies for automated content moderation, various models and techniques have been proposed and evaluated in the literature. This section discusses the key findings and comparisons with existing approaches.

The mSVM model, as discussed in MacAvaney et al. (2019)[6], offers an interpretable solution for automatic hate speech detection. Its competitive performance on specific datasets, such as Stormfront and TRAC, highlights its efficacy in certain contexts. However, the model's performance variation across different datasets suggests the need for customization to accommodate diverse types of content. In contrast, the proposed LSTM approach prioritizes capturing long-term dependencies and contextual nuances, potentially offering a more generalizable solution for toxic comment classification across various platforms. Additionally, the integration of word embeddings in the SVM approach represents an advancement over traditional mSVM, enhancing classification accuracy by incorporating semantic information.

Del Vigna et al. [4] introduced a hate speech classifier for Italian texts utilizing SVM and LSTM models. While leveraging morpho-syntactical features and sentiment polarity, their approach differs from the proposed system, which employs a 'PatternTokenizer' and an 'Embedding' layer to understand the semantics of the text. Moreover, the proposed system aims for broader application by employing preprocessing techniques and embeddings that are language-agnostic, potentially increasing its versatility for different languages and types of online content.

Deep learning approaches, as highlighted by Maslej et al[7]., emphasize the use of neural network architectures like RNNs and CNNs for toxic comment classification. These studies explore transfer learning and fine-grained classification schemes to improve accuracy.

9 RESULTS

The results obtained from the different models reveal distinct levels of performance concerning accuracy, precision, recall, and F1 score. The Gemiini model exhibited an accuracy rate of approximately 33.33

In contrast, the Llama 2 model displayed more favorable performance with an accuracy of roughly 69.17

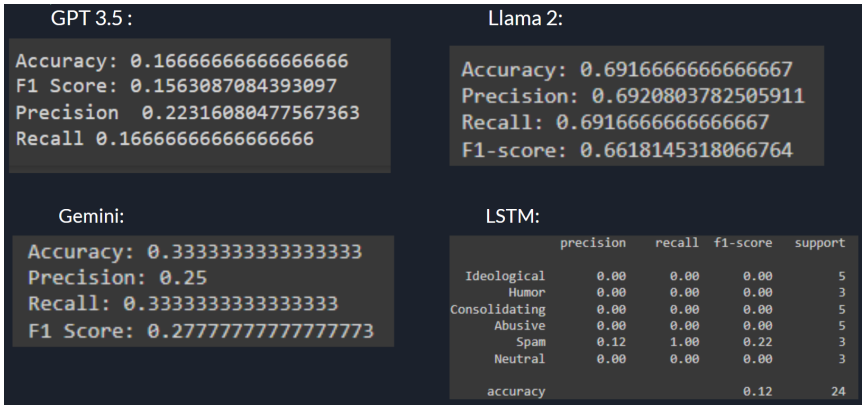

Fig. 3. Obtained parameter values for different models

Conversely, the LSTM model exhibited notably poor performance, as evidenced by its low accuracy of only 12

Lastly, the GPT 3.5 model achieved an accuracy rate of approximately 16.67

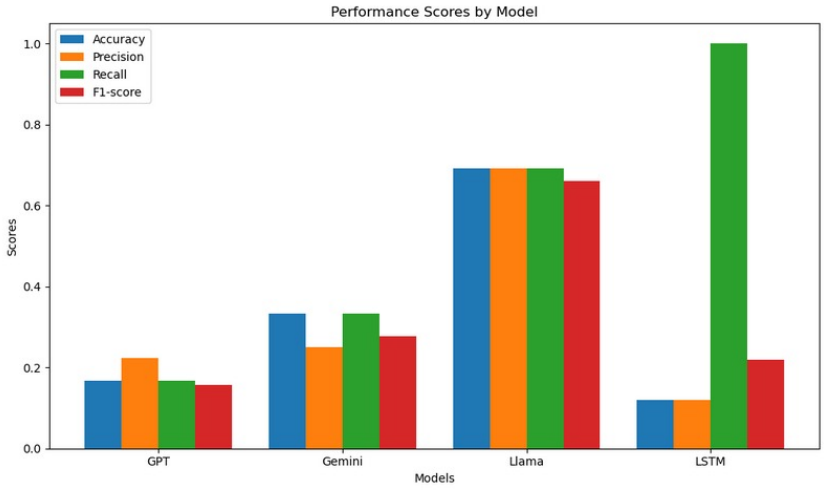

Fig. 4. Bar graph comparison for multiple models

In conclusion, the models varied significantly in terms of accuracy, precision, recall, and F1 score, thereby highlighting the importance of selecting an appropriate model for accurate and effective natural language processing tasks. Among the models evaluated, the Llama 2 model demonstrated the most favorable performance, whereas the LSTM model

exhibited the poorest results. These findings underscore the significance of choosing a suitable model to achieve accurate and effective classification in the realm of natural language processing.

Error Analysis

Below are the predicted reasons on why the models are having low accuracies and incorrect outputs.

Insufficient training data: The models may not have been trained on a sufficiently large or diverse dataset. Having a limited range of examples to learn from can hinder their ability to generalize and accurately classify new instances.

Complex or ambiguous language: Natural language processing tasks often involve nuances, idiomatic expressions, and context-dependent meanings. If the models were not adequately trained to understand and handle such complexities, their performance could be impacted.

Inadequate model complexity: The models used may not have had sufficient complexity or capacity to capture the intricacies of the classification task. A more sophisticated model architecture or additional layers of complexity may be required to improve accuracy.

Lack of fine-tuning or hyperparameter optimization: The models may not have been fine-tuned or optimized using appropriate techniques. Fine-tuning and hyperparameter optimization can help improve the models' performance by finding the best configuration for the specific task.

Inherent challenges in the classification task: Some classification tasks, especially in natural language processing, can be inherently challenging. The presence of ambiguous instances, subjective categorizations, or overlapping classes can make it difficult to achieve high accuracy.

Models like GPT 3.5 and Gemini cannot be finetuned liike Llama model, In order to obtain more accuracy for GPT 3.5 and Gemini finetuning is neccesary but practically it is expensive and challenging.

Overall, error analysis allows us to gain a deeper understanding of why certain instances were misclassified and provides valuable insights for refining the models and enhancing their accuracy and effectiveness in future iterations.

Additionally, we envision building a Chrome extension based on our implementation, allowing users to seamlessly integrate our comment categorization tool into their browsing experience. This extension will provide users with real-time insights into the categorization of comments, empowering them to engage more effectively and responsibly in online discussions.

10 CONCLUSION

Comparing the scores of all the Models:

Accuracy: Llama has the highest accuracy (0.69), indicating it has the highest proportion of correct predictions overall.

Precision: Llama and Gemini have the highest precision (0.69 and 0.25 respectively), meaning they make fewer false positive predictions compared to the other models.

Recall: LSTM has the highest recall (1.0), meaning it retrieves all relevant instances in the dataset. However, its low precision suggests it also retrieves many irrelevant instances.

F1-score: Llama has the highest F1-score (0.66), which balances precision and recall effectively.

Considering all metrics, Llama appears to be the best-performing model overall. It achieves a good balance between precision and recall, with high accuracy. LSTM, while having perfect recall, suffers from low precision and overall performance. GPT performs relatively poorly compared to Gemini and Llama across all metrics. Gemini, though having decent performance, falls short compared to Llama in terms of precision and F1-score.

On examining why Accuracy scores are not high for each model:

Metric	GPT 3.5	Llama 2	LSTM	Gemini
Accuracy	0.20	**0.69**	0.12	0.33
Precision	0.18	**0.69**	0.12	0.25
Recall	0.20	**0.69**	1.00	0.33
F1 Score	0.18	**0.66**	0.22	0.27

Table 1. Comparison Table

GPT:

GPT might not have been specifically fine-tuned for the task at hand. If the task requires nuanced understanding or domain-specific knowledge, GPT's general language model capabilities may not be sufficient.

GPT might not have been trained on a dataset that adequately represents the task's complexity or diversity. Insufficient training data can lead to suboptimal performance, especially in tasks where patterns are subtle or context-dependent.

Gemini: Similar Reasons as GPT: Many of the reasons for GPT's lower accuracy apply here as well, especially if Gemini shares similar architecture or training data.

If Gemini's architecture is complex or overfitting to the training data, it might struggle with generalization to unseen data, leading to lower accuracy.

If the training data for Gemini is noisy or contains errors, it can negatively impact its ability to learn and generalize, resulting in lower accuracy.

Llama: Llama might be trained on a dataset where one class is much more prevalent than the other(s), leading to a bias towards predicting the majority class. This can lower overall accuracy, especially if the minority class is of interest.

The evaluation metric used for determining accuracy might not be well-suited for the task. For instance, in highly imbalanced datasets, accuracy might not be the most informative metric, and other metrics like F1-score might provide a clearer picture of model performance.

LSTM: Overfitting: LSTM might have overfit to the training data, meaning it learned to memorize patterns specific to the training set rather than generalizing to new data. This can result in poor performance on unseen data and lower accuracy.

Class Imbalance: Similar to Llama, if LSTM is trained on an imbalanced dataset, it might struggle to accurately predict the minority class, affecting overall accuracy.

Model Complexity: If LSTM's architecture is too complex or not appropriately tuned for the task, it might struggle with generalization, leading to lower accuracy.

For working well with huge training and testing data:

Llama and GPT: Both Llama and GPT have shown potential for handling large volumes of data during training and testing. GPT, being a transformer-based model, can efficiently process large amounts of text data. Llama, with its high accuracy and balanced performance metrics, could handle large datasets effectively as well.

REFERENCES

[1] G Akash, Himanshu Kumar, and D Bharathi. Toxic comment classification using transformers. In *Proceedings of the 11 th Annual International Conference on Industrial Engineering and Operations Management Singapore*, pages 1895–1905, 2021.

[2] Ahlam Alrehili. Automatic hate speech detection on social media: A brief survey. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6. IEEE, 2019.

[3] Ona De Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. Hate speech dataset from a white supremacy forum. *arXiv preprint arXiv:1809.04444*, 2018.

[4] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the first Italian conference on cybersecurity (ITASEC17)*, pages 86–95, 2017.

[5] Anusha Garlapati, Neeraj Malisetty, and Gayathri Narayanan. Classification of toxicity in comments using nlp and lstm. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 16–21. IEEE, 2022.

[6] Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152, 2019.

[7] Viera Maslej-Krešňáková, Martin Sarnovskỳ, Peter Butka, and Kristína Machová. Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, 10(23):8631, 2020.

[8] Ankit Pal and Malaikannan Sankarasubbu. Gemini goes to med school: Exploring the capabilities of multimodal large language models on medical challenge problems & hallucinations. *arXiv preprint arXiv:2402.07023*, 2024.

[9] Chanathip Pornprasit and Chakkrit Tantithamthavorn. Gpt-3.5 for code review automation: How do few-shot learning, prompt design, and model fine-tuning impact their performance? *arXiv preprint arXiv:2402.00905*, 2024.

[10] Julian Risch and Ralf Krestel. Toxic comment detection in online discussions. *Deep learning-based approaches for sentiment analysis*, pages 85–109, 2020.

[11] Anna Wolters, Kilian Müller, and Dennis M Riehle. Incremental machine learning for text classification in comment moderation systems. In *Multidisciplinary International Symposium on Disinformation in Open Online Media*, pages 138–153. Springer, 2022.

[12] Xuan Zhang, Navid Rajabi, Kevin Duh, and Philipp Koehn. Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with qlora. In *Proceedings of the Eighth Conference on Machine Translation*, pages 468–481, 2023.