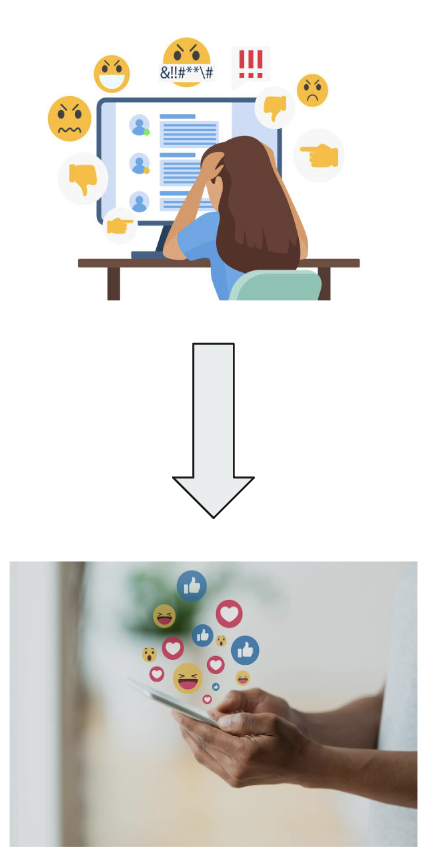
**CS6120: Natural Language Processing - Prof. Prashant Mittal**

**Identifying and Classifying Toxic Online Behavior**

**Group F: Ruijia Xiong, Gaurav, Jheel Kamdar, Xiaowen Sun**

**Introduction**

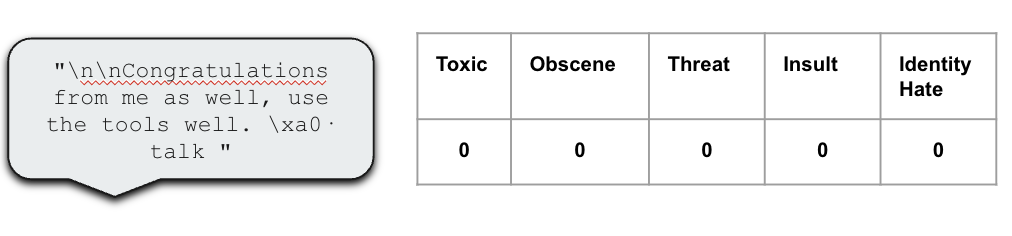
* **Problem Statement:**
  + Negative online behavior encompasses **personal insults, hate speech, cyberbullying, and harassment**, which can significantly impact users' mental health and participation in online communities.
  + The urgency to address these issues is amplified by their increasing prevalence across various digital platforms.
* **Objective:**
  + The project's objective is to develop a **sophisticated model capable of accurately predicting and classifying toxic comments**.
  + Such a tool will **empower moderators and platform developers to intervene more effectively**, thereby improving user interactions and community health.
* **Impact on Users:**
  + By reducing exposure to toxic content, the proposed solution aims to enhance user engagement and promote a **safer, more inclusive online environment**, which is crucial for mental well-being and sustained participation.

****

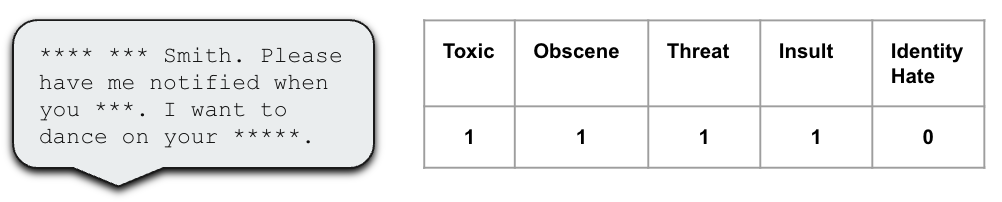
**Dataset Overview**

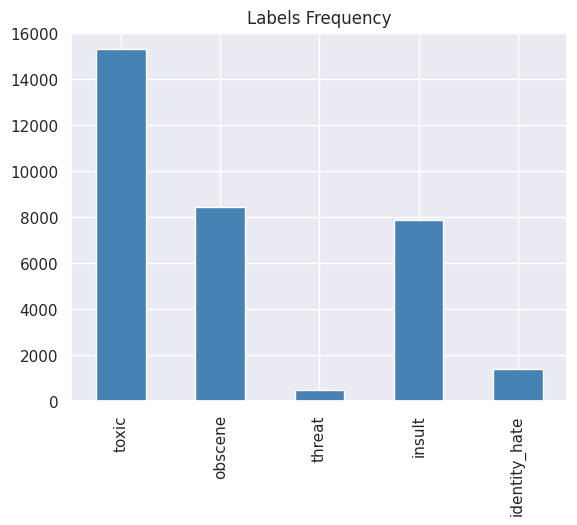
* **Source:**
  + The dataset, provided by **Jigsaw as part of a Kaggle competition**, comprises comments from Wikipedia talk pages, annotated for various forms of toxicity such as severe **toxicity, threats, obscenity, insults, and identity hate speech**.
* **Statistics:**
  + The dataset includes **160,000 comments**, with a nuanced labeling system that marks comments based on the type and severity of toxicity.
  + This comprehensive tagging aids in fine-grained analysis and modeling.

Example of Positive Comment:



Example of Negative Comment:





Analysis reveals that among all the labels, 'toxic' is the most frequently observed in the training dataset, whereas 'threat' appears the least often.

* **Data Preprocessing:**
  + *Text Cleaning*

Normalization: The primary goal of normalization in text processing is to **reduce variability** in the data.

Here, all text is converted to **recognize patterns indicative of harmful online behavior to ensure uniformity**, aiding in consistent analysis across different text inputs.

Special characters such as **commas, periods, question marks, and exclamation points are retained** due to their significance in understanding the text structure and sentiment, while other non-essential characters are removed.

Additionally, **URLs and HTML tags**, which could disrupt text analysis, are systematically **stripped** from the text to maintain a focus on meaningful content.

* + *Tokenization*

This stage involves **breaking down the text into smaller parts**, known as tokens, typically words or phrases.

Using the Natural Language Toolkit (NLTK), the process **includes removing stopwords**—words that are extremely common and carry little meaningful variance for analysis (e.g., "the", "is", "at").

This helps to **concentrate on the words that contribute more significantly** to the meaning of the text.

* + *Embeddings*

FastText (vector size = 300): FastText is particularly adept at **understanding the morphology of words by capturing subword information**. This feature is extremely beneficial for processing languages where words are often formed by combining simpler morphemes, including prefixes and suffixes. A **vector size of 300** dimensions is chosen to adequately capture these complex relationships without excessively increasing computational requirements.

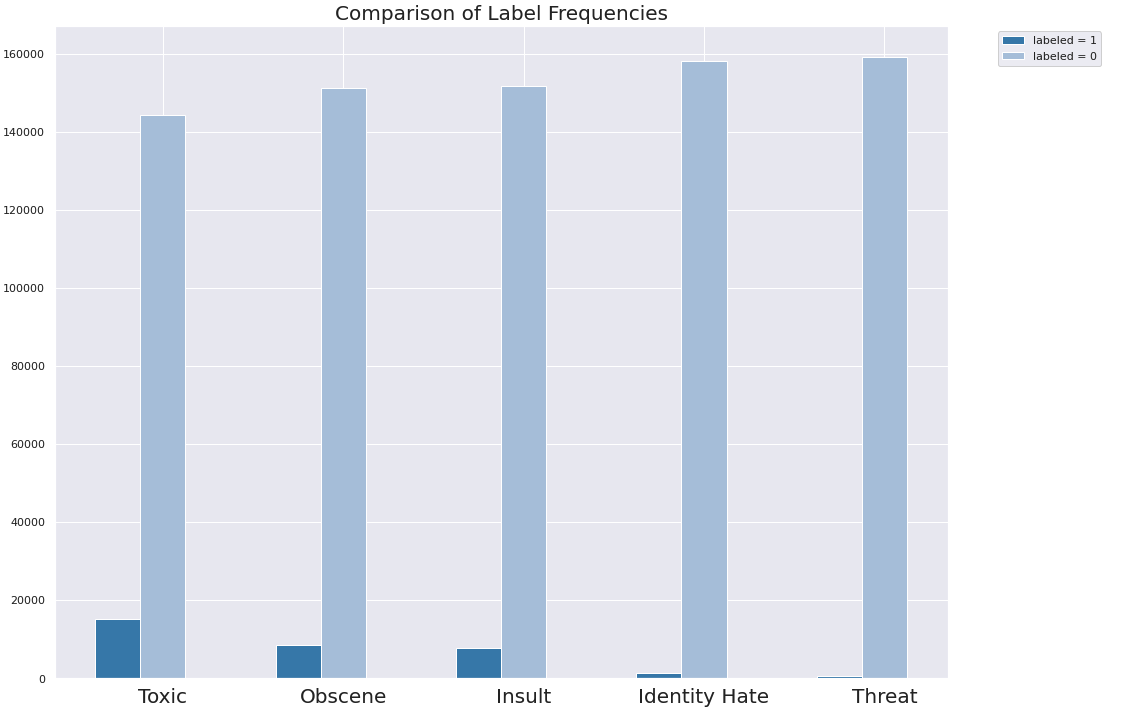
Word2Vec (vector size = 100): Word2Vec, on the other hand, **excels in capturing the semantic relationships between words**, making it ideal for contexts where nuanced understanding and inferences about text are crucial. A smaller **vector size of 100** is used to optimize performance and efficiency, balancing detail with computational speed.

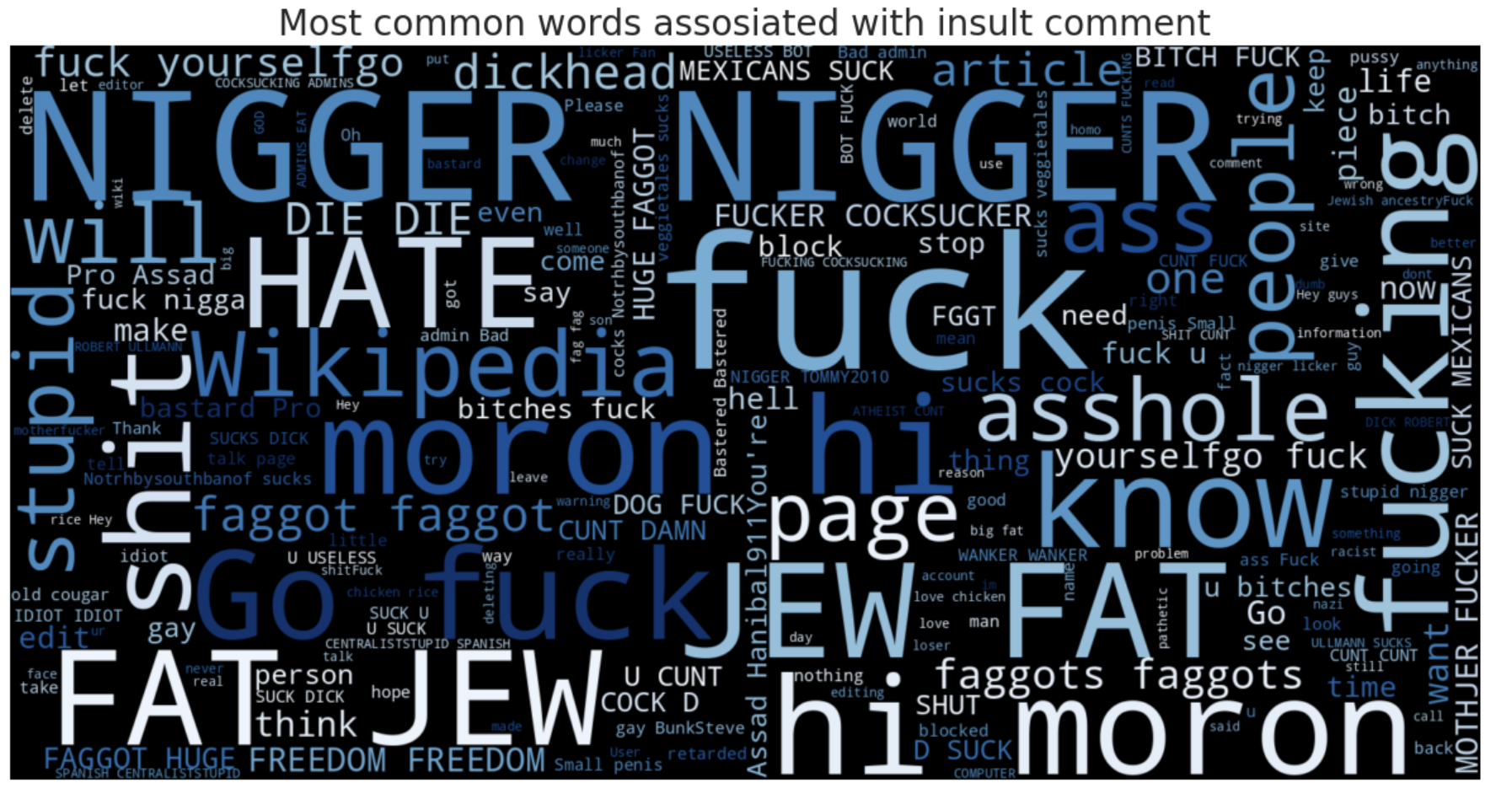
**FastText was selected** for real-time processing in the **UI** due to its robustness in handling various word forms, which enhances user experience by accommodating diverse inputs. However, **for training the Convolutional Neural Network** (CNN) model, **Word2Vec vectors were utilized** as they yielded better performance by effectively capturing deeper semantic meanings.

* + *Visualization*

Visualization tools such as **word clouds and frequency histograms** are employed to illustrate common words and phrases within each class of the dataset (toxic, non-toxic, etc.).

These visualizations aid in identifying prevalent themes and terms in different categories, **providing intuitive insight**s that can guide further analysis and model training. By understanding the words that frequently appear in toxic versus non-toxic comments, for instance, researchers can better tailor their models to recognize patterns indicative of harmful online behavior.





**Model development**

**Overview of Models Used:**

The project tests multiple models, including Logistic Regression and Random Forest for baseline assessments, and more complex models like Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) for deeper learning and pattern recognition. Zero-Shot Classification is explored as a novel approach to leverage pre-trained models without additional training.

* 1. Logistic Regression:

Utilized for its effectiveness in binary classification tasks, providing interpretable probabilities of outcomes based on predictor variables.

Architecture: 3 fully connected layers. The first two layers have a dropout rate of 0.5.

* 2. Random Forest with OneVsRestClassifier:

Beneficial for tackling multi-class classification challenges by breaking down the problem into several binary classification subtasks, enabling efficient classification of instances into multiple classes.

* 3. CNN (Convolutional Neural Network):

Architecture: 2 convolutional layers followed by a fully connected classifier.

* 4. LSTM (Long Short-Term Memory):

Architecture: 1 LSTM layer followed by a fully connected classifier.

Note: LSTM is not performing as expected, possibly due to not using sequential input.

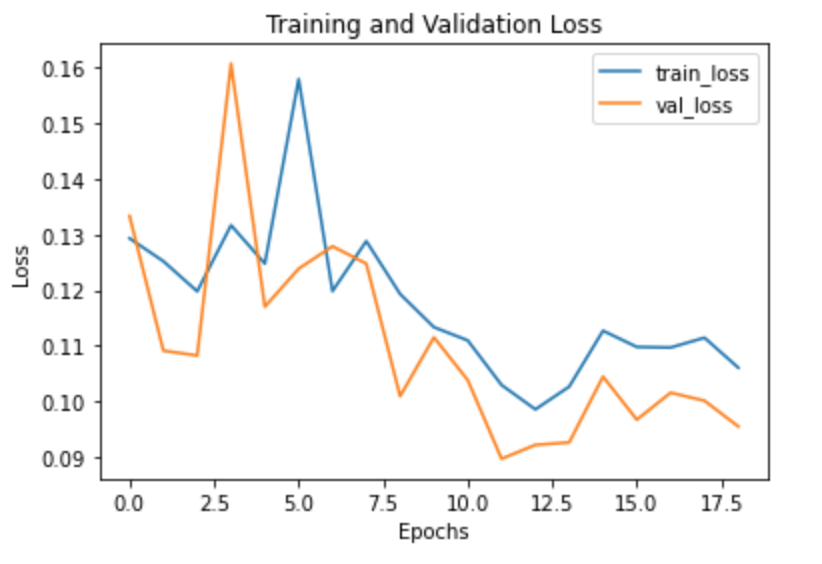
**Model Performance**

Each model's effectiveness is gauged using accuracy, precision, recall, and F1-score. The CNN and LSTM models show promising results in capturing the nuances of language used in toxic comments.

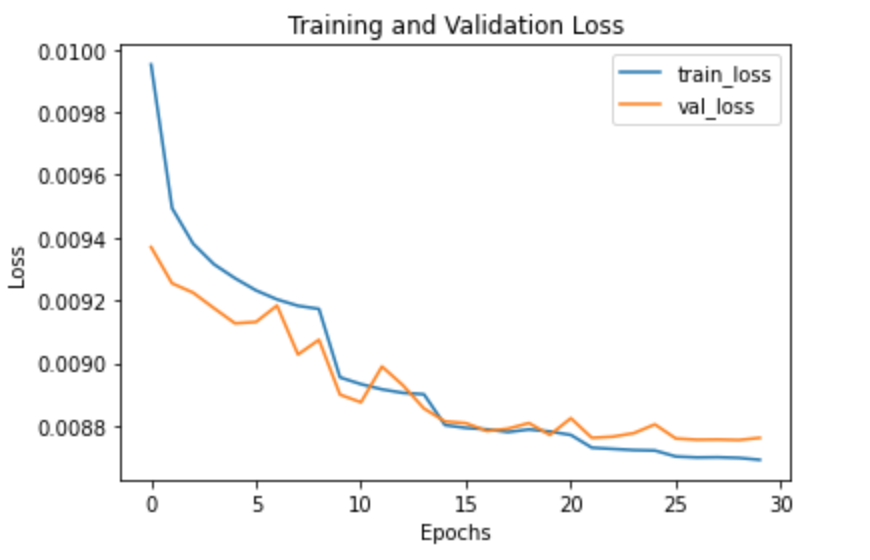
**Logistic Regression**

****

**LSTM**

****

**CNN**

****

|  | **Gross Accuracy**  **(Row-wise)** | **Toxic** | **Obscene** | **Threat** | **Insult** | **Identity Hate** |
| --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression** | 0.91 | 0.94 | 0.96 | 1.00 | 0.96 | 0.99 |
| **Random Forest** | 0.91 | 0.94 | 0.97 | 1.00 | 0.96 | 0.99 |
| **CNN** | 0.92 | 0.94 | 0.97 | 1.00 | 0.96 | 0.98 |
| **LSTM** | 0.89 | 0.91 | 0.94 | 1.00 | 0.95 | 0.99 |

Comparison:

Overall, all models demonstrate high accuracy in classifying toxic, obscene, threat, insult, and identity hate comments.

Logistic Regression, Random Forest, and CNN exhibit consistently high accuracy across all categories.

LSTM shows slightly lower accuracy compared to other models, particularly in toxic and obscene categories, possibly due to the input data not being effectively utilized in a sequential manner.

Conclusion:

For this specific task, Logistic Regression, Random Forest, and CNN appear to be more suitable models based on their performance metrics.

Further investigation is required to understand why LSTM is underperforming and whether sequential input processing could improve its performance.

Model selection should be based on the specific requirements of the application and the trade-offs between interpretability, complexity, and computational efficiency.

**Zeroshot Classification**

Zero-Shot Classification enables our model to predict labels for new types of toxic behavior without requiring explicit training examples for those categories.

**Data Preparation**

* Data Loading and Cleaning:

The process begins by loading a dataset from a CSV file into a pandas DataFrame. The data is cleaned by removing unnecessary columns such as 'severe\_toxic' and 'id', simplifying the dataset and focusing on relevant features for classification.

* Defining Candidate Labels:

Labels such as 'toxic', 'obscene', 'threat', 'insult', and 'identity hate' are defined for the zero-shot classification. These labels represent different categories of offensive content that the model will classify without explicit prior training on these specific categories.

**Data Analysis**

* Analyzing Distribution:

The distribution of message categories within the data is analyzed. This involves calculating the percentage of messages labeled under each category and identifying messages that do not fit any predefined category (labeled as 'other').

* Sample Selection:

For a detailed analysis, a sample of 200 labeled messages is selected randomly. This subset provides a manageable size for testing the classification models.

**Models Used**

* General-Purpose Models:

Roberta-large-mnli: Trained on the Multi-Genre Natural Language Inference Dataset, this model is designed for broad adaptability.

Facebook/bart-large-mnli: Another model based on the Multi-Genre NLI dataset, known for its effectiveness in complex inference tasks.

Typeform/distilbert-base-uncased-mnli: A lighter model that maintains decent performance, making it suitable for environments with limited computational resources.

* Binary Toxicity Specialized Models:

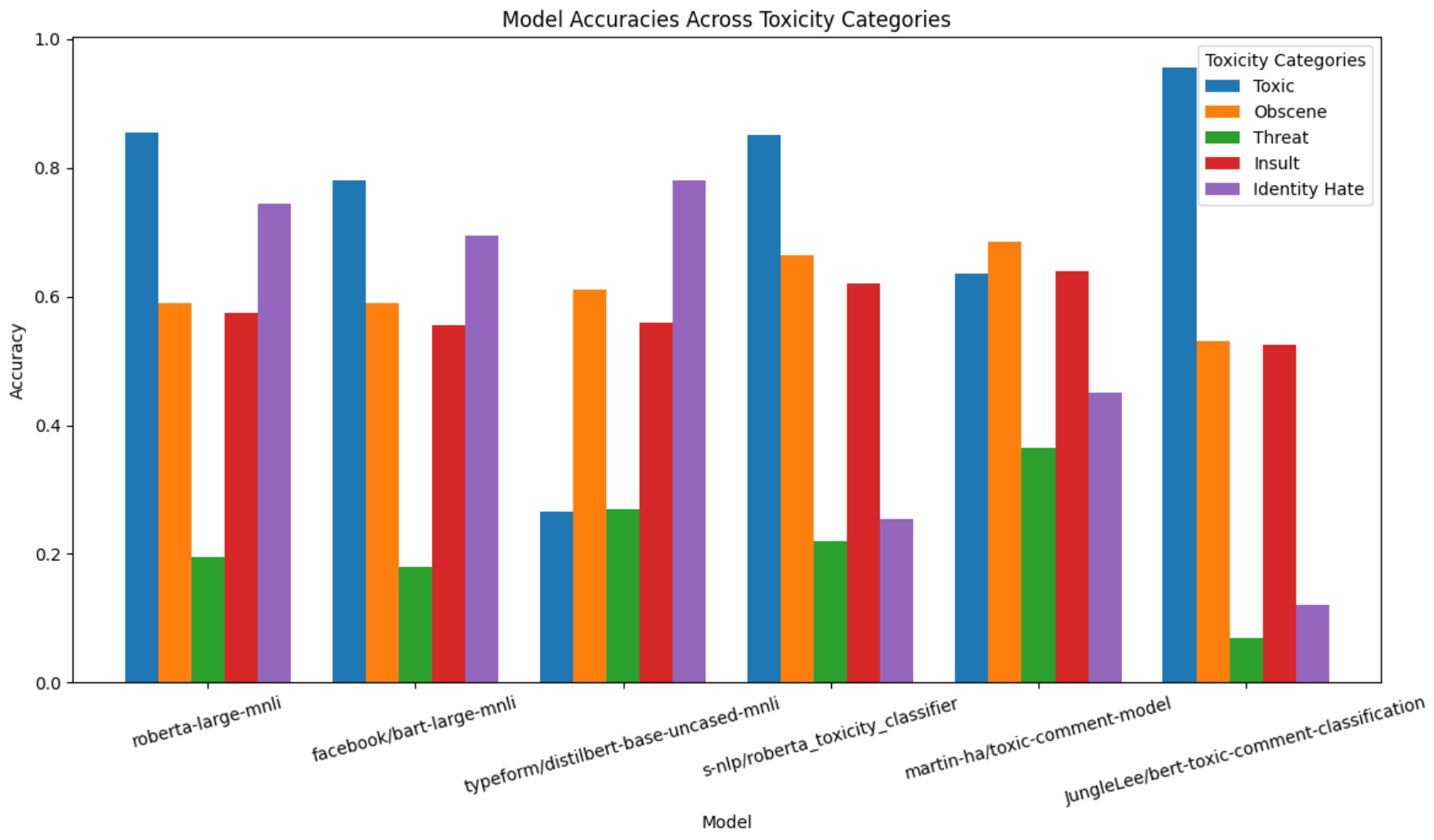
S-nlp/roberta\_toxicity\_classifier: Fine-tuned on the Jigsaw Toxic Comment Classification Dataset, this model is highly effective for specific toxicity detection.

Martin-ha/toxic-comment-model: Another model fine-tuned for toxicity but using a different dataset (Civil Comments platform).

JungleLee/bert-toxic-comment-classification: This model is also specialized in detecting toxic comments, optimized for high precision in certain categories.

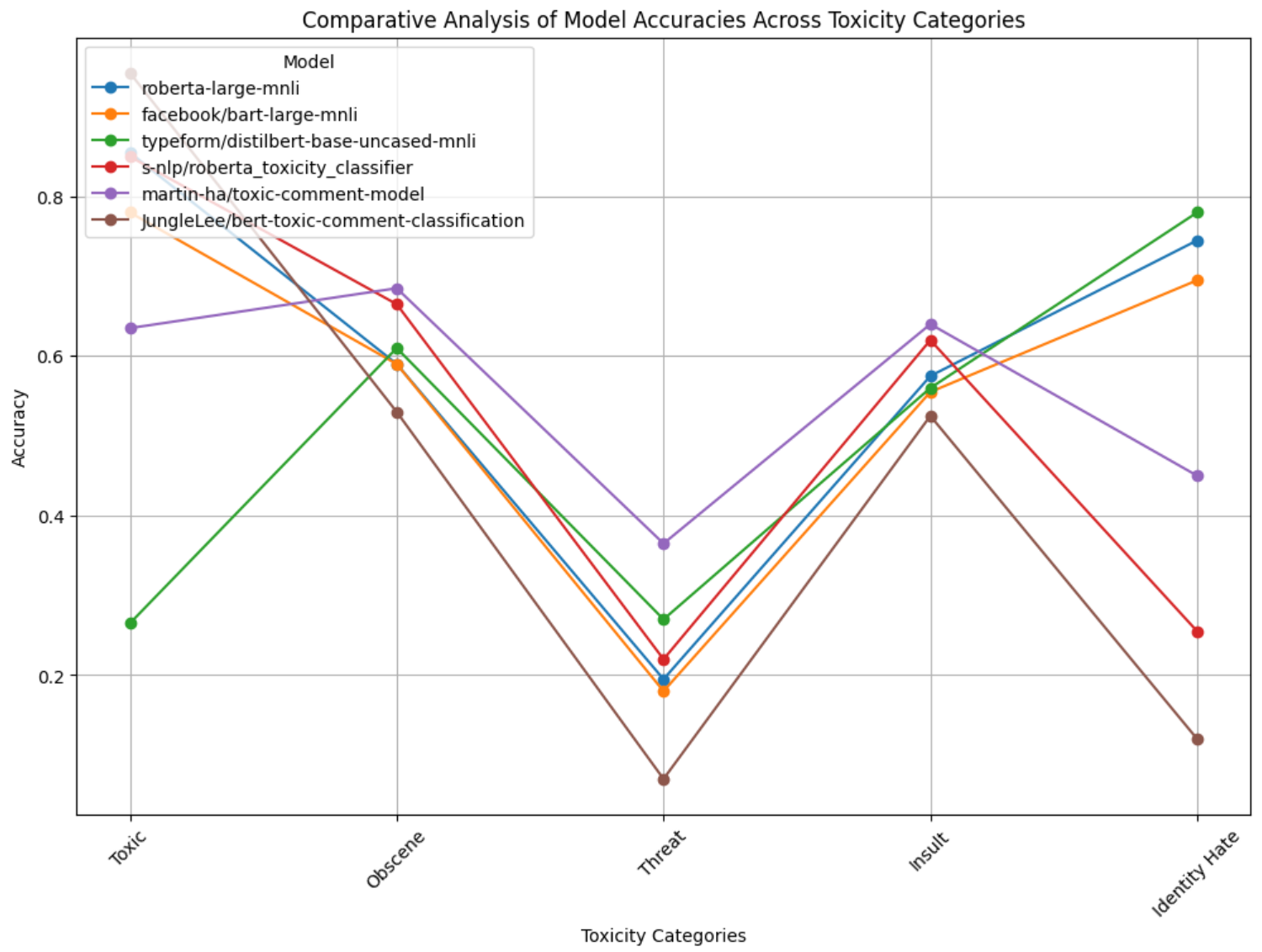
**Result**

| **Model** | **Toxic** | **Obscene** | **Threat** | **Insult** | **Identity Hate** |
| --- | --- | --- | --- | --- | --- |
| roberta-large-mnli | 0.855 | 0.590 | 0.195 | 0.575 | 0.745 |
| facebook/bart-large-mnli | 0.780 | 0.590 | 0.180 | 0.555 | 0.695 |
| typeform/distilbert-base-uncased-mnli | 0.265 | 0.610 | 0.270 | 0.560 | **0.780** |
| s-nlp/roberta\_toxicity\_classifier | 0.850 | 0.665 | 0.220 | 0.620 | 0.255 |
| martin-ha/toxic-comment-model | 0.635 | **0.685** | **0.365** | **0.640** | 0.450 |
| JungleLee/bert-toxic-comment-classification | **0.955** | 0.530 | 0.070 | 0.525 | 0.120 |



The results indicate that fine-tuned toxicity models generally outperform general-purpose models in identifying toxic behavior.

However, achieving uniformly high performance across all categories remains a challenge, particularly in nuanced areas like threats.



It is suggested that continuous model evaluation and updating of model training sets with emerging types of toxic content could enhance detection capabilities. Moreover, combining predictions from multiple models might also improve accuracy, especially in handling complex or subtle cases of toxicity.

Limitations and Lessons Learned

- Dataset Limitations: The dataset’s skewed distribution might not represent the true variety of toxic behavior across different platforms. Furthermore, potential biases in annotation can affect model training and evaluation.

- Technical Challenges: Limitations in computational resources restricted the ability to experiment with transformer models, which require substantial memory and processing power.

- Future Recommendations: Future work could include the application of synthetic data generation techniques such as translation-based augmentation to enrich the dataset, or exploring unsupervised learning methods to uncover additional patterns in unlabeled data.

Conclusion

- Summary of Findings: The findings highlight the potential of machine learning models to effectively identify different types of toxic online behavior, thereby supporting the development of automated moderation tools.

- Future Directions: Recommendations for future research include exploring more advanced deep learning architectures and incorporating real-time detection systems into live platforms.