# New York Times Articles & Comments (2020)

A data-driven analysis of reader interaction patterns across New York Times content from 2020.

**by Akash K P, Luv Aggarwal, Rachit Mehta, Vivek Rajput**

# Problem Statements

The New York Times receives millions of comments annually. Understanding and predicting reader engagement can improve content strategy and user experience.

### Comment Prediction

Predict the number of comments an article will receive

### Recommendation Count

Predict how many recommendations a comment will get

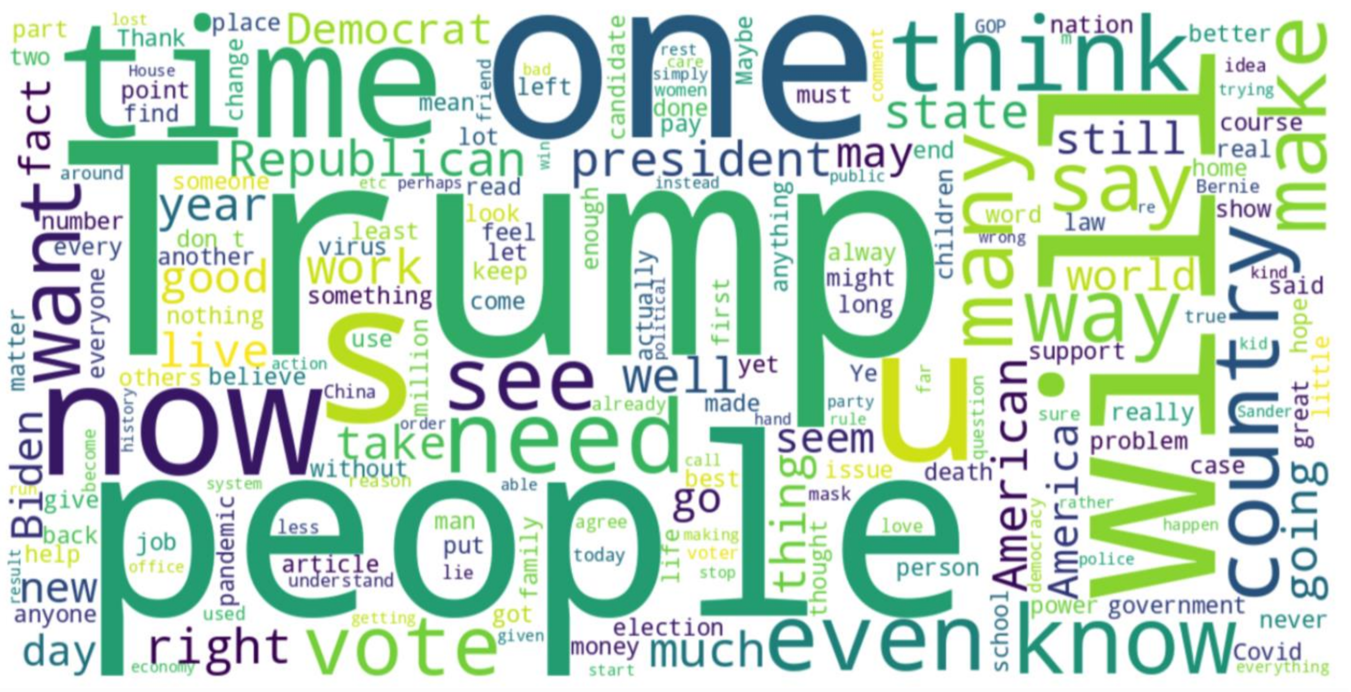
### Editor's Pick Classification

Predict which comments editors will select as Times Picks

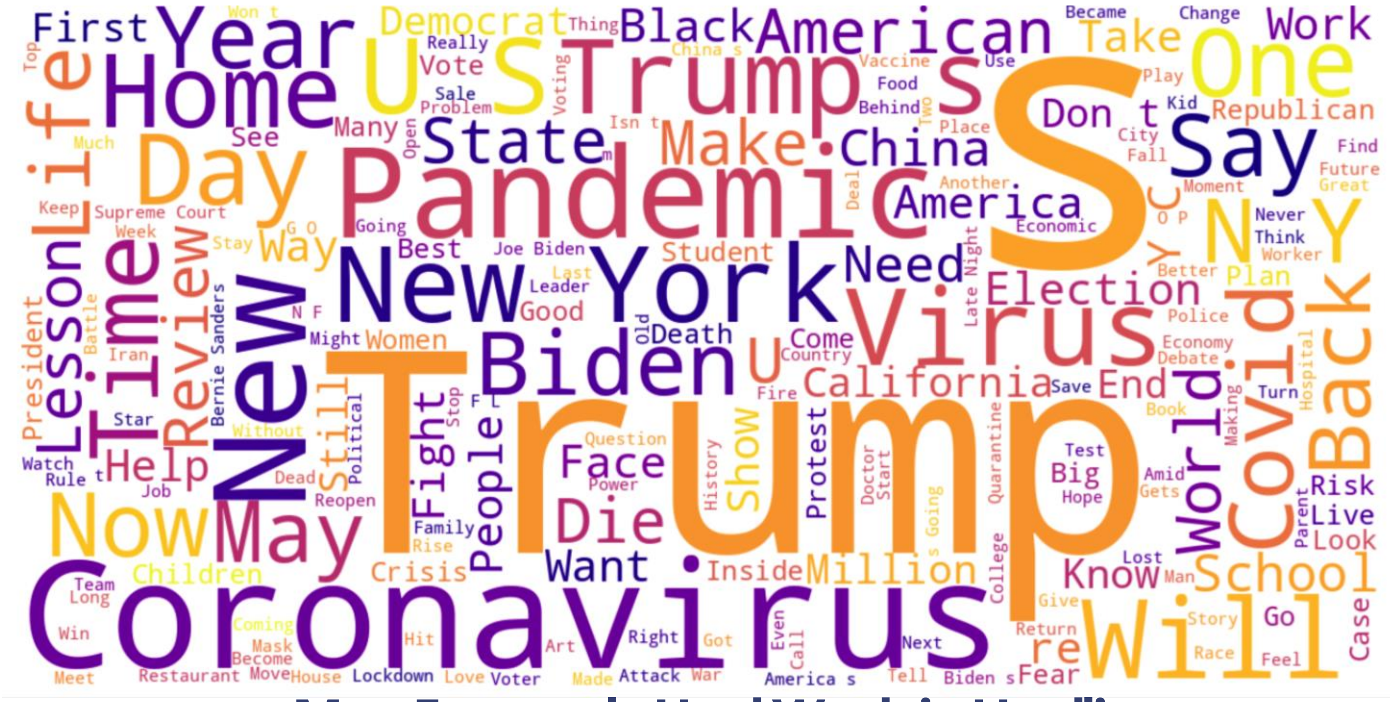
### Headline Generation

Generate headlines using LSTM neural networks

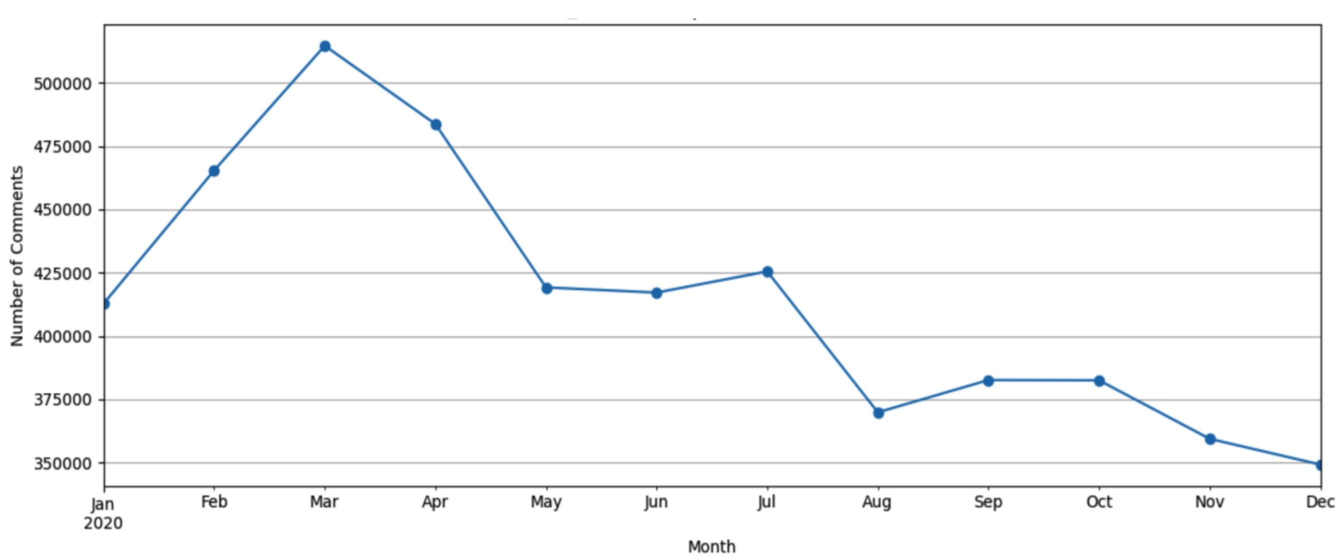
# Dataset Overview



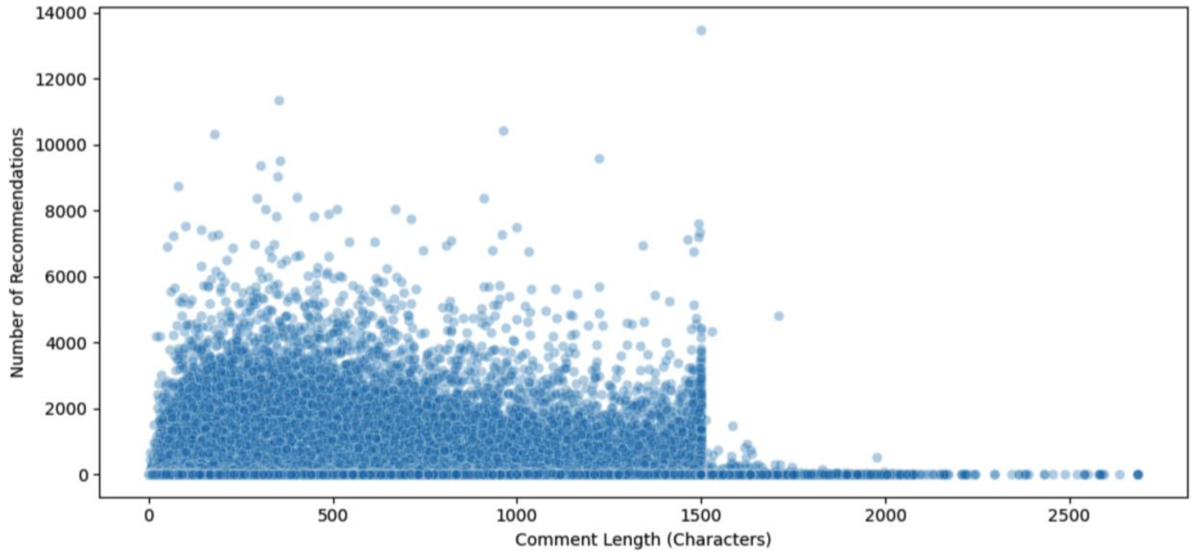

Most Frequently Used Words in Comments




Most Frequently Used Words in Headlines




Number of Comments Posted
in 2020 ( Month-wise )




Comment Length vs No. of Recommendations

# Task 1: Predicting No. of Comments Using Machine Learning

**Goal:** Use metadata from NYT articles to predict reader engagement (comment count).

**Pipeline:**

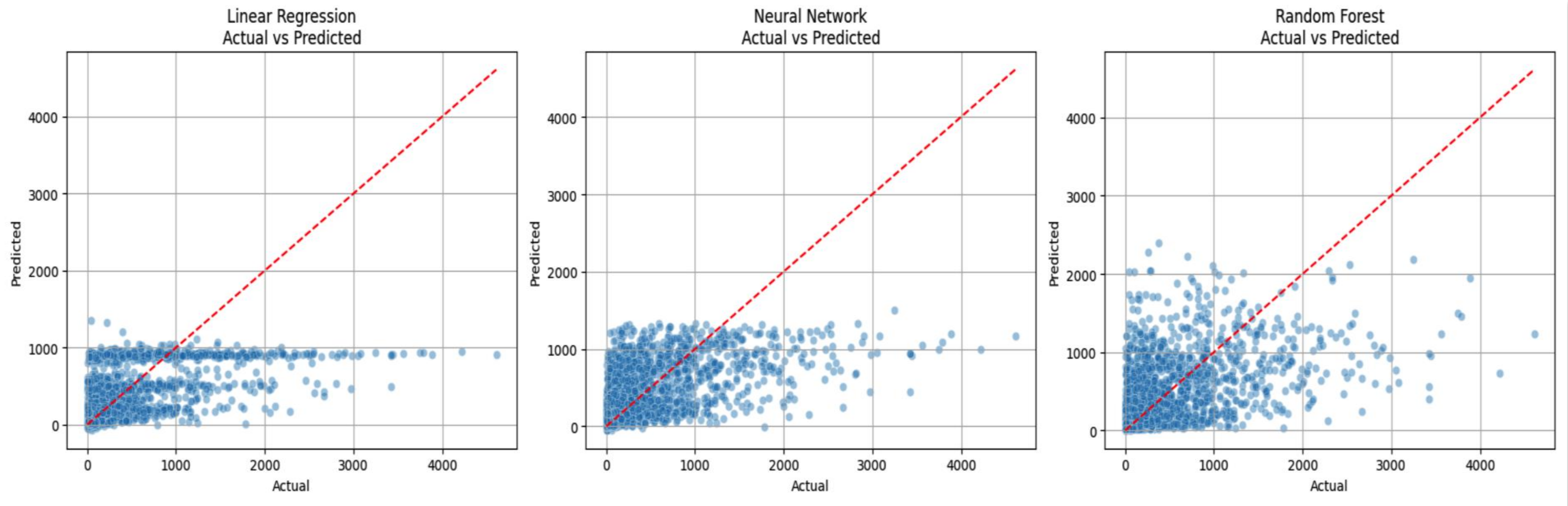
•Cleaned & engineered text features
•Built a scalable ML pipeline
•Models used: Linear Regression, Neural Network, Random Forest, XGBoost

**Feature Engineering (novelty):**

- Extracted word counts from the headline and abstract
- Created interaction term: word count × headline word count
- One-hot encoded the section categorical column
- Standard scaled numerical features for model compatibility

**Results After Feature engineering:**

| Model | $R^2$ Score |
|---|---|
| Linear Regression | 0.3361 |
| Neural Network | 0.3794 |
| Random Forest | 0.28 |
| XGBoost | 0.3710 |




**Target Variable Transformation:**

•The target variable (number of comments) had a **right-skewed distribution**
•Applied **logarithmic transformation**: y = log(1 + number of comments)
•Helped normalise the target for better model learning, Reduced the impact of extreme values (outliers)
•Resulted in a **significant improvement in $R^2$ score** and overall model performance

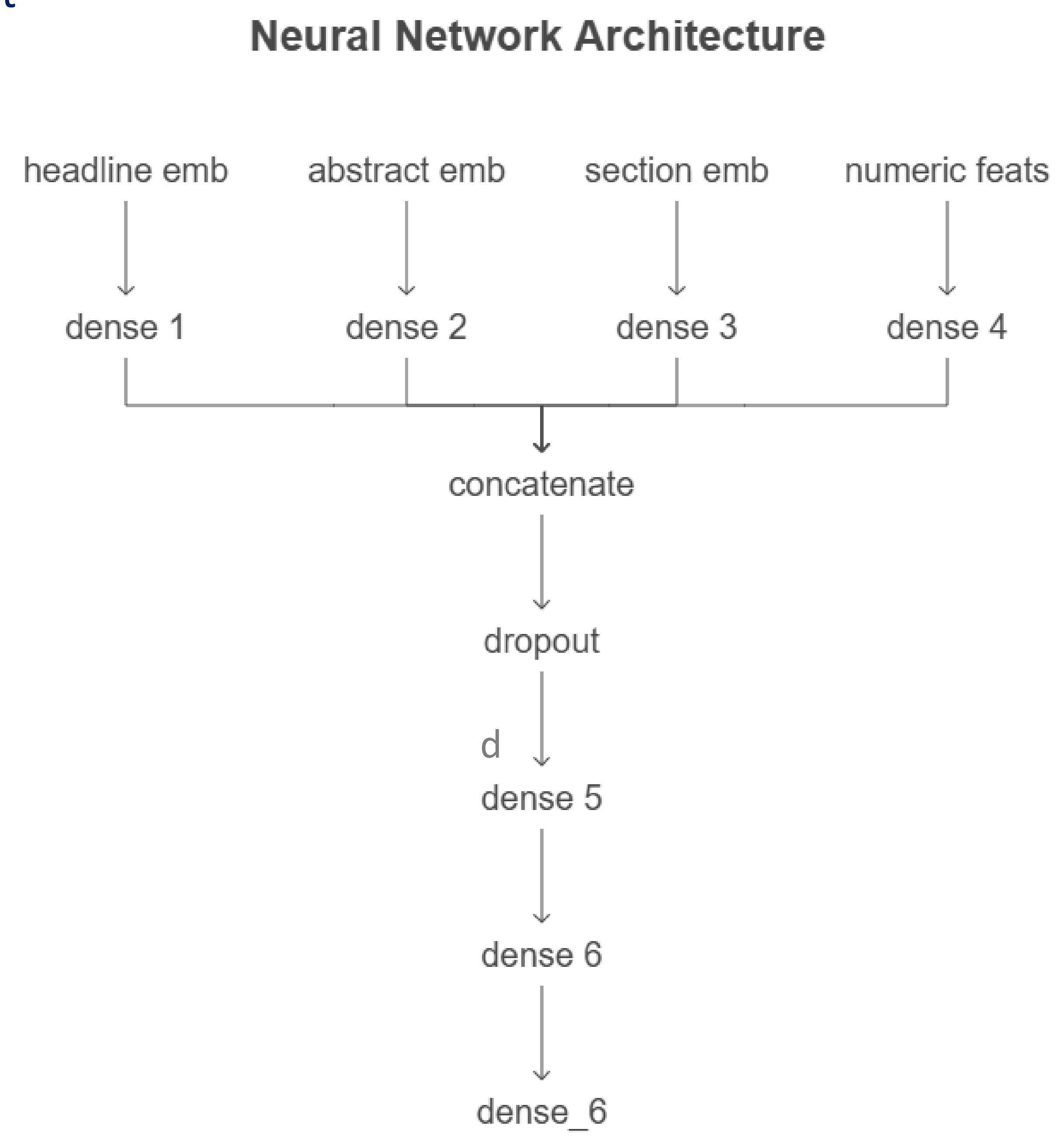
| Model | $R^2$ Score |
|---|---|
| Linear Regression | 0.4183 |
| Neural Network | 0.4712 |
| Random Forest | 0.4277 |
| XGBoost | 0.4713 |

# Task 1: Predicting No. of Comments Using Machine Learning

**TF-IDF Vectorization :**

•Applied **TF-IDF (Term Frequency–Inverse Document Frequency)** to convert text data into a numerical format
•Focused on headline and/or abstract columns for feature extraction
•Captured the importance of words relative to the entire dataset (not just raw counts)
•Reduced the influence of common words and highlighted meaningful terms
•Enabled models to understand text features like article titles and summaries
•Improved prediction performance by turning text into structured input

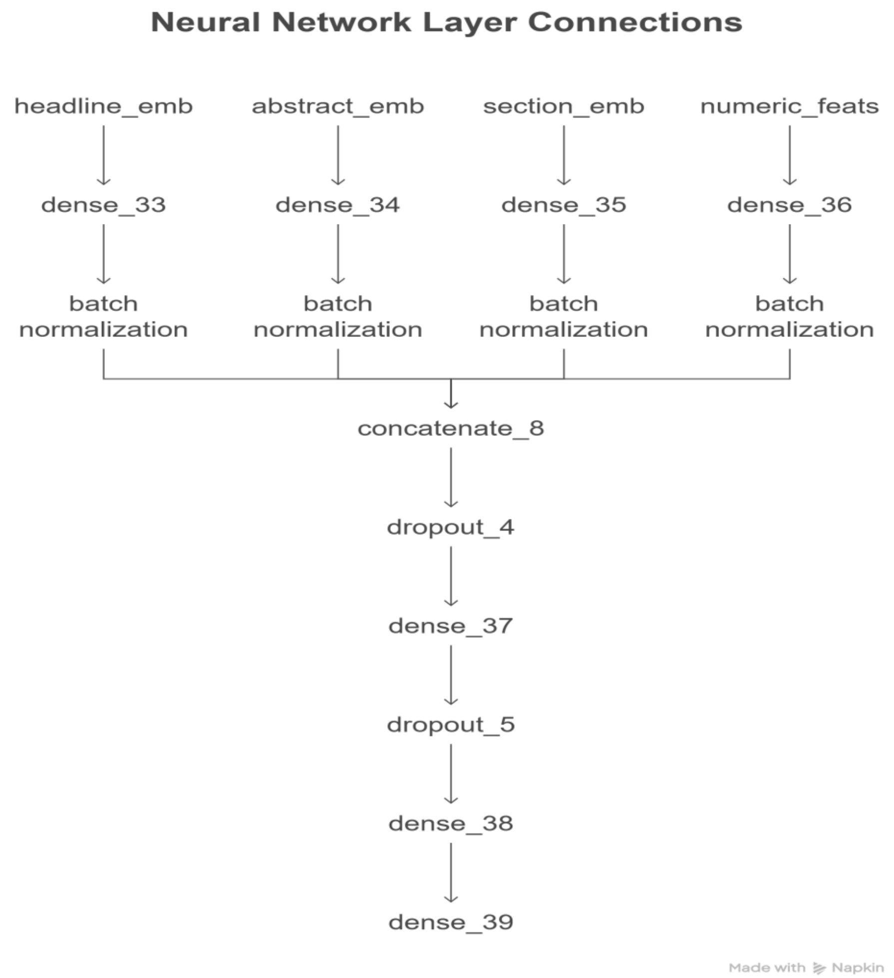
| Model | $R^2$ Score |
|---|---|
| Linear Regression | 0.4148 |
| Neural Network | 0.4722 |
| Random Forest | 0.4237 |
| XGBoost | 0.4658 |




**Architecture 1 using Fusion layers for embedding:**

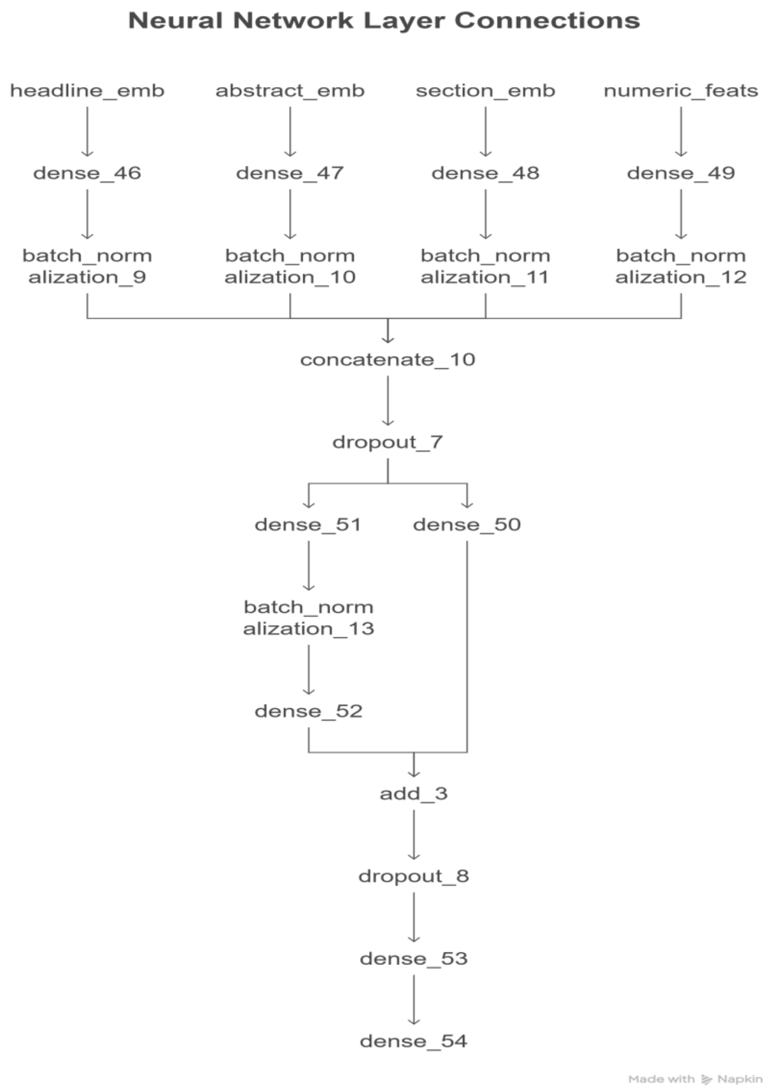
•Used **BERT** Model sentence-transformers All mini LM L6 v2 to generate embeddings
•Then used fusion dense Neural Network layers and applied dropout after concatenation.
•**$R^2$ Score: 0.49**

# Task 1: Predicting No. of Comments Using Machine Learning




**Architecture 2 using Fusion layers for embedding:**

- Used Batch Normalisation.
- **$R^2$ Score: 0.50** with more numerical features.




**Architecture 3 using Fusion layers for embedding:**

- Used Residual Learning and Skip Connection.
- **$R^2$ Score: 0.5325**

# Task 1: Predicting No. of Comments Using Machine Learning

## BERT Embeddings:

- Used **pre-trained BERT embeddings** to convert text (e.g., headlines) into dense vectors
- Captured **contextual meaning** of words based on their position and usage in a sentence
- Provided richer, more accurate text representation compared to traditional methods
- Used as input for **deep learning models**, boosting prediction accuracy

| Model | $R^2$ Score |
|---|---|
| Linear Regression | -3.01 e18 |
| Neural Network | -0.0637 |
| Random Forest | 0.5417 |
| XGBoost | 0.5474 |

## BERT with Feature Engineering:

- Extracted **BERT embeddings** from textual data like headlines and/or abstracts
- Engineered a custom **interaction term**: word count × headline word count
- Combined **BERT vectors** with structured features (e.g., interaction term, word counts) into a unified input
- Enabled models to learn from both **semantic text context** and **quantitative patterns**
- Resulted in improved model performance by leveraging both deep language understanding and structured signals

| Model | $R^2$ Score |
|---|---|
| Linear Regression | -1.04 e18 |
| Neural Network | -0.046 |
| Random Forest | 0.5419 |
| XGBoost | 0.5499 |

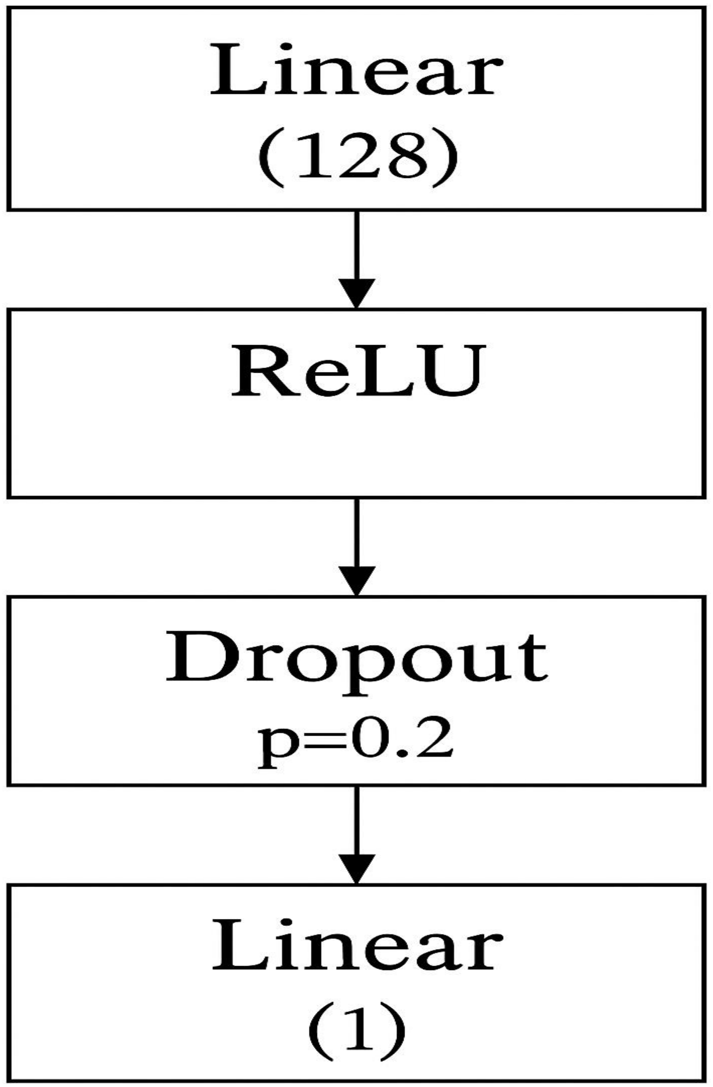
# Task 1: Predicting No. of Comments Using Machine Learning

## Tuned Bert Embeddings:

**Using Raw BERT / DistilBERT (AutoModel from Hugging Face)**
•Outputs last hidden states (sequence of token embeddings)
•Optionally includes the [CLS] token representation
•Provides **raw contextualized word representations**
•We get **token-level embeddings** for each input token
**Flexibility and Control:**
•We **fine-tune** the transformer model
•Multiple strategies to extract sentence-level representations:

- **Mean Pooling:** Average across token embeddings
- **[CLS] Token:** Use embedding of the [CLS] token
- **Attention Pooling:** Learnable attention weights to pool token representations

•Gives **more control** over how final features are formed
•**$R^2$ Score: 0.5589**

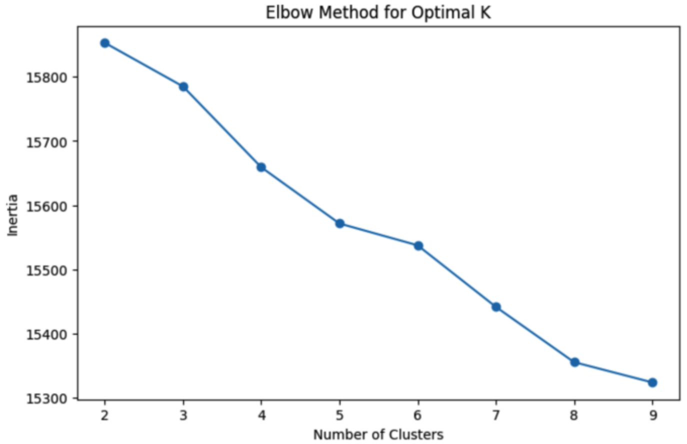


# Task 2: Unsupervised Topic Discovery in NY Times Headline (2020)

**Goal:** To identify hidden themes in NY Times headlines from 2020 using **clustering** and **topic modeling**

## Elbow Method

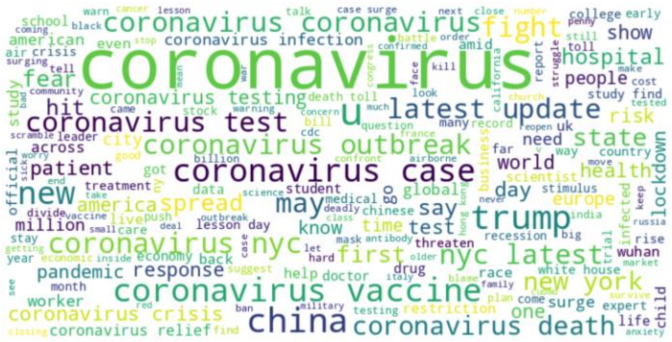
- The Elbow Method helps determine the optimal number of clusters by plotting inertia against different K values.
- Inertia decreases as the number of clusters increases, but the rate of decrease slows after a certain point.
- The "elbow" is observed at **K = 5**, indicating it as the optimal number of clusters.



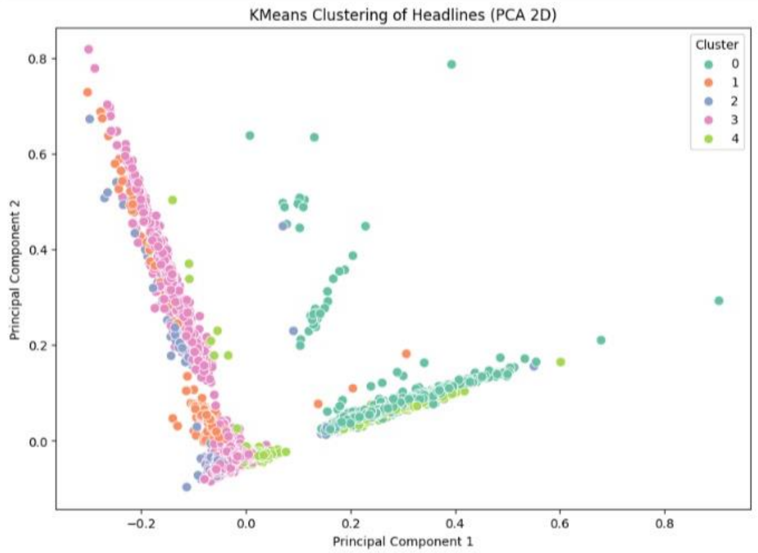

## K-Means Clustering:


Cluster 1


Cluster 2

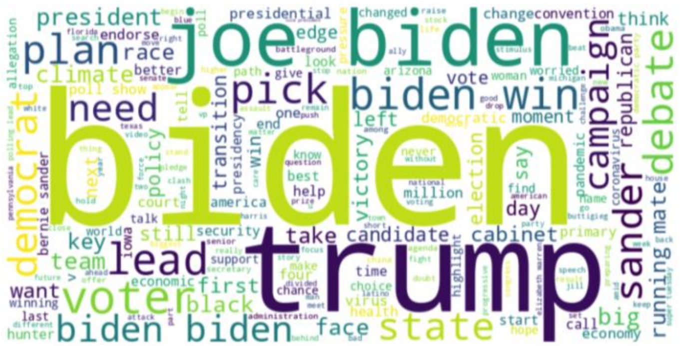



Cluster 3





Cluster 4

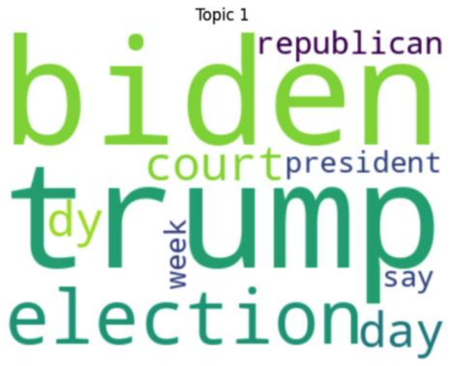



Cluster 5

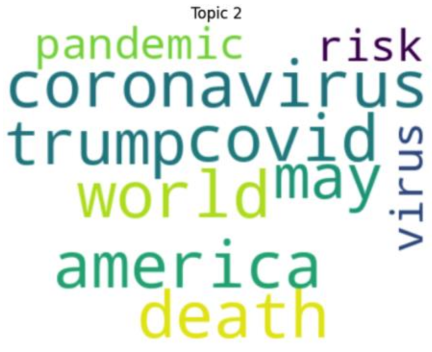
# Task 2: Unsupervised Topic Discovery in NY Times Headline (2020)

## LDA Clustering:

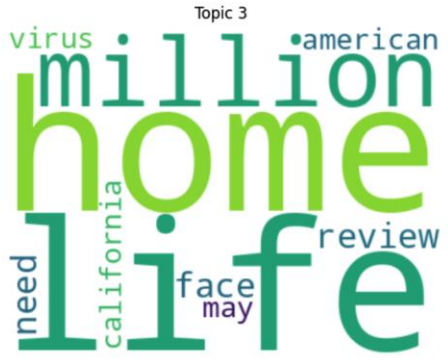

Cluster 1




Cluster 2




Cluster 3




Cluster 4


Cluster 5

### Comparing K-Means Clustering vs LDA Topic Modelling:

| Feature | **KMeans Clustering** | **LDA Topic Modeling** |
|---|---|---|
| **Clustering Type** | Hard clustering (each doc → 1 cluster) | Soft clustering (each doc → mix of topics) |
| **Vectorization Used** | TF-IDF (Term Frequency–Inverse Document Frequency) | Count Vectorizer (word frequency) |

```
                                                         headline  \
10156                Picasso Mural Torn From Building After Years of Dispute
15209  Nail Salons, Lifeline for Immigrants, Have Lost Half Their Business
5267                                                Kicked Out of China
11723                      Tiwa Savage, Queen of Afrobeats, Makes a New Start
16039                          2020: The Year in Sports When Everybody Lost

       cluster   Topic_0   Topic_1   Topic_2   Topic_3   Topic_4
10156        3  0.733332  0.066667  0.066667  0.066667  0.066667
15209        3  0.050379  0.051229  0.549085  0.299089  0.050218
5267         3  0.100000  0.100000  0.100993  0.101753  0.597254
11723        3  0.302284  0.040579  0.361586  0.040615  0.254936
16039        3  0.798221  0.050450  0.051099  0.050230  0.050000
```

# Task 3: Deep Learning-Based Text Generation on NYT Dataset

**Goal**: Generate news headlines using a Long Short-Term Memory (LSTM) neutral network.

**Embedding**:

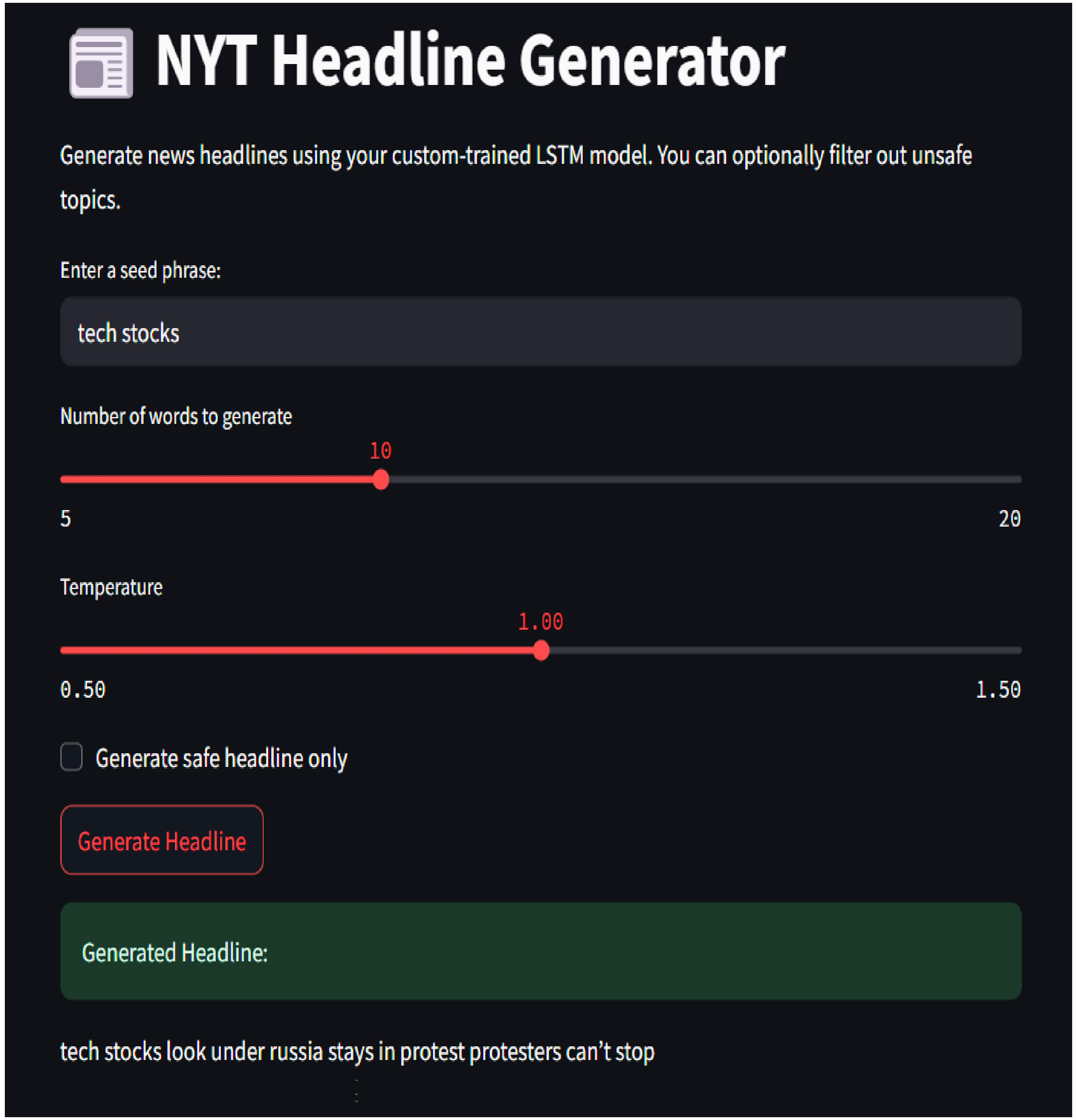
- Used **Keras Tokenizer** to convert text into integer sequences (word index).
- Passed tokenized input through an **Embedding layer** which maps each word index to a dense vector, capturing semantic relationships between words.

**Model Training**:

Input → Embedding Layer → LSTM Layer → Dense Layer (ReLU) → Dense Layer (Softmax)

**Semantic Safety Filtering**:

- Used a zero-shot classification pipeline from Hugging Face Transformers.
- Each generated headline was checked against a list of sensitive topics: e.g. violence, tragedy, politics, hate speech
- **e.g. "tech stock look under Russia stays in protest protesters can't stop."** (without safe mode)



# Task 3: Deep Learning-Based Text Generation on NYT Dataset

**Goal**: Generate news headlines using a Long Short-Term Memory (LSTM) neutral network.

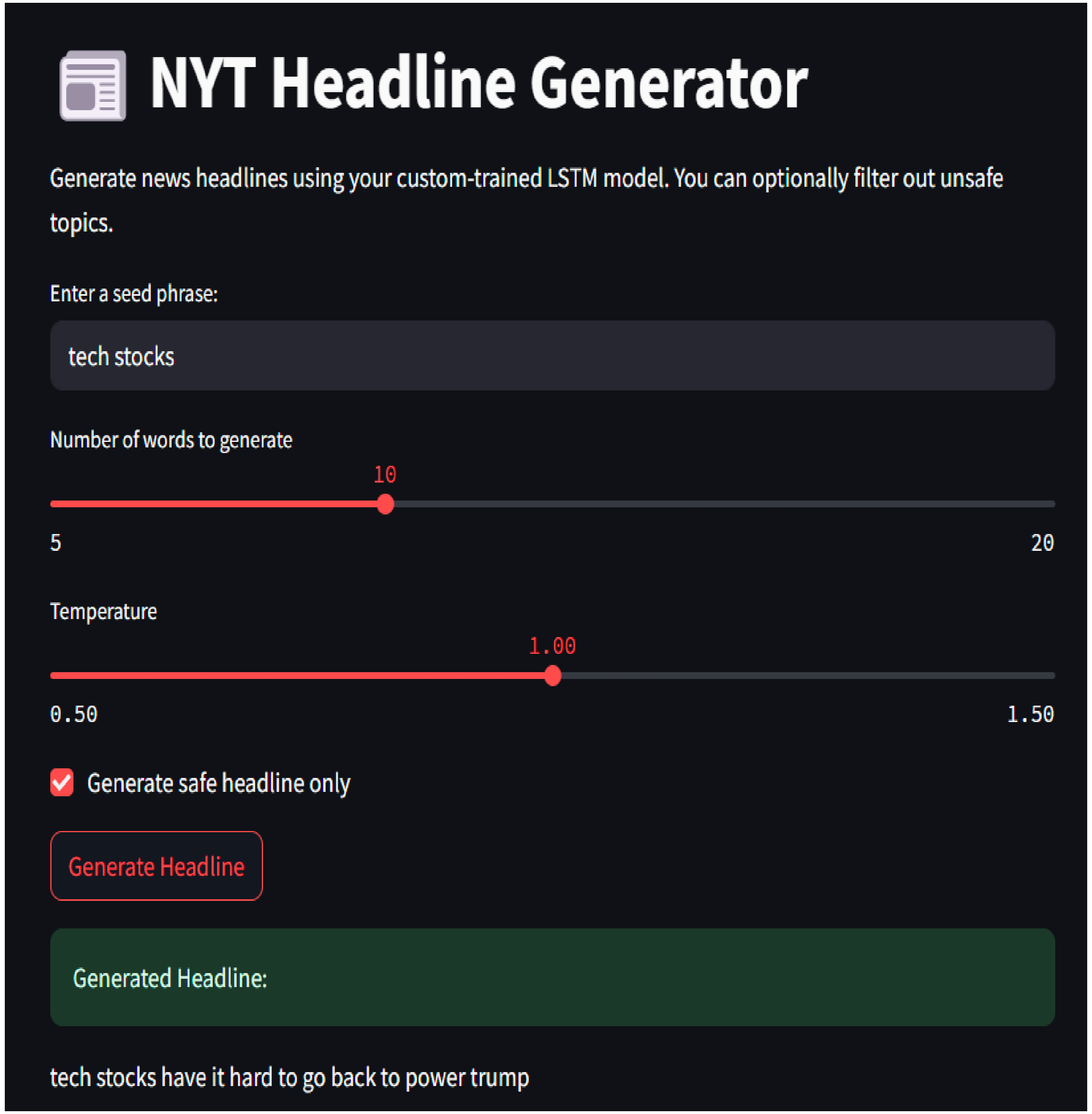
Headline Generation:

- Input a seed phrase.
- Predicts one word at a time using previously generated words.
- Used temperature sampling:
  - Low temperature is more focused & repetitive
  - High temperature is more diverse & creative
- Controls for Generation: Number of Words, Safe Mode

Example:

User Seed: tech stocks

Generated Headlines: "***tech stocks have it hard to go back to power trump***" *(with safe mode)*



# Task 4: Editor's Pick Prediction

## **Goal**: Predict whether a New York Times comment will be selected as an **Editor's Pick.**

**Initial Experiments**

**50,000 Points Dataset (XG Boost Classifier Model)**

- Balanced Performance with Class 0 Leading: Achieved 92% accuracy with solid performance for Class 0.
- Class 0 Precision: 0.94, Recall: 0.95, F1-score: 0.95
- Class 1 Precision: 0.85, Recall: 0.82, F1-score: 0.84
- Strong Weighted Performance: Weighted F1-score: 0.92, indicating a balanced overall model despite class imbalance.

**Full Dataset (Times Pick Data)**

- High Overall Accuracy with Class Imbalance: Achieved 96.74% accuracy, though impacted by class imbalance.
- Class 0 Precision: 0.9889, Recall: 0.9780, F1-score: 0.9834
- Class 1 Precision: 0.0257, Recall: 0.0502, F1-score: 0.0340
- Imbalance Challenge: Model is heavily biased toward Class 0, requiring strategies to improve Class 1 detection.

•**BERT-Based Modelling**

•**BERT + metadata model**

  - Significant improvement in recall for class 1: 85.91%
  - Precision: 0.1627, Recall: 0.8591,
  - F1-score: 0.2736
  - Maintained strong overall performance (accuracy: 94.78%)
  - Demonstrated that combining contextual text modelling with metadata significantly improves minority class performance

| | **precision** | **recall** | **F1-score** |
|---|---|---|---|
| 0 | 0.94 | 0.95 | 0.95 |
| 1 | 0.85 | 0.82 | 0.84 |
| Accuracy | | 0.92 | |
| Macro avg | 0.9 | | |
| Weighted avg | 0.92 | | |

| | **precision** | **recall** | **F1-score** |
|---|---|---|---|
| 0 | 0.9889 | 0.9780 | 0.9834 |
| 1 | 0.0257 | 0.0502 | 0.0340 |
| Accuracy | | | 0.9674 |

| | **precision** | **recall** | **F1-score** |
|---|---|---|---|
| 0 | 0.9983 | 0.9488 | 0.9729 |
| 1 | 0.1627 | 0.8591 | 0.2736 |
| Accuracy | | 0.9478 | |
| Macro avg | 0.5805 | | |
| Weighted avg | 0.9887 | | |

•**Accuracy : 0.9478**

# Task 4: Editor’s Pick Prediction

**Goal**: Predict whether a New York Times comment will be selected as an **Editor’s Pick.**

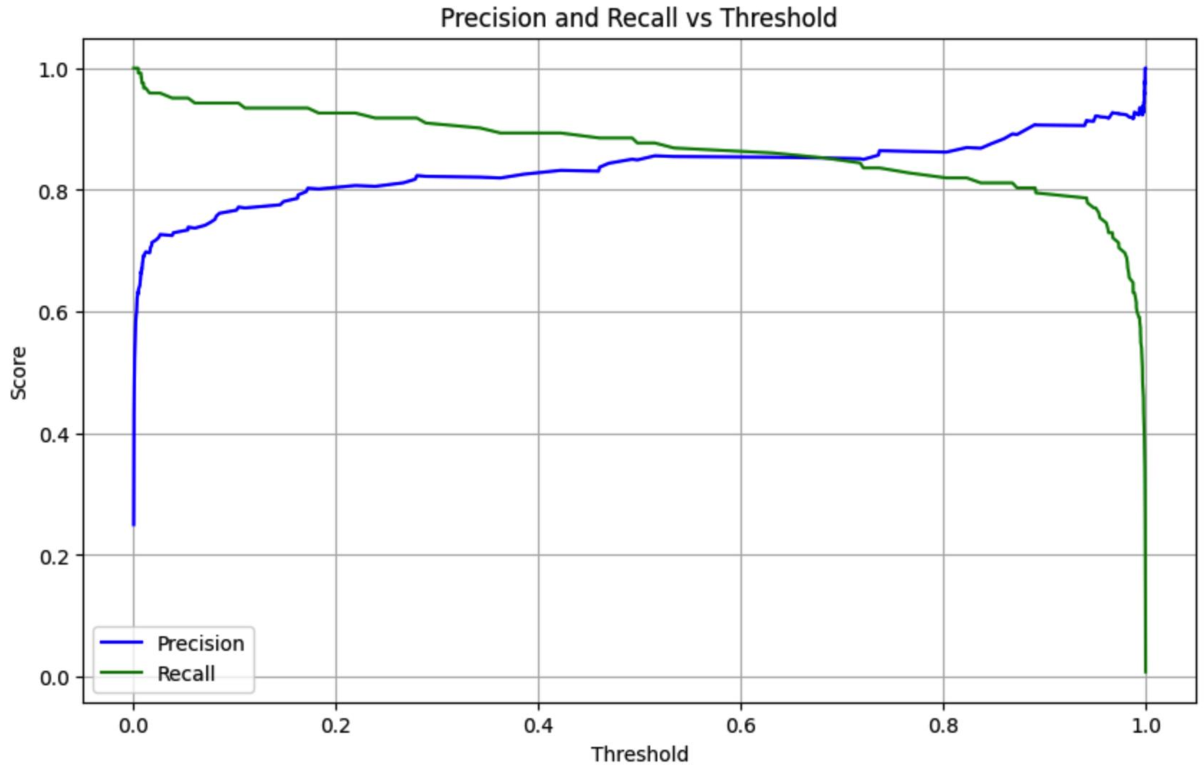
**Threshold Tuning Phase**

•**Performed precision-recall vs threshold analysis**

- Adjusted the classification threshold (0.7) to better balance recall and precision
- Retrained the model using the optimized threshold

•**Final model performance:**

- Class 1 (Editor’s Picks):
  - Precision: 0.2590
  - Recall: 0.7523
  - F1-score: 0.3853 (highest so far)
- Class 0 retained high performance (F1-score: 0.9860)
- Overall accuracy: 97.25%
- Macro F1-score: 0.6856
- Weighted F1-score: 0.9791




**Techniques and Key Insights**

•Addressed a highly imbalanced classification problem

•Leveraged BERT for advanced contextual understanding of text

•Augmented BERT with metadata features for better model grounding

•Tuned classification thresholds using precision-recall trade-off analysis

•Iteratively improved model performance on both majority and minority classes

•Achieved a well-balanced, production-ready model with high accuracy and strong recall for editor’s picks

| | **precision** | **recall** | **F1-score** |
|---|---|---|---|
| 0 | 0.9971 | 0.9751 | 0.9860 |
| 1 | 0.2590 | 0.7523 | 0.3853 |
| Accuracy | | | 0.9725 |
| Macro avg | 0.6280 | 0.8637 | 0.6856 |
| Weighted avg | 0.9886 | 0.9725 | 0.9791 |

•**Accuracy : 0.9725**

# Future Work

- Ensemble Methods: Apply ensemble techniques (e.g., stacking, boosting) to combine multiple models for more.
- Model Improvements: Replace or augment LSTM models with Transformer-based architectures (e.g., GPT, T5) to generate higher-quality, more coherent headlines.
- Model the full thread or parent-child structure of comments using techniques like graph neural networks (GNNs)