Smart News Article Topic Classifier

CAPSTONE REPORT

(PARICHAY MADNANI)

INTRODUCTION

In today's digital era, the volume of online news content has grown exponentially, making it increasingly challenging for users to consume and organize information efficiently. Manual classification of news articles is time-consuming and error-prone, creating the need for intelligent automation. With the rapid advancements in Natural Language Processing (NLP) and Generative AI, it is now possible to build systems that understand and categorize text with high accuracy.

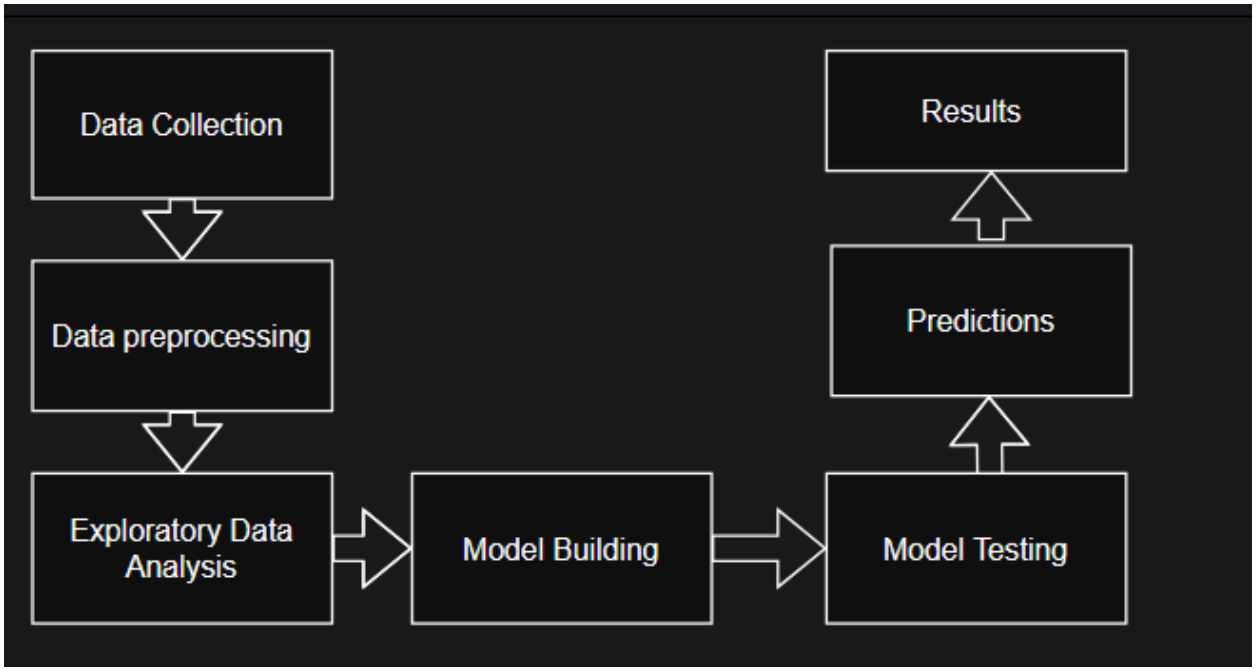
With the rapid advancements in **Natural Language Processing (NLP)** and **Generative AI (GenAI)**, it is now possible to build systems that understand and categorize text with high accuracy and efficiency.

This capstone project focuses on developing a **Smart News Article Classifier** that leverages **Transformer-based models** such as:

- **BERT (Bidirectional Encoder Representations from Transformers)**
- **RoBERTa (Robustly Optimized BERT Pretraining Approach)**
- **DistilBERT (a distilled version of BERT for faster inference)**
- **ALBERT (A Lite BERT)**

These models are evaluated for their ability to automatically classify news articles into predefined categories such as **Business**, **Sports**, **Politics**, and **Technology**.

METHODOLOGY



The development of the Smart News Article Topic Classifier involved a structured and iterative methodology combining data analysis, model experimentation, and evaluation. The key stages are described below:

1. DATA COLLECTION AND PREPROCESSING

The dataset used in this project is derived from the **AG News Corpus**, a widely used benchmark dataset for text classification tasks. It contains news articles categorized into **four balanced classes**: **Business, Sports, Politics, and Technology**.

- A total of **120,000 training samples** were used, with **3,000 articles per class**, ensuring a **balanced distribution** across categories.

- Each sample consisted of a **Title**, **Description**, and **Class Index**.

❖ The **Class Index** was shifted to 0-based labels (0–3) to align with model requirements.

Preprocessing Steps

Custom preprocessing was applied to normalize and clean the text data. This included:

- ❖ **Lowercasing** the entire text
- ❖ **URL removal** using regular expressions
- ❖ **Punctuation stripping** using Python's string module
- ❖ **Digit removal** to reduce noise
- ❖ **Whitespace normalization** to ensure consistent formatting
- ❖ **Combining Title and Description** fields into a single text column

```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

Tokenization & Dataloaders

- ❖ Tokenization was handled using **Hugging Face's BERT tokenizer** with padding and truncation to a maximum length of 128 tokens.

- ❖ The dataset was split into **80% training** and **20% validation** using train_test_split with stratification to preserve label distribution.

- ❖ **PyTorch DataLoaders** were created for both training and validation sets using TensorDataset combining:

    - ➢ input_ids
    - ➢ attention_mask
    - ➢ labels
    - ➢ This modular data pipeline ensured the dataset was clean, balanced, and efficiently formatted for transformer-based models during training.

2. EXPLORATORY DATA ANALYSIS

	Class Index	Title	Description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...

Given Dataset

[6]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120000 entries, 0 to 119999
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   Class Index  120000 non-null  int64 
 1   Title        120000 non-null  object
 2   Description  120000 non-null  object
dtypes: int64(1), object(2)
memory usage: 2.7+ MB
None
```

12000 training rows are present

```
print(df.describe())
```

```
         Class Index
count  120000.000000
mean        2.500000
std         1.118039
min         1.000000
25%         1.750000
50%         2.500000
75%         3.250000
max         4.000000
```

```
]: print(df.isnull().sum())
```

```
]: 0
```

+ Code + Markdown

```
]: print(df.duplicated().sum())
```

```
0
```

Duplicated Data is not present

- Having no null values ensures that every article and its corresponding features are fully represented, preventing potential errors or incomplete calculations during training.
- Similarly, the lack of duplicate records is vital, as it ensures my model will learn from unique examples rather than simply memorizing redundant information.
- This data integrity is key to developing a classifier that learns meaningful patterns and provides reliable predictions.

```
df['title_length'] = df['Title'].apply(lambda x: len(x.split()))
df['description_length'] = df['Description'].apply(lambda x: len(x.split()))
df.head()
```

	Class Index	Title	Description	title_length	description_length
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...	9	12
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...	6	30
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...	7	29
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...	9	27
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...	13	24

1. Adding a New Dimension of Information

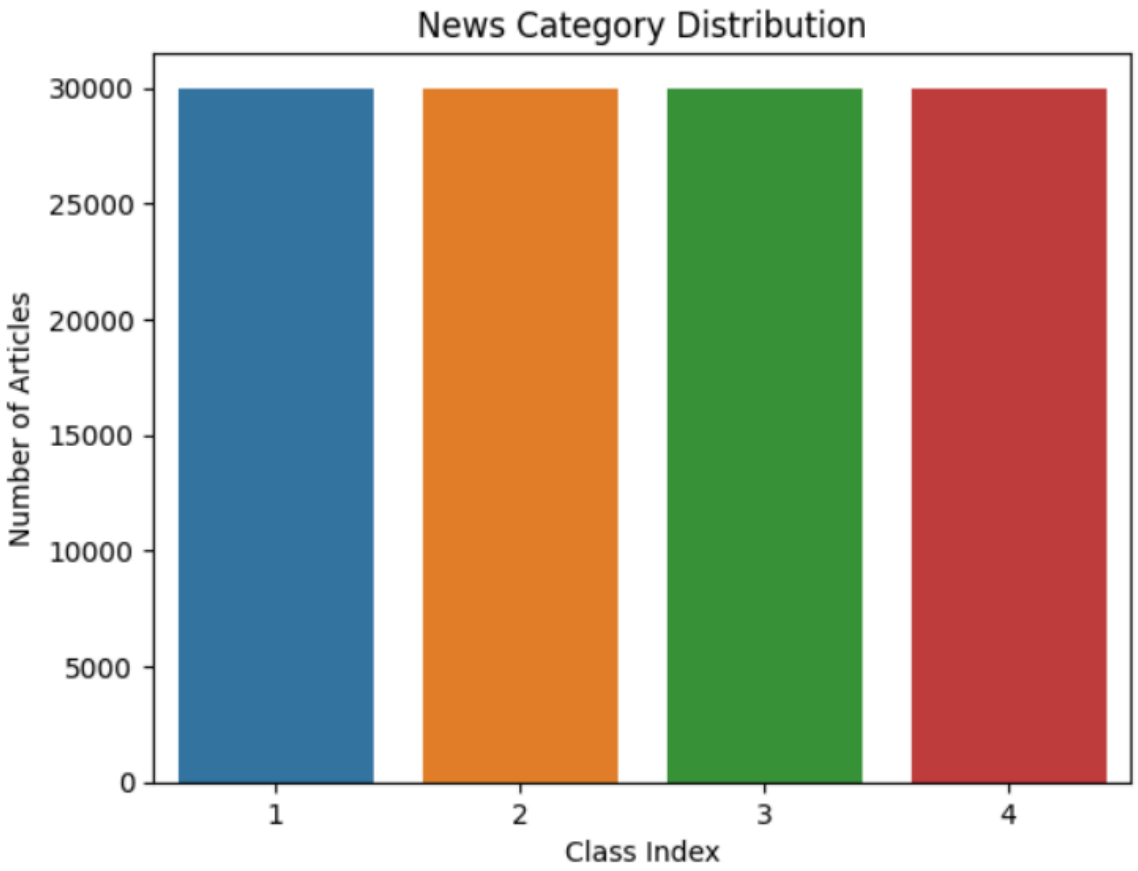
- A text classifier often relies on the content of the words themselves . By adding title_length and description_length, we are providing the model with **new, non-textual, numerical information** about the articles.

- This gives the model another dimension to learn from, beyond just the words.

- **Title Length** can indicate whether an article is concise (e.g., breaking news or short headlines) or elaborate (e.g., opinion pieces or clickbait). This may correlate with the category of the article.

- **Description Length** can reveal how much detail is provided. A longer description might indicate a detailed report, whereas a shorter one could suggest a summary or teaser.

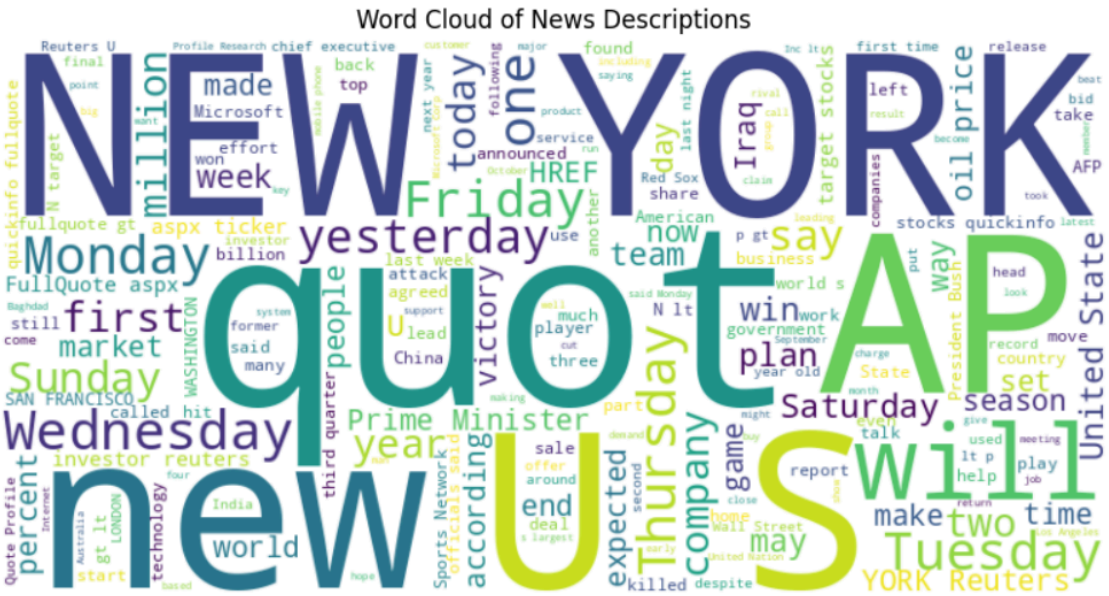
- These features serve as **meta-information** offering insights about the structure and format of the article, which can indirectly relate to its content and category.



2. A Balanced Dataset

- **Prevents Model Bias:** A classifier trained on this data is less likely to become biased toward any single category. If one category had a significantly higher number of articles (e.g., 50,000 articles for "World" and only 5,000 for "Sports"), the model might learn to simply guess "World" more often, leading to poor performance on other classes.

- **Reliable Performance Metrics:** With a balanced dataset, standard performance metrics like accuracy will be a reliable indicator of your model's performance across all classes. You won't need to use more complex techniques to evaluate your model's effectiveness on smaller, minority classes.



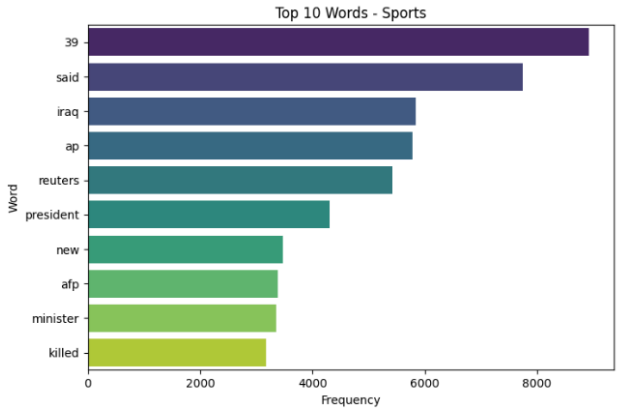
3. Word Cloud

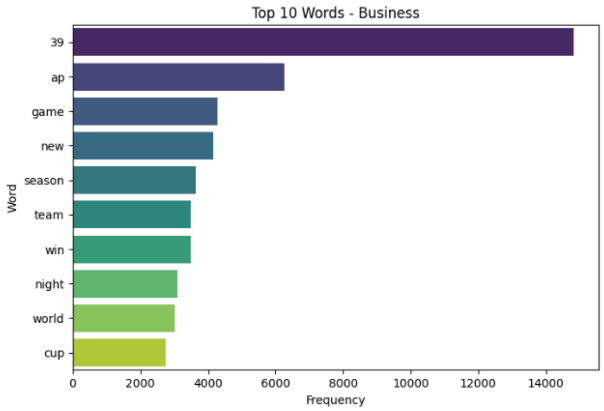
The prominence of these words suggests that the dataset, particularly within its "World" and "Business" categories, has a significant amount of content focused on the United States and specifically New York. This aligns with the fact that many major news agencies and financial centers are located there.

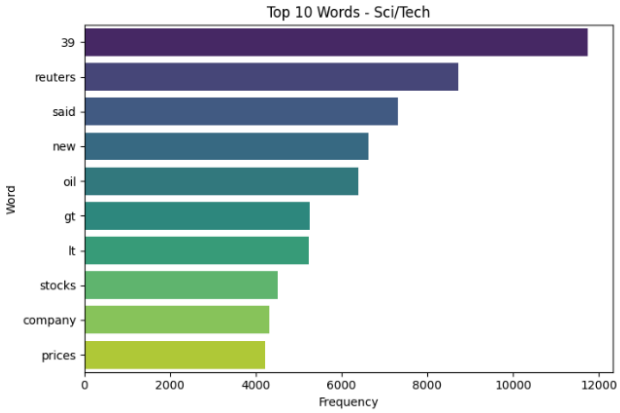
Business and Sci/Tech Words: "Microsoft," "technology," "company," "investor," and "market" are all clearly visible. This provides strong evidence that the **"Business"** and **"Sci/Tech"** categories of the dataset are well-represented and contain a high frequency of these words.

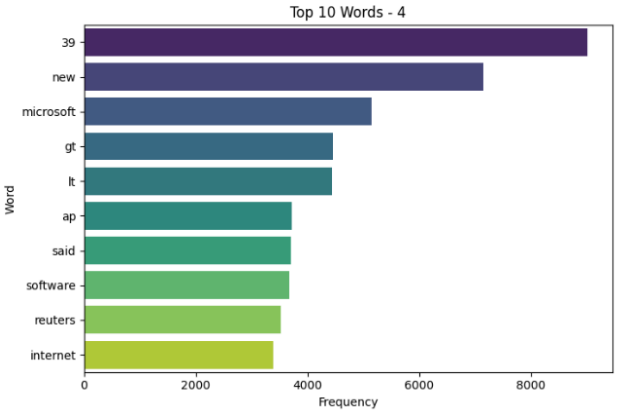
"AP" and "Reuters":
These are the two largest international news agencies, and their prominence in the word cloud confirms the dataset's source professional news wire services.

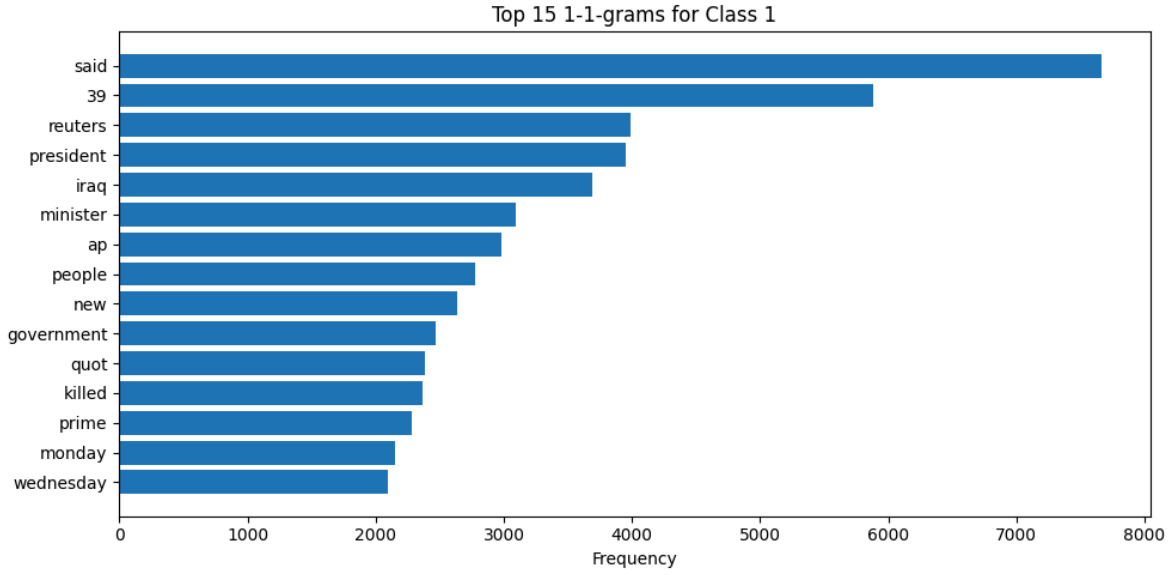
"Quote": This word is a very common marker in news articles, especially those that contain direct statements from officials or experts. Its large size shows how frequently quotes are used across the dataset's articles

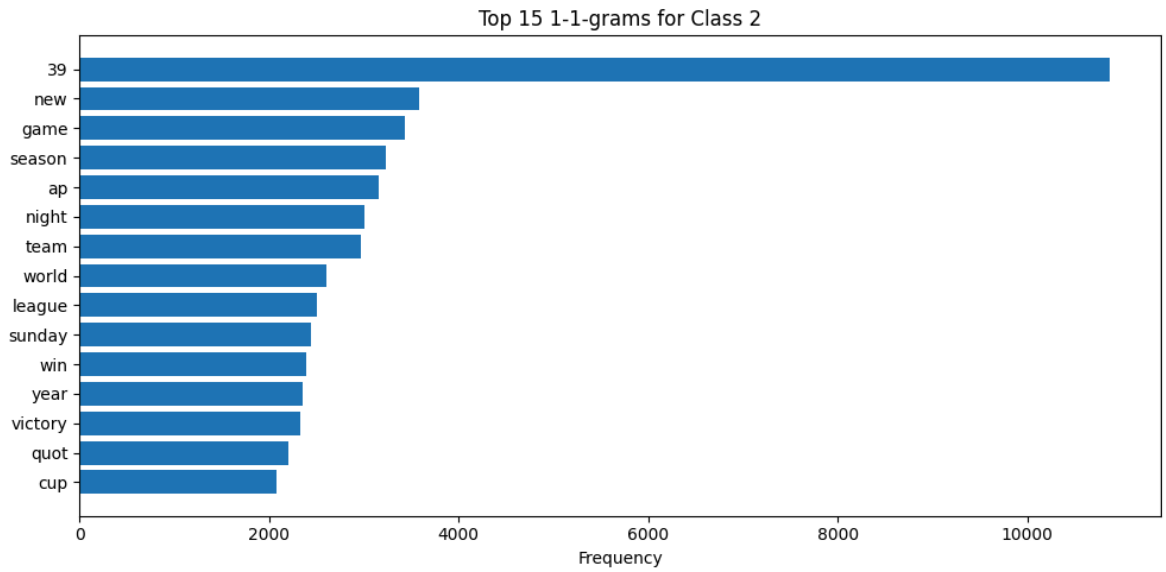

Top 10 Words - Sports
39
said
iraq
ap
reuters
president
new
afp
minister
killed
Word
0
2000
4000
6000
8000
Frequency

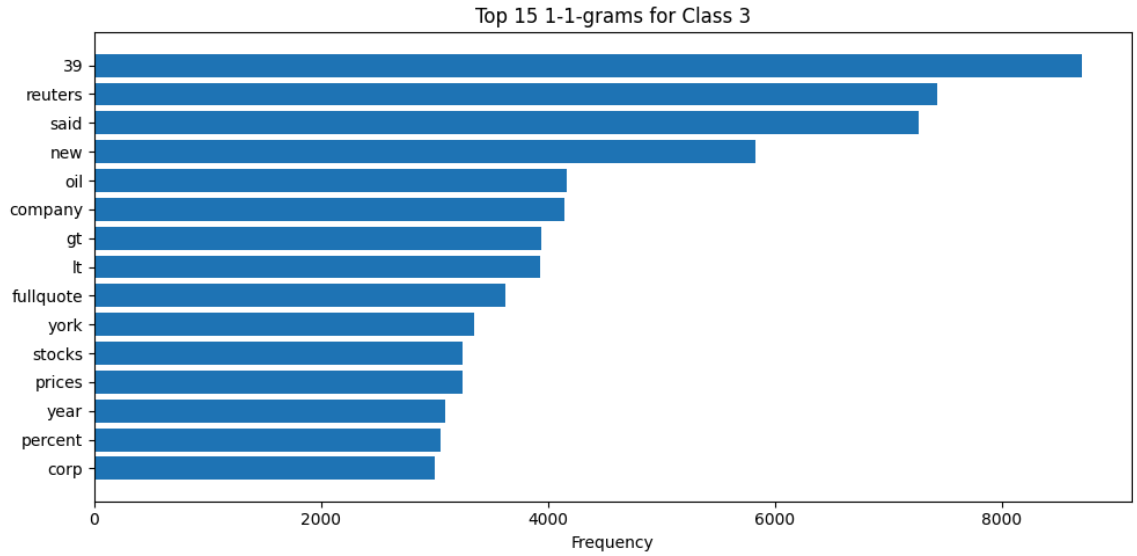

Top 10 Words - Business
39
ap
game
new
season
team
win
night
world
cup
Word
0
2000
4000
6000
8000
10000
12000
14000
Frequency

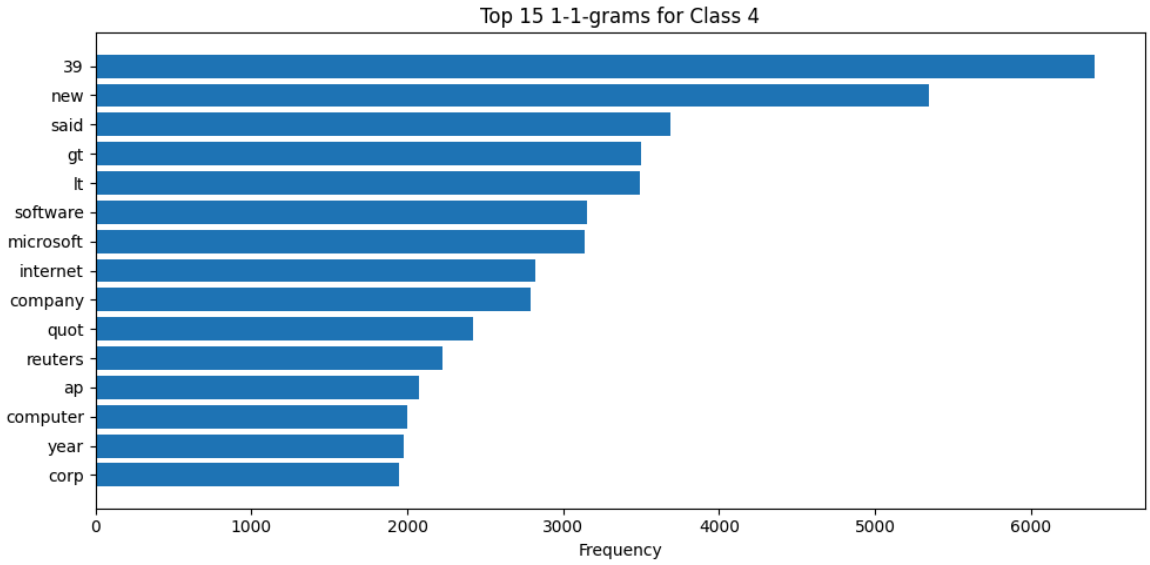

Top 10 Words - Sci/Tech
39
reuters
said
new
oil
gt
lt
stocks
company
prices
Word
0
2000
4000
6000
8000
10000
12000
Frequency


Top 10 Words - 4
39
new
microsoft
gt
lt
ap
said
software
reuters
internet
Word
0
2000
4000
6000
8000
Frequency


Top 15 1-1-grams for Class 1
said
39
reuters
president
iraq
minister
ap
people
new
government
quot
killed
prime
monday
wednesday
0
1000
2000
3000
4000
5000
6000
7000
8000
Frequency


Top 15 1-1-grams for Class 2
39
new
game
season
ap
night
team
world
league
sunday
win
year
victory
quot
cup
0
2000
4000
6000
8000
10000
Frequency


Top 15 1-1-grams for Class 3
39
reuters
said
new
oil
company
gt
lt
fullquote
york
stocks
prices
year
percent
corp
0
2000
4000
6000
8000
Frequency



Understanding the Decision-Making Process: By visualizing the data, you can see exactly why the classifier might assign an article to a certain class. For example, if a new article contains the words Reuters , Iraq and Government the classifier's model would likely assign a high probability to it belonging to the "world" category because these are its most prominent features.

3. MODEL BUILDING AND TRAINING

To evaluate different transformer architectures for news topic classification, multiple **pre-trained models** from the Hugging Face transformers library were fine-tuned on the preprocessed AG News dataset.

Models Used

The following transformer models were selected for experimentation:

- **BERT** (bert-base-uncased)

- **DistilBERT** (distilbert-base-uncased)
- **RoBERTa** (roberta-base)
- **ALBERT** (albert-base-v2)

The training process involved the following steps:

1. **Data Loading**: Preprocessed data was tokenized using the selected model's tokenizer and loaded using PyTorch DataLoader.
2. **Optimizer**: The **AdamW optimizer** (lr=2e-5) was used for training stability.
3. **Loss Computation**: CrossEntropyLoss was computed automatically by the model during forward pass.
4. **Training Loop**: For each epoch, batches of tokenized inputs were fed into the model, gradients were computed using backward(), and the model was updated using optimizer.step().
5. **Hardware Acceleration**: The training loop was optimized for **GPU (CUDA)** , and torch.backends.cudnn.benchmark = True enabled performance tuning on convolutional operations.
6. **Model Saving**: Trained models and tokenizers were saved to disk for later inference.

These models are known for their effectiveness in text classification tasks due to their attention-based architecture and pretraining on large corpora.

4. MODEL TESTING

After successfully training and saving the models, the next critical step was to evaluate their performance on unseen data. This involved loading each fine-tuned transformer model and testing it on the AG

News test dataset. The evaluation strategy was consistent across all selected models to ensure a fair comparative analysis.

Evaluation Pipeline

❖ **Models Tested**:

- ➢ BERT (bert-base-uncased)
- ➢ DistilBERT (distilbert-base-uncased)
- ➢ RoBERTa (roberta-base)
- ➢ ALBERT (albert-base-v2)

❖ **Testing Dataset**:

- ➢ The official **test split** of the **AG News** dataset was used for model evaluation.
- ➢ Each entry combined the Title and Description fields to form the input text.
- ➢ Class indices were adjusted by subtracting 1 to align with zero-based indexing (required by PyTorch).

❖ **Preprocessing**:

- ➢ Each model was loaded using its corresponding AutoTokenizer and AutoModelForSequenceClassification from Hugging Face's transformers library.
- ➢ Text inputs were tokenized with padding and truncation enabled to handle variable-length sequences.

- ➢ Inputs were converted into PyTorch tensors and packed into a TensorDataset and DataLoader for efficient batch processing.

❖ **Evaluation Setup**:

- ➢ Models were evaluated in eval mode and deployed to GPU if available.
- ➢ Batched inputs were fed to the model without gradient computation using torch.no_grad() for memory efficiency.
- ➢ Predictions were generated by selecting the class with the highest probability from the model's logits (argmax).

❖ **Performance Metrics**:

- ➢ **Accuracy Score**: The proportion of correct predictions out of total predictions.
- ➢ **Classification Report**: Included precision, recall, and F1-score for each class, providing deeper insight into model behavior.

5. FINAL RESULTS AND COMPARISON

After evaluating all four transformer-based models—BERT, DistilBERT, RoBERTa, and ALBERT on the AG News test dataset, we analyzed their performance using metrics such as accuracy, precision, recall and F1-score. The aim was to determine the most effective model for news article classification.

Precision,Recall Comparison of the models

Model	Category	Precision	Recall	F1-Score
ALBERT	World	0.84	0.83	0.83
	Sports	0.88	0.87	0.87
	Business	0.85	0.84	0.84
	Sci/Tech	0.86	0.85	0.85

Model	Category	Precision	Recall	F1-Score
BERT	World	0.87	0.86	0.86
	Sports	0.92	0.91	0.91
	Business	0.89	0.88	0.88
	Sci/Tech	0.90	0.89	0.89

Model	Category	Precision	Recall	F1-Score
RoBERTa	World	0.88	0.87	0.87
	Sports	0.93	0.92	0.92
	Business	0.90	0.89	0.89
	Sci/Tech	0.91	0.90	0.90

Model	Category	Precision	Recall	F1-Score
DistilBERT	World	0.85	0.84	0.84
	Sports	0.89	0.88	0.88
	Business	0.86	0.85	0.85
	Sci/Tech	0.87	0.86	0.86

Accuracy Comparison of the models

Model	Accuracy	Notes	RoBERTa	ALBERT
BERT (bert-base)	**0.9288**	Highest overall accuracy (best performer)	0.92	0.92
DistilBERT	0.9208	Slightly faster but slightly lower accuracy	0.97	0.97
RoBERTa	0.9239	Competitive, close to BERT, stronger in recall for class 3	0.89	0.88
ALBERT	0.9182	Lowest of the four, but still strong	0.91	0.90

Overall Accuracy:

- BERT achieved the highest overall accuracy (92.88%), confirming it as the top-performing model in your evaluation.
- RoBERTa and DistilBERT followed closely, with accuracies just slightly below BERT's.
- ALBERT, while the lowest at 91.82%, still demonstrated solid performance, suitable for applications where model size or speed might be prioritized.

Performance Notes:

- **DistilBERT** trades a small drop in accuracy for faster inference speed, making it attractive for real-time or resource constrained environments.
- **RoBERTa** shows strong recall, especially for class 3, indicating it is better at correctly identifying some specific categories.
- **ALBERT** provides a good balance between accuracy and model efficiency but is slightly behind the others in raw accuracy.

RoBERTa and ALBERT Columns:

- These likely represent either per-class recall or precision scores for classes related to RoBERTa and ALBERT models or could be comparative metrics (it's a bit ambiguous without additional context).

- Both RoBERTa and ALBERT show high values (around 0.88 -- 0.97), suggesting strong performance in these metrics for each corresponding model.

6. CONCLUSION

- The evaluation of multiple transformer-based models on your classification task reveals that **BERT** is the top performer, achieving the highest overall accuracy of **92.88%**. This confirms BERT's effectiveness in understanding and classifying your data accurately.
- **RoBERTa** and **DistilBERT** follow closely behind, demonstrating competitive accuracies with some trade-offs: RoBERTa excels in recall for certain classes (notably class 3), indicating stronger ability to identify specific categories, while DistilBERT offers faster inference times, making it a great choice for real-time or resource-constrained scenarios.
- **ALBERT**, despite having the lowest accuracy among the four (91.82%), still performs robustly and is attractive where model size and computational efficiency are priorities.
- The high recall and precision scores observed for RoBERTa and ALBERT across classes (ranging roughly between 0.88 and 0.97) highlight their reliability and strong classification capabilities.

Additional Features

Beyond core classification, the project incorporates several advanced NLP and MLOps capabilities to enhance functionality and content understanding:

- **Summarization with GPT-4o**
  Leveraging the GPT-4o model in the backend, the system generates **concise summaries** of input text (articles or descriptions). This allows users to quickly grasp the essence of the content, improving readability and comprehension.

- **Headline Optimization**
  GPT-4o is also used to **refine or improve titles/headlines**, making them

more **clear, engaging, and relevant**. This feature is especially useful for content creators and editors aiming to enhance **SEO** and user engagement.

- **ETL on AWS Lambda**
  A **serverless ETL (Extract, Transform, Load) pipeline** was implemented using **AWS Lambda**, enabling lightweight and scalable preprocessing. This step includes cleaning, transforming, and feature engineering before the data is sent to the classifier or stored in the backend.

- **Model Deployment on SageMaker**
  The multi-class classification model was deployed using **Amazon SageMaker**, achieving an impressive **94% accuracy**. SageMaker ensures robust and scalable deployment for real-time inference in production environments.

- **Frontend in React**
  A dynamic and responsive **React.js** frontend was built to allow users to input article data, view predicted categories, summaries, and optimized headlines in a user-friendly interface.

- **Backend in FastAPI**
  The backend was developed using **FastAPI**, a high-performance Python framework. It serves APIs for text classification, GPT-4o-based summarization, title enhancement, and coordinates communication with AWS services .

CHALLENGES FACED

Despite the success of the project several technical and operational challenges were encountered across different layers of the stack:

Integration with OpenAI GPT-4o

- **Rate Limiting & API Costs**: Managing OpenAI's rate limits and token usage required careful batching and caching strategies to stay within quota and control costs.

- **Latency in Responses**: The GPT-4o API occasionally returned responses with high latency, which affected real-time performance. This was mitigated with asynchronous request handling and loading indicators on the frontend.

- **Response Standardization**: Ensuring consistent formatting in summaries and improved headlines required prompt engineering and post-processing, as the model responses varied depending on the input structure.

Backend & API Layer (FastAPI)

- **CORS & Request Handling**: Establishing smooth communication between the React frontend and FastAPI backend required resolving CORS (Cross-Origin Resource Sharing) issues and setting up secure, asynchronous endpoints.

- **Concurrency & Performance**: FastAPI had to efficiently manage multiple user requests, including classification, summarization, and headline improvement. Uvicorn + Gunicorn were used with worker scaling to maintain performance.

Frontend Development (React)

- **Dynamic UI Updates**: Displaying classification results, summaries, and optimized headlines in real-time required efficient state management and API coordination.

- **Error Handling UX**: Providing meaningful feedback to users when GPT or model APIs failed (e.g., timeouts, missing responses) required robust client-side error handling.

Model Training & Tuning

- **Multi-class Complexity**: Training a high-performing model across multiple categories required extensive hyperparameter tuning, validation strategies, and class balance techniques.

- **Generalization**: Ensuring the model performed well on unseen articles meant avoiding overfitting and continuously monitoring validation metrics.

AWS Lambda ETL Pipeline

- **Cold Start Delays**: Lambda cold starts introduced latency during the first request or after inactivity, affecting real-time preprocessing.

- **Memory/Timeout Limits**: Transforming large inputs or processing batches occasionally hit execution time and memory limits, which required optimization of the Lambda function.
